

Guia do Desenvolvedor

# AWS SDK para Ruby



# AWS SDKpara Ruby: Guia do Desenvolvedor

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

As marcas comerciais e imagens de marcas da Amazon não podem ser usadas no contexto de nenhum produto ou serviço que não seja da Amazon, nem de qualquer maneira que possa gerar confusão entre os clientes ou que deprecie ou desprestige a Amazon. Todas as outras marcas comerciais que não pertencem à Amazon pertencem a seus respectivos proprietários, que podem ou não ser afiliados, patrocinados pela Amazon ou ter conexão com ela.

---

# Table of Contents

O que é o AWS SDK para Ruby? .....	1
Documentação e recursos adicionais .....	1
Implantação no AWS Cloud .....	2
Manutenção e suporte para as versões principais do SDK .....	2
Conceitos básicos .....	3
Autenticação do SDK com a AWS .....	3
Iniciar uma sessão do portal de acesso da AWS .....	4
Mais informações de autenticação .....	5
Instalar o SDK .....	6
Pré-requisitos .....	6
Instalar o SDK .....	6
Tutorial de boas-vindas .....	7
Escreva o código .....	7
Execução do programa .....	8
Observação para usuários do Windows .....	9
Próximas etapas .....	9
Use AWS Cloud9 com o SDK .....	9
Etapa 1: configure seu Conta da AWS para usar AWS Cloud9 .....	10
Etapa 2: configure o ambiente de desenvolvimento do AWS Cloud9 .....	10
Etapa 3: configurar o AWS SDK para Ruby .....	11
Etapa 4: faça o download do código de exemplo .....	12
Etapa 5: executar o código de exemplo .....	13
Configurar o SDK .....	15
Cadeia de provedores de credenciais .....	15
Criação de um token de AWS STS acesso .....	17
Definir uma região .....	17
Definir a região usando o arquivo config compartilhado .....	18
Definir a região usando variáveis de ambiente .....	18
Definir a região com o <code>Aws.config</code> .....	18
Definir a região em um objeto de cliente ou de recurso .....	19
Definir um endpoint não padrão .....	19
Use o SDK .....	20
Usar o utilitário REPL .....	20
Pré-requisitos .....	20

Configuração do Bundler .....	21
Executar o REPL .....	21
Use o SDK com Ruby on Rails .....	22
Dica de depuração: obtenha informações de rastreamento de comunicação de um cliente .....	22
Simular respostas e erros de clientes .....	23
Simular respostas de clientes .....	23
Simular de erros de clientes .....	25
Paginação .....	25
As respostas paginadas são enumeráveis .....	25
Manipulação manual de respostas paginadas .....	26
Classes de dados paginadas .....	26
Waiters .....	26
Invocar um waiter .....	27
Falhas de espera .....	27
Configurar um waiter .....	28
Estender um waiter .....	28
Especificar comportamento de nova tentativa do cliente .....	29
Migrar da versão 1 ou 2 para a versão 3 do AWS SDK para Ruby .....	30
Uso lado a lado .....	30
Diferenças gerais .....	30
Diferenças de clientes .....	31
Diferenças de recursos .....	32
Exemplos de código .....	34
Aurora .....	35
Auto Scaling .....	36
CloudTrail .....	38
Ações .....	38
CloudWatch .....	42
Ações .....	38
Provedor de identidade do Amazon Cognito .....	54
Amazon Comprehend .....	56
Cenários .....	56
Amazon DocumentDB .....	57
Exemplos sem servidor .....	57
DynamoDB .....	58
Conceitos básicos .....	60

---

Ações .....	38
Cenários .....	56
Exemplos sem servidor .....	57
Amazon EC2 .....	87
Ações .....	38
Elastic Beanstalk .....	123
Ações .....	38
EventBridge .....	129
Cenários .....	56
AWS Glue .....	150
Conceitos básicos .....	60
Ações .....	38
IAM .....	180
Ações .....	38
Cenários .....	56
Kinesis .....	237
Exemplos sem servidor .....	57
AWS KMS .....	239
Ações .....	38
Lambda .....	243
Ações .....	38
Cenários .....	56
Exemplos sem servidor .....	57
Amazon MSK .....	273
Exemplos sem servidor .....	57
Amazon Polly .....	274
Ações .....	38
Cenários .....	56
Amazon RDS .....	278
Ações .....	38
Exemplos sem servidor .....	57
Amazon S3 .....	286
Conceitos básicos .....	60
Ações .....	38
Cenários .....	56
Exemplos sem servidor .....	57

Amazon SES .....	318
Ações .....	38
Amazon SES API v2 .....	324
Ações .....	38
Amazon SNS .....	325
Ações .....	38
Exemplos sem servidor .....	57
Amazon SQS .....	335
Ações .....	38
Exemplos sem servidor .....	57
AWS STS .....	348
Ações .....	38
Amazon Textract .....	350
Cenários .....	56
Amazon Translate .....	351
Cenários .....	56
Amazon WorkDocs .....	352
Ações .....	38
Segurança .....	356
Proteção de dados .....	356
Identity and Access Management .....	357
Compliance Validation .....	358
Resiliência .....	359
Segurança da infraestrutura .....	359
Aplicando uma versão mínima TLS .....	360
Verificando a SSL versão aberta .....	360
Suporte de atualização TLS .....	361
Migração do cliente de criptografia do S3 .....	361
Visão geral da migração .....	361
Atualizar os clientes existentes para ler novos formatos .....	361
Migrar clientes de criptografia e descriptografia para a V2 .....	363
Histórico do documento .....	367
.....	ccclxix

# O que é o AWS SDK para Ruby?

Bem-vindo ao Guia do desenvolvedor do AWS SDK para Ruby. O AWS SDK para Ruby fornece bibliotecas de suporte para quase todos os Serviços da AWS, incluindo Amazon Simple Storage Service (Amazon S3), Amazon Elastic Compute Cloud (Amazon EC2) e Amazon DynamoDB.

O Guia do desenvolvedor do AWS SDK para Ruby fornece informações sobre como instalar, configurar e usar o AWS SDK para Ruby para criar aplicativos Ruby que usam Serviços da AWS.

## [Conceitos básicos sobre AWS SDK para Ruby](#)

## Documentação e recursos adicionais

Para acessar mais recursos para desenvolvedores do AWS SDK para Ruby, consulte as informações a seguir:

- [Guia de referência de ferramentas e SDKs da AWS](#): contém configurações, recursos e outros conceitos fundamentais comuns entre SDKs da AWS
- [Referência da API do AWS SDK for Ruby - Versão 3](#)
- [Repositório de exemplos de código da AWS](#) no GitHub
- [RubyGems.org](#): a versão mais recente do SDK é modularizada em gems específicos de serviços disponíveis aqui
  - [Serviços compatíveis](#): lista todos os gems compatíveis com o AWS SDK para Ruby
- Fonte do AWS SDK para Ruby no GitHub:
  - [Fonte e README](#)
  - [Alterar logs em cada gem](#)
  - [Atualização da v2 para a v3](#)
  - [Problemas](#)
  - [Observações sobre a atualização de núcleo](#)
- [Blog de desenvolvedores](#)
- [Canal Gitter](#)
- [@awsforruby](#) no Twitter

## Implantação no AWS Cloud

É possível usar Serviços da AWS como AWS Elastic Beanstalk, AWS OpsWorks e AWS CodeDeploy para implantar o aplicativo no AWS Cloud. Para a implantação de aplicativos Ruby com Elastic Beanstalk, consulte [Implantação de aplicativos Elastic Beanstalk no Ruby usando EB CLI e Git](#) no Guia do desenvolvedor do AWS Elastic Beanstalk. Para implantar um aplicativo Ruby on Rails com o AWS OpsWorks, consulte [Implantação de aplicativos Ruby on Rails no AWS OpsWorks](#). Para obter uma visão geral dos serviços de implantação da AWS, consulte [Visão geral das opções de implantação na AWS](#).

## Manutenção e suporte para as versões principais do SDK

Para obter informações sobre manutenção e suporte para versões principais do SDK e suas dependências subjacentes, consulte o seguinte no [Guia de referência de AWS SDKs e ferramentas](#):

- [Política de manutenção de ferramentas e SDKs da AWS](#)
- [Matriz de suporte a versões de ferramentas e SDKs da AWS](#)



# Conceitos básicos sobre AWS SDK para Ruby

Saiba como instalar, configurar e usar o SDK para criar um aplicativo Ruby para acessar um recurso da AWS de forma programada.

## Tópicos

- [Autenticação do SDK com a AWS](#)
- [Instalar o AWS SDK para Ruby](#)
- [Tutorial de boas-vindas para o AWS SDK para Ruby](#)
- [Use AWS Cloud9 com o AWS SDK para Ruby](#)

## Autenticação do SDK com a AWS

Você precisa estabelecer como seu código deve ser autenticado com a AWS ao desenvolver com Serviços da AWS. É possível configurar o acesso programático aos recursos da AWS de maneiras diferentes, dependendo do ambiente e do acesso da AWS disponível para você.

Para escolher seu método de autenticação e configurá-lo para o SDK, consulte [Autenticação e acesso](#) no Guia de referência de ferramentas e SDKs da AWS.

Recomendamos que novos usuários que estão desenvolvendo localmente e não receberam um método de autenticação do empregador configurem o AWS IAM Identity Center. Esse método inclui a instalação da AWS CLI para facilitar a configuração e entrar regularmente no portal de acesso da AWS. Se você escolher esse método, seu ambiente deverá conter os seguintes elementos depois de concluir o procedimento de [autenticação do IAM Identity Center](#) no Guia de referência de ferramentas e SDKs da AWS:

- A AWS CLI, que você usa para iniciar uma sessão do portal de acesso da AWS antes de executar a aplicação.
- Um [arquivo `AWSconfig` compartilhado](#) com um perfil de `[default]` com um conjunto de valores de configuração que podem ser referenciados a partir do SDK. Para encontrar a localização desse arquivo, consulte [Localização dos arquivos compartilhados](#) no Guia de referência de ferramentas e SDKs da AWS.
- O arquivo `config` compartilhado define a configuração do [region](#). Isso define o Região da AWS padrão que o SDK usa para solicitações da AWS. Essa região é usada para solicitações de serviço do SDK que não são fornecidas com uma Região específica para uso.

- O SDK usa a [configuração do provedor do token de SSO](#) do perfil para adquirir credenciais antes de enviar solicitações para a AWS. O valor `sso_role_name`, que é uma função do IAM conectada a um conjunto de permissões do IAM Identity Center, deve permitir o acesso aos Serviços da AWS usados na aplicação.

O arquivo config de amostra a seguir mostra um perfil padrão configurado com o provedor de token de SSO. A configuração `sso_session` do perfil se refere à [seção do sso-session](#). A seção `sso-session` contém configurações para iniciar uma sessão do portal de acesso da AWS.

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

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

O AWS SDK para Ruby não precisa que pacotes adicionais (como SSO e SSO0IDC) sejam adicionados ao seu aplicativo para usar a autenticação do IAM Identity Center.

## Iniciar uma sessão do portal de acesso da AWS

Antes de executar um aplicativo que acessa os Serviços da AWS, você precisa de uma sessão ativa do portal de acesso da AWS para que o SDK use a autenticação do IAM Identity Center para resolver as credenciais. Dependendo da duração da sessão configurada, o seu acesso acabará expirando e o SDK encontrará um erro de autenticação. Para entrar no portal de acesso da AWS, execute o seguinte comando na AWS CLI.

```
aws sso login
```

Se você seguiu as orientações e tem um perfil padrão configurado, não precisará chamar o comando com uma opção de `--profile`. Se a configuração do provedor de token de SSO estiver usando um perfil nomeado, o comando será `aws sso login --profile named-profile`.

Para, como opção, testar se você já tem uma sessão ativa, execute o seguinte comando AWS CLI.

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

Se a sua sessão estiver ativa, a resposta a este comando relata a conta do IAM Identity Center e o conjunto de permissões configurados no arquivo `config` compartilhado.

### Note

Se você já tiver uma sessão ativa do portal de acesso da AWS e executar `aws sso login`, não será necessário fornecer credenciais.

O processo de login pode solicitar que você permita que a AWS CLI acesse seus dados. Como a AWS CLI é construída sobre o SDK para Python, as mensagens de permissão podem conter variações do nome do `botocore`.

## Mais informações de autenticação

Os usuários humanos, também conhecidos como identidades humanas, são as pessoas, os administradores, os desenvolvedores, os operadores e os consumidores de suas aplicações. Eles devem ter uma identidade para acessar seus ambientes e aplicações da AWS. Usuários humanos que são membros da sua organização (ou seja, você, o desenvolvedor) são conhecidos como identidades da força de trabalho.

Use credenciais temporárias ao acessar a AWS. Você pode usar um provedor de identidade para que seus usuários humanos recebam acesso federado a contas da AWS, assumindo funções que fornecem credenciais temporárias. Para gerenciamento de acesso centralizado, recomendamos que você use o AWS IAM Identity Center (IAM Identity Center) para gerenciar o acesso às suas contas e as permissões nessas contas. Para obter mais alternativas, consulte as informações a seguir.

- Para saber mais sobre as práticas recomendadas, consulte [Práticas recomendadas de segurança no IAM](#) no Guia do usuário do IAM.
- Para criar credenciais de curto prazo da AWS, consulte [Credenciais de segurança temporárias](#) no Guia do usuário do IAM.
- Para saber mais sobre outros provedores de credenciais AWS SDK para Ruby, consulte [Provedores de credenciais padronizados](#) no Guia de referência de ferramentas e SDKs da AWS.

# Instalar o AWS SDK para Ruby

Esta seção inclui pré-requisitos e instruções de instalação do AWS SDK para Ruby.

## Pré-requisitos

Antes de usar o AWS SDK para Ruby, você deve fazer a autenticação com a AWS. Para obter informações sobre a configuração da autenticação, consulte [Autenticação do SDK com a AWS](#).

## Instalar o SDK

Você pode instalar o AWS SDK para Ruby como faria com qualquer gem do Ruby. Os gems estão disponíveis em [RubyGems](#). O AWS SDK para Ruby foi projetado para ser modular e é separado por Serviço da AWS. A instalação de todo o gem `aws-sdk` é grande e pode levar mais de uma hora.

Recomendamos instalar apenas os gems para os Serviços da AWS que você usa. Eles são nomeados como `aws-sdk-service_abbreviation` e a lista completa pode ser encontrada na tabela [Serviços compatíveis](#) do arquivo README do AWS SDK para Ruby. Por exemplo, o gem para interface com o serviço Amazon S3 está disponível diretamente em [aws-sdk-s3](#).

## Gerenciador de versões do Ruby

Em vez de usar o sistema Ruby, recomendamos usar um gerenciador de versões do Ruby, como o seguinte:

- [RVM](#)
- [chruby](#)
- [rbenv](#)

Por exemplo, se você estiver usando um sistema operacional Amazon Linux 2, os comandos a seguir podem ser usados para atualizar o RVM, listar as versões disponíveis do Ruby e, em seguida, escolher a versão que você deseja usar para desenvolvimento com o AWS SDK para Ruby. A versão mínima do Ruby necessária é a 2.3.

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

## Bundler

Se você usa o [Bundler](#), os comandos a seguir instalam a gem AWS SDK para Ruby para o Amazon S3:

1. Instale o Bundler e crie o Gemfile:

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

2. Abra o Gemfile criado e adicione uma linha de gem para cada gem de serviço da AWS que seu código usará. Para acompanhar o exemplo do Amazon S3, adicione a seguinte linha ao final do arquivo:

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

3. Salve o Gemfile.
4. Instale as dependências especificadas em seu Gemfile:

```
$ bundle install
```

## Tutorial de boas-vindas para o AWS SDK para Ruby

Diga olá ao Amazon S3 usando o AWS SDK para Ruby. O exemplo a seguir exibe uma lista dos buckets do Amazon S3.

### Escreva o código

Copie e cole o código a seguir em um novo arquivo fonte. Nomeie o arquivo `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
end
```

```
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__
```

O AWS SDK para Ruby foi projetado para ser modular e é separado por Serviço da AWS. Depois que o gem é instalado, a instrução do `require` na parte superior do arquivo fonte do Ruby importa as classes e os métodos do AWS SDK para o serviço Amazon S3. Para obter uma lista completa dos gems de serviço da AWS disponíveis, consulte a tabela [Serviços compatíveis](#) do arquivo README do AWS SDK para Ruby.

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

## Execução do programa

Abra um prompt de comando para executar seu programa Ruby. A sintaxe de comando típica para executar um programa Ruby é:

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

Esse código de amostra não usa argumentos. Para executar esse código, insira o seguinte no prompt de comando:

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

## Observação para usuários do Windows

Quando você usar certificados SSL no Windows e executar o código Ruby, verá um erro semelhante ao seguinte.

```
C:\Ruby>ruby buckets.rb
C:/Ruby200-x64/lib/ruby/2.0.0/net/http.rb:921:in `connect': SSL_connect returned=1
errno=0 state=SSLv3 read server certificate B: certificate verify failed
(Seahorse::Client::NetworkingError)
    from C:/Ruby200-x64/lib/ruby/2.0.0/net/http.rb:921:in `block in connect'

    from C:/Ruby200-x64/lib/ruby/2.0.0/timeout.rb:66:in `timeout'
    from C:/Ruby200-x64/lib/ruby/2.0.0/net/http.rb:921:in `connect'
    from C:/Ruby200-x64/lib/ruby/2.0.0/net/http.rb:862:in `do_start'
    from C:/Ruby200-x64/lib/ruby/2.0.0/net/http.rb:857:in `start'
...

```

Para corrigir esse problema, adicione a seguinte linha ao seu arquivo fonte Ruby em algum lugar antes da primeira chamada da AWS.

```
Aws.use_bundled_cert!
```

Se você estiver usando somente o gem `aws-sdk-s3` em seu programa Ruby e se deseja usar o certificado incluso, também precisará adicionar o gem `aws-sdk-core`.

## Próximas etapas

Para testar muitas outras operações do Amazon S3, confira o [Repositório de exemplos de AWS código](#) em GitHub

## Use AWS Cloud9 com o AWS SDK para Ruby

AWS Cloud9 é um ambiente de desenvolvimento integrado baseado na Web (IDE) que contém uma coleção de ferramentas que você usa para codificar, criar, executar, testar, depurar e lançar software

na nuvem. Você pode usar AWS Cloud9 com o AWS SDK for Ruby para escrever e executar seu código Ruby usando um navegador. AWS Cloud9 inclui ferramentas como editor de código e terminal. Como AWS Cloud9 IDE é baseado na nuvem, você pode trabalhar em seus projetos no escritório, em casa ou em qualquer lugar usando uma máquina conectada à Internet. Para obter informações gerais sobre AWS Cloud9, consulte o [Guia AWS Cloud9 do usuário](#).

Siga estas instruções para configurar AWS Cloud9 com o AWS SDK for Ruby:

- [Etapa 1: configure seu Conta da AWS para usar AWS Cloud9](#)
- [Etapa 2: configure o ambiente de desenvolvimento do AWS Cloud9](#)
- [Etapa 3: configurar o AWS SDK para Ruby](#)
- [Etapa 4: baixar o código de exemplo](#)
- [Etapa 5: executar o código de exemplo](#)

## Etapa 1: configure seu Conta da AWS para usar AWS Cloud9

Para usar AWS Cloud9, entre no AWS Cloud9 console a partir do AWS Management Console.

### Note

Se você estiver usando AWS IAM Identity Center para se autenticar, talvez seja necessário adicionar a permissão necessária de `iam:ListInstanceProfilesForRole` à política anexada pelo usuário no IAM console.

Para configurar uma IAM entidade em sua AWS conta para acessar AWS Cloud9 e entrar no AWS Cloud9 console, consulte [Configuração da equipe AWS Cloud9](#) no Guia do AWS Cloud9 usuário.

## Etapa 2: configure o ambiente de desenvolvimento do AWS Cloud9

Depois de entrar no AWS Cloud9 console, use o console para criar um ambiente de AWS Cloud9 desenvolvimento. Depois de criar o ambiente, AWS Cloud9 abre o IDE para esse ambiente.

Para obter detalhes, consulte [Criação de um ambiente no AWS Cloud9](#) no Guia do usuário do AWS Cloud9 .



### Note

Ao criar seu ambiente no console pela primeira vez, recomendamos que você escolha a opção Criar uma nova instância para environment (EC2). Essa opção diz AWS Cloud9 para criar um ambiente, iniciar uma EC2 instância da Amazon e, em seguida, conectar a nova instância ao novo ambiente. Essa é a maneira mais rápida de começar a usar AWS Cloud9.

Se o terminal ainda não estiver aberto no IDE, abra-o. Na barra de menu do IDE, escolha Janela, Novo terminal. Você pode usar a janela do terminal para instalar ferramentas e criar aplicações.

## Etapa 3: configurar o AWS SDK para Ruby

Depois de AWS Cloud9 abrir o IDE para seu ambiente de desenvolvimento, use a janela do terminal para configurar o AWS SDK para Ruby em seu ambiente.

Você pode instalar o AWS SDK for Ruby como faria com qualquer gem do Ruby. As joias estão disponíveis em [RubyGems](#). O AWS SDK for Ruby foi projetado para ser modular e é separado por Serviço da AWS. A instalação de todo o gem `aws-sdk` é grande e pode levar mais de uma hora.

Recomendamos instalar apenas as gemas Serviços da AWS que você usa. Eles são nomeados como `aws-sdk-service_abbreviation` e a lista completa é encontrada na tabela [Serviços suportados](#) do README arquivo AWS SDK for Ruby. Por exemplo, o gem para interface com o serviço Amazon S3 está disponível diretamente em [aws-sdk-s3](#).

## Gerenciador de versões do Ruby

Em vez de usar o sistema Ruby, recomendamos usar um gerenciador de versões do Ruby, como o seguinte:

- [RVM](#)
- [chruby](#)
- [rbenv](#)

Por exemplo, se você estiver usando um sistema operacional Amazon Linux 2, os comandos a seguir podem ser usados para atualizar RVM, listar as versões disponíveis do Ruby e, em seguida, escolher a versão que você deseja usar para desenvolvimento com o AWS SDK for Ruby. A versão mínima do Ruby necessária é a 2.3.

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

## Bundler

Se você usa o [Bundler](#), os seguintes comandos instalam a gem AWS SDK for Ruby para Amazon S3:

1. Instale o Bundler e crie o Gemfile:

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

2. Abra o criado Gemfile e adicione uma gem linha para cada gem AWS de serviço que seu código usará. Para acompanhar o exemplo do Amazon S3, adicione a seguinte linha ao final do arquivo:

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

3. Salve o Gemfile.
4. Instale as dependências especificadas em seu Gemfile:

```
$ bundle install
```

## Etapa 4: faça o download do código de exemplo

Use a janela do terminal para baixar o código de exemplo do AWS SDK for Ruby no ambiente de AWS Cloud9 desenvolvimento.

Para baixar uma cópia de todos os exemplos de código usados na AWS SDK documentação oficial no diretório raiz do seu ambiente, execute o seguinte comando:

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

Os exemplos de código AWS SDK para o Ruby estão localizados no `ENVIRONMENT_NAME/aws-doc-sdk-examples/ruby` diretório, onde `ENVIRONMENT_NAME` está o nome do seu ambiente de desenvolvimento.

Para continuar usando um exemplo do Amazon S3, recomendamos começar com um `ENVIRONMENT_NAME/aws-doc-sdk-examples/ruby/example_code/s3/bucket_list.rb` de exemplo de código. Use a janela do terminal para navegar até o diretório `s3` e listar os arquivos.

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

Para abrir o arquivo AWS Cloud9, você pode clicar `bucket_list.rb` diretamente na janela do terminal.

## Etapa 5: executar o código de exemplo

Para executar código em seu ambiente de AWS Cloud9 desenvolvimento, escolha o botão Executar na barra de menu superior. AWS Cloud9 detectará automaticamente a extensão do `.rb` arquivo e usará o executor Ruby para executar o código. Para obter mais informações sobre como executar código em AWS Cloud9, consulte [Executar seu código](#) no Guia AWS Cloud9 do usuário.

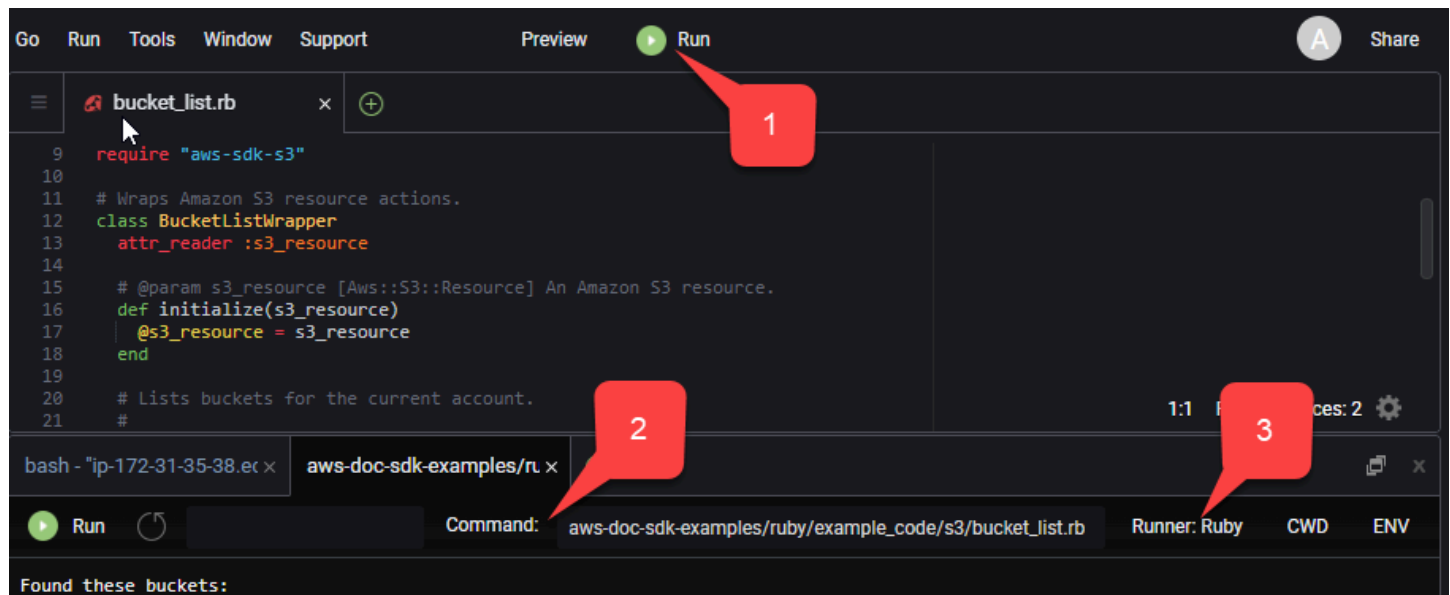
Na captura de tela a seguir, observe essas áreas básicas:

- 1: Executar. O botão Executar está localizado na barra de menu superior. Isso abre uma nova guia para seus resultados.

### Note

Você também pode criar manualmente novas configurações de execução. Na barra de menus, selecione Executar, Configurações de execução, Nova configuração de execução.

- 2: Comando. AWS Cloud9 preenche a caixa de texto Comando com o caminho e o nome do arquivo executado. Se seu código espera que os parâmetros da linha de comando sejam passados, eles podem ser adicionados à linha de comando da mesma forma que você faria ao executar o código em uma janela do terminal.
- 3: Corredor. AWS Cloud9 detecta que sua extensão de arquivo é `.rb` e seleciona o Ruby Runner para executar seu código.



Qualquer saída gerada pelo código em execução é exibida na guia.

Para testar muitas outras operações do Amazon S3, confira o [Repositório de exemplos de AWS código](#) em GitHub

# Configurar o AWS SDK para Ruby

Saiba como configurar o AWS SDK para Ruby. Você precisa estabelecer como seu código deve ser autenticado com a AWS ao desenvolver com os Serviços da AWS. Você também deve definir o que Região da AWS deseja usar.

## Cadeia de provedores de credenciais

Todos SDKs têm uma série de locais (ou fontes) que eles verificam para obter credenciais válidas para usar para fazer uma solicitação a um Serviço da AWS. Depois que as credenciais válidas são encontradas, a pesquisa é interrompida. Essa busca sistemática é chamada de cadeia de provedores de credenciais padrão.

Para cada etapa da cadeia, há várias maneiras de atribuir os valores. A definição de valores diretamente no código sempre tem precedência, seguida pela configuração como variáveis de ambiente e, em seguida, no AWS `config` arquivo compartilhado. Para obter mais informações, consulte [Precedência de configurações](#) no Guia AWS SDKs de referência de ferramentas.

O Guia de Referência de Ferramentas AWS SDKs e Ferramentas tem informações sobre as SDK configurações usadas por todos AWS SDKs e pelo AWS CLI. Para saber mais sobre como configurar o SDK por meio do AWS `config` arquivo compartilhado, consulte Arquivos de [configuração e credenciais compartilhados](#). Para saber mais sobre como configurar o SDK por meio da definição de variáveis de ambiente, consulte [Suporte a variáveis de ambiente](#).

Para se autenticar AWS, o AWS SDK for Ruby verifica os provedores de credenciais na ordem listada na tabela a seguir.

Provedor de credenciais por precedência	AWS SDKs Guia de referência de ferramentas e ferramentas	Referência ao AWS SDK for Ruby API do
AWS chaves de acesso (credenciais temporárias e de longo prazo)	<a href="#">AWS teclas de acesso</a>	<a href="#">Aws::Credentials</a> <a href="#">Aws::SharedCredentials</a>
Token de identidade da Web de AWS Security Token Service (AWS STS)	<a href="#">Assuma a função de provedor de credenciais</a>	<a href="#">Aws::AssumeRoleWebIdentityCredentials</a>

Provedor de credenciais por precedência	AWS SDKs Guia de referência de ferramentas e ferramentas	Referência ao AWS SDK for Ruby API do
	Usando <code>role_arn</code> , <code>role_session_name</code> e <code>web_identity_token_file</code>	
AWS IAM Identity Center. Neste guia, consulte <a href="#">Autenticação do SDK com a AWS</a> .	<a href="#">IAM Provedor de credenciais do Identity Center</a>	<a href="#">Aws::SSOCredentials</a>
Provedor de entidades confiável (como <code>AWS_ROLE_ARN</code> ). Neste guia, consulte <a href="#">Criação de um token de AWS STS acesso</a> .	<a href="#">Assuma a função de provedor de credenciais</a>  Usando <code>role_arn</code> e <code>role_session_name</code>	<a href="#">Aws::AssumeRoleCredentials</a>
Provedor de credenciais de processo	<a href="#">Provedor de credenciais de processo</a>	<a href="#">Aws::ProcessCredentials</a>
Credenciais do Amazon Elastic Container Service (Amazon ECS)	<a href="#">Provedor de credenciais de contêiner</a>	<a href="#">Aws::ECSCredentials</a>
Credenciais de perfil de instância do Amazon Elastic Compute Cloud (Amazon EC2) (provedor de IMDS credenciais)	<a href="#">IMDS provedor de credenciais</a>	<a href="#">Aws::InstanceProfileCredentials</a>

Se a variável de ambiente AWS SDK for Ruby `AWS_SDK_CONFIG_OPT_OUT` for definida, o AWS config arquivo compartilhado, normalmente em `~/.aws/config`, não será analisado em busca de credenciais.

Se você seguiu a abordagem recomendada para novos usuários começarem, configurou a autenticação do AWS IAM Identity Center durante a [Autenticação do SDK com a AWS](#) do tópico Conceitos básicos. Outros métodos de autenticação são úteis para situações diferentes. Para

evitar riscos de segurança, recomendamos sempre usar credenciais de curto prazo. Para outros procedimentos de método de autenticação, consulte [Autenticação e acesso](#) no AWS SDKs Guia de referência de ferramentas.

## Criação de um token de AWS STS acesso

Assumir uma função envolve o uso de um conjunto de credenciais de segurança temporárias que você pode usar para acessar AWS recursos aos quais você normalmente não teria acesso. Essas credenciais de segurança temporárias consistem em um ID de chave de acesso, uma chave de acesso secreta e um token de segurança. Você pode usar o método [Aws::AssumeRoleCredentials](#) para criar um token de acesso do AWS Security Token Service (AWS STS).

O exemplo a seguir usa um token de acesso para criar um objeto cliente do Amazon S3, onde `linked::account::arn` é o Amazon Resource Name (ARN) da função a ser assumida e `session-name` é um identificador para a sessão da função assumida.

```
role_credentials = Aws::AssumeRoleCredentials.new(  
  client: Aws::STS::Client.new,  
  role_arn: "linked::account::arn",  
  role_session_name: "session-name"  
)  
  
s3 = Aws::S3::Client.new(credentials: role_credentials)
```

Para obter mais informações sobre como configurar `role_arn` ou `role_session_name`, ou sobre como configurá-las usando o AWS `config` arquivo compartilhado, consulte [Assumir a função de provedor de credenciais](#) no Guia de referência de ferramentas AWS SDKs e ferramentas.

## Definir uma região

Você precisa definir uma região ao usar a maioria dos Serviços da AWS. O AWS SDK for Ruby pesquisa uma região na seguinte ordem:

1. [Definir a região em um objeto de cliente ou de recurso](#)
2. [Definir a região usando o `Aws.config`](#)
3. [Definir a região usando variáveis de ambiente](#)
4. [Definir a região usando o arquivo `config` compartilhado](#)

Para obter mais informações sobre a região configuração, consulte o Guia [Região da AWS](#) de referência de ferramentas AWS SDKs e ferramentas. O restante desta seção descreve como definir uma região, começando com a abordagem mais comum.

## Definir a região usando o arquivo **config** compartilhado

Defina a região definindo a `region` variável no AWS `config` arquivo compartilhado. Para obter mais informações sobre o `config` arquivo compartilhado, consulte [Arquivos de configuração e credenciais compartilhados no Guia](#) de referência de ferramentas AWS SDKs e ferramentas.

Exemplo de configuração desse valor no arquivo `config`:

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

O arquivo `config` compartilhado não é verificado se a variável de ambiente `AWS_SDK_CONFIG_OPT_OUT` estiver definida.

## Definir a região usando variáveis de ambiente

Defina a região definindo a variável de ambiente `AWS_REGION`.

Use o comando `export` para definir essa variável em sistemas baseados em Unix, como Linux ou macOS. O exemplo a seguir define a região como `us-west-2`.

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

Para configurar essa variável no Windows, use o comando `set`. O exemplo a seguir define a região como `us-west-2`.

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

## Definir a região com o **Aws.config**

Defina a região adicionando um valor `region` ao hash `Aws.config`. O exemplo a seguir atualiza o hash `Aws.config` para usar a região `us-west-1`.

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



Todos os clientes ou recursos que você criar posteriormente estarão vinculados a essa região.

## Definir a região em um objeto de cliente ou de recurso

Defina a região ao criar um AWS cliente ou recurso. O exemplo a seguir cria um objeto de recurso do Amazon S3 na região `us-west-1`. Escolha a região correta para seus AWS recursos. Um objeto de cliente de serviço é imutável, então você deve criar um novo cliente para cada serviço para o qual você faz solicitações e para fazer solicitações ao mesmo serviço usando uma configuração diferente.

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

## Definir um endpoint não padrão

A região é usada para construir um SSL endpoint a ser usado para AWS solicitações. Se você precisar usar um endpoint não padrão na região selecionada, adicione uma entrada de `endpoint` ao `Aws.config`. Como alternativa, defina o `endpoint`: ao criar um cliente de serviço ou objeto de recurso. O exemplo a seguir cria um objeto de recurso do Amazon S3 no endpoint `other_endpoint`.

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

Para usar um endpoint de sua escolha para API solicitações e fazer com que essa escolha persista, consulte a opção de configuração de [endpoints específicos do serviço](#) no Guia de referência de ferramentas e ferramentas.AWS SDKs

# Use o AWS SDK para Ruby

Esta seção fornece informações sobre o desenvolvimento de software com o AWS SDK for Ruby, incluindo como usar alguns dos recursos avançados SDK do.

O [Guia de Referência de Ferramentas AWS SDKs e Ferramentas](#) também contém configurações, recursos e outros conceitos fundamentais comuns entre muitos dos AWS SDKs.

## Tópicos

- [Usar o utilitário REPL do AWS SDK para Ruby](#)
- [Use o SDK com Ruby on Rails](#)
- [Dica de depuração: obtenha informações de rastreamento de comunicação de um cliente](#)
- [Simular respostas e erros de clientes](#)
- [Paginação](#)
- [Waiters](#)
- [Especificar comportamento de nova tentativa do cliente](#)
- [Migrar da versão 1 ou 2 para a versão 3 do AWS SDK para Ruby](#)

## Usar o utilitário REPL do AWS SDK para Ruby

O gem `aws-sdk` inclui uma interface de linha de comando interativa Read-Eval-Print-Loop (REPL) na qual você pode testar o SDK para Ruby e ver os resultados imediatamente. Os gems do SDK para Ruby estão disponíveis em [RubyGems.org](http://RubyGems.org).

## Pré-requisitos

- [Instalar o AWS SDK para Ruby](#).
- O `aws-v3.rb` está localizado no gem [aws-sdk-resources](#). O gem `aws-sdk-resources` também foi incluído pelo gem `aws-sdk` principal.
- Você precisará de uma biblioteca xml, como a do gem `rexml`.
- Embora o programa funcione com o Interactive Ruby Shell (IRB) (`irb`), recomendamos que você instale o `gempyr`, que fornece um ambiente REPL mais poderoso.

## Configuração do Bundler

Se você usa o [Bundler](#), as seguintes atualizações em seu Gemfile abordarão os gems de pré-requisito:

1. Abra o Gemfile que você criou quando instalou o AWS SDK para Ruby. Adicione as seguintes linhas ao arquivo:

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

2. Salve o Gemfile.
3. Instale as dependências especificadas em seu Gemfile:

```
$ bundle install
```

## Executar o REPL

É possível acessar o REPL executando o `aws-v3.rb` a partir da linha de comando.

```
aws-v3.rb
```

Como alternativa, você pode habilitar o log de comunicação HTTP definindo a flag de verbose. O log de comunicação HTTP fornece informações sobre a comunicação entre o AWS SDK para Ruby e a AWS. Observe que a flag de verbose também adiciona overhead que pode tornar a execução do seu código mais lenta.

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

O SDK para Ruby inclui classes de cliente que fornecem interfaces para os Serviços da AWS. Cada classe de cliente oferece suporte a um determinado Serviço da AWS. No REPL, cada classe de serviço tem um auxiliar que retorna um novo objeto cliente para interagir com esse serviço. O nome do auxiliar será o nome do serviço convertido em minúsculas. Por exemplo, os nomes dos objetos auxiliares do Amazon S3 e do Amazon EC2 são `s3` e `ec2`, respectivamente. Para listar os buckets do Amazon S3 em sua conta, você pode inserir `s3.list_buckets` no prompt.

Você pode digitar `quit` no prompt do REPL para sair.

## Use o SDK com Ruby on Rails

O [Ruby on Rails](#) fornece uma estrutura de desenvolvimento da Web que facilita a criação de sites com o Ruby.

AWS fornece a `aws-sdk-rails` gema para permitir uma fácil integração com o Rails. Você pode usar AWS Elastic Beanstalk, AWS OpsWorks AWS CodeDeploy, ou o [AWS Rails Provisioner](#) para implantar e executar seus aplicativos Rails na nuvem. AWS

Para obter informações sobre como instalar e usar o `aws-sdk-rails` gem, consulte o GitHub repositório. <https://github.com/aws/aws-sdk-rails>

## Dica de depuração: obtenha informações de rastreamento de comunicação de um cliente

Você pode obter informações de rastreamento de comunicação de um cliente da AWS definindo o `http_wire_trace` booleano. Essas informações ajudam a diferenciar alterações de clientes, problemas de serviços e erros de usuários. Quando `true`, a configuração mostra o que está sendo enviado na comunicação. O exemplo a seguir cria um cliente do Amazon S3 com rastreamento de comunicação habilitado no momento da criação do cliente.

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

Considerando o seguinte código e o argumento `bucket_name`, o resultado exibe uma mensagem que diz se um bucket com esse nome existe.

```
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 o bucket existir, o resultado será semelhante ao seguinte. (Retornos foram adicionados à linha HEAD para legibilidade.)

```
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\r\n"
-> "x-amz-id-2: XxB2J+kpHgTjmMUwpkUI1EjaFSPxAjWRgkn/+z7YwWc/
iAX5E30XRBzJ37cfc8T4D7ELC1KFELM=\r\n\r\n"
-> "x-amz-request-id: 5MD4APQoS815QVBR\r\n\r\n"
-> "Date: Thu, 27 Apr 2023 14:31:47 GMT\r\n\r\n"
-> "x-amz-bucket-region: us-east-1\r\n\r\n"
-> "x-amz-access-point-alias: false\r\n\r\n"
-> "Content-Type: application/xml\r\n\r\n"
-> "Server: AmazonS3\r\n\r\n"
-> "\r\n\r\n"
Conn keep-alive
Bucket bucket_name exists
```

Você também pode ativar o rastreamento de comunicação após a criação do cliente.

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

Para obter mais informações sobre os campos nas informações de rastreamento de comunicação relatadas, consulte os [cabeçalhos de solicitação obrigatórios do Transfer Family](#).

## Simular respostas e erros de clientes

Saiba como simular respostas e erros de clientes em um aplicativo AWS SDK para Ruby.

### Simular respostas de clientes

Quando você simula uma resposta, o AWS SDK para Ruby desabilita o tráfego de rede e o cliente retorna dados simulados (ou falsos). Se você não fornecer dados simulados, o cliente retornará:

- Listas como matrizes vazias
- Mapas como hashes vazios
- Valores numéricos como zeros
- Datas como now

O exemplo a seguir retorna nomes simulados para a lista de buckets do 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
```

A execução desse código exibe o seguinte.

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

#### Note

Depois de você fornecer dados simulados, os valores padrão não se aplicam mais aos atributos de instância restantes. Isso significa que, no exemplo anterior, o atributo de instância restante, `creation_date`, não é `now`, mas sim `nil`.

O AWS SDK para Ruby valida seus dados simulados. Se você transmitir dados do tipo errado, ele lançará uma exceção `ArgumentError`. Por exemplo, se, em vez da atribuição anterior a `bucket_data`, você tiver usado o seguinte:

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

O AWS SDK para Ruby gera duas exceções de `ArgumentError`.

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

## Simular de erros de clientes

Você também pode simular de erros gerados pelo AWS SDK para Ruby para métodos específicos. O exemplo a seguir exibe `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
```

## Paginação

Algumas chamadas da AWS fornecem respostas paginadas para limitar a quantidade de dados retornados com cada resposta. Uma página de dados representa até 1000 itens.

### As respostas paginadas são enumeráveis

A maneira mais simples de lidar com dados de respostas paginadas é usar o enumerador interno no objeto de resposta, conforme mostrado no exemplo a seguir.

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

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

Isso produz um objeto de resposta por chamada à API feita e enumera objetos no bucket designado. O SDK recupera páginas adicionais de dados para completar a solicitação.

## Manipulação manual de respostas paginadas

Para lidar com a paginação por conta própria, use o método `next_page?` da resposta para verificar se há mais páginas para recuperar ou use o método `last_page?` para verificar se não há mais páginas para recuperar.

Se houver mais páginas, use o método `next_page` (observe que não há `?`) para recuperar a próxima página de resultados, conforme mostrado no exemplo a seguir.

```
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 você chamar o método `next_page` e não houver mais páginas para recuperar, o SDK lançará uma exceção [Aws::PageableResponse::LastPageError](#).

## Classes de dados paginadas

Os dados paginados no AWS SDK para Ruby são tratados pela classe [Aws::PageableResponse](#), que é incluída com [Seahorse::Client::Response](#) para fornecer acesso a dados paginados.

## Waiters

Agentes de espera são métodos utilitários que sondam um determinado estado para verificar se ele ocorreu em um cliente. Esses agentes podem falhar após uma série de tentativas em um intervalo de sondagem definido para o cliente do serviço. Para ver um exemplo de como um waiter é usado, consulte o método [create\\_table](#) do Amazon DynamoDB Encryption Client no repositório de exemplos de código da AWS.



## Invocar um waiter

Para invocar um waiter, chame o `wait_until` em um cliente de serviço. No exemplo a seguir, um waiter aguarda até que a instância `i-12345678` esteja em execução antes de continuar.

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

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

O primeiro parâmetro é o nome do waiter, que é específico para o cliente de serviço e indica qual operação está sendo esperada. O segundo parâmetro é um hash de parâmetros que são transmitidos ao método de cliente chamado pelo waiter, que varia de acordo com o nome desse agente.

Para obter uma lista das operações que podem ser esperadas e os métodos de cliente chamados para cada operação, consulte a documentação dos campos `waiter_names` e `wait_until` do cliente que você está usando.

## Falhas de espera

Os waiters podem falhar com uma das seguintes exceções.

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

Um estado de falha foi encontrado durante a espera.

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

O nome especificado do waiter não está definido para o cliente que está sendo usado.

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

O número de tentativas excedeu o valor `max_attempts` do waiter.

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

Ocorreu um erro inesperado durante a espera.

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

Um dos estados de espera foi excedido ou outra falha ocorreu durante a espera.

Todos esses erros, exceto `NoSuchWaiterError`, são baseados em `WaiterFailed`. Para capturar erros em um waiter, use o `WaiterFailed`, conforme mostrado no exemplo a seguir.

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

## Configurar um waiter

Cada servidor tem um intervalo de sondagem padrão e um número máximo de tentativas que ele fará antes de retornar o controle ao seu programa. Para definir esses valores, use os parâmetros `max_attempts` e `delay`: na sua chamada `wait_until`. O exemplo a seguir aguarda até 25 segundos, pesquisando a cada cinco segundos.

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

Para desabilitar falhas de espera, defina o valor de qualquer um desses parâmetros como `nil`.

## Estender um waiter

Para modificar o comportamento dos waiters, você pode registrar retornos de chamada que são disparados antes de cada tentativa de sondagem e antes da espera.

O exemplo a seguir implementa um recuo exponencial em um waiter, duplicando a quantidade de tempo de espera em todas as tentativas.

```
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
```

```
end
end
```

O exemplo a seguir desabilita o número máximo de tentativas e aguarda uma hora (3.600 segundos) antes de falhar.

```
started_at = Time.now
client.wait_until(...) do |w|
  # Disable max attempts
  w.max_attempts = nil

  # Poll for one hour, instead of a number of attempts
  w.before_wait do |attempts, response|
    throw :failure if Time.now - started_at > 3600
  end
end
```

## Especificar comportamento de nova tentativa do cliente

Por padrão, o AWS SDK para Ruby executa até três novas tentativas, com 15 segundos entre elas, com um total de até quatro tentativas. Portanto, uma operação pode demorar até 60 segundos para expirar.

O exemplo a seguir cria um cliente do Amazon S3 na região us-west-2 e especifica um período de espera de cinco segundos entre duas tentativas em cada operação de cliente. Portanto, as operações de cliente do Amazon S3 podem demorar até 15 segundos para expirar.

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

Este exemplo mostra como alterar os parâmetros de novas tentativas diretamente no código. No entanto, você também pode usar variáveis de ambiente ou o arquivo `AWS config` compartilhado para defini-las para seu aplicativo. Para obter mais informações sobre essas configurações, consulte [Comportamento de novas tentativas](#) no Guia de referência de ferramentas e SDKs da AWS. Qualquer configuração explícita definida no código ou no próprio cliente de serviço tem precedência sobre aquelas definidas nas variáveis de ambiente ou no arquivo `config` compartilhado.

# Migrar da versão 1 ou 2 para a versão 3 do AWS SDK para Ruby

O objetivo deste tópico é ajudar a migrar da versão 1 ou 2 do AWS SDK para Ruby para a versão 3.

## Uso lado a lado

Não é necessário substituir a versão 1 ou 2 do AWS SDK para Ruby pela versão 3. Você pode usá-las em conjunto no mesmo aplicativo. Veja esta [postagem de blog](#) para obter mais informações.

Um exemplo rápido.

```
require 'aws-sdk-v1' # version 1
require 'aws-sdk'   # version 2
require 'aws-sdk-s3' # version 3

s3 = AWS::S3::Client.new # version 1
s3 = Aws::S3::Client.new # version 2 or 3
```

Não é necessário reescrever o código existente da versão 1 ou 2 em funcionamento para começar a usar o SDK da versão 3. Uma estratégia de migração válida é simplesmente escrever um novo código com base no SDK da versão 3.

## Diferenças gerais

A versão 3 difere da versão 2 de uma maneira importante.

- Cada serviço está disponível como um gem separado.

A versão 2 difere da versão 1 de várias maneiras importantes.

- Namespace raiz diferente – `Aws` versus `AWS`. Isso permite o uso lado a lado.
- `Aws.config` — agora é um hash Ruby vanilla, em vez de um método.
- Opções de construtor restritas - Ao construir um objeto de cliente ou recurso no SDK da versão 1, opções de construtor desconhecidas são ignoradas. Na versão 2, as opções de construtor desconhecidas acionam um `ArgumentError`. Por exemplo:

```
# version 1
AWS::S3::Client.new(http_reed_timeout: 10)
# oops, typo'd option is ignored
```

```
# version 2
Aws::S3::Client.new(http_read_timeout: 10)
# => raises ArgumentError
```

## Diferenças de clientes

Não há diferenças entre as classes de cliente na versão 2 e na versão 3.

Entre a versão 1 e a versão 2, as classes de clientes têm o menor número possível de diferenças externas. Muitos clientes de serviços terão interfaces compatíveis após a construção do cliente.

Algumas diferenças importantes:

- `Aws::S3::Client` - A classe de cliente do Amazon S3 na versão 1 foi codificada manualmente. A versão 2 é gerada a partir de um modelo de serviço. Os nomes e entradas do método são muito diferentes na versão 2.
- `Aws::EC2::Client` - A versão 2 usa nomes plurais para listas de saída, enquanto a versão 1 usa o sufixo `_set`. Por exemplo:

```
# 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` – a versão 2 usa respostas estruturadas, enquanto a versão 1 usa hashes Ruby vanilla.
- Renomeações de classes de serviços – a versão 2 usa um nome diferente para vários serviços:
  - `AWS::SimpleWorkflow` se tornou `Aws::SWF`
  - `AWS::ELB` se tornou `Aws::ElasticLoadBalancing`
  - `AWS::SimpleEmailService` se tornou `Aws::SES`
- Opções de configuração de cliente: algumas das opções de configuração da versão 1 foram renomeadas na versão 2. Outras são removidas ou substituídas. Veja a seguir as principais mudanças:

- `:use_ssl` foi removido. A versão 2 usa SSL em todos os locais. Para desabilitar o SSL, você deve configurar um `:endpoint` que use `http://`.
- `:ssl_ca_file` é agora `:ssl_ca_bundle`
- `:ssl_ca_path` é agora `:ssl_ca_directory`
- Adição do `:ssl_ca_store`.
- O `:endpoint` agora deve ser um URI HTTP ou HTTPS totalmente qualificado, em vez de um nome de host.
- As opções de `:*_port` foram removidas para cada serviço, agora são substituídas por `:endpoint`.
- `:user_agent_prefix` é agora `:user_agent_suffix`

## Diferenças de recursos

Não há diferenças entre as interfaces de recursos na versão 2 e na versão 3.

Existem diferenças significativas entre as interfaces de recursos na versão 1 e na versão 2. A versão 1 era codificada manualmente na íntegra, enquanto as interfaces de recursos da versão 2 são geradas a partir de um modelo. As interfaces de recursos da versão 2 são significativamente mais consistentes. Algumas das diferenças sistêmicas incluem:

- Classe de recurso separada: na versão 2, o nome do serviço é um módulo, não uma classe. Neste módulo, ela é a interface de recurso:

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

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

- Referência a recursos – o SDK da versão 2 separa coleções e getters de recursos individuais em dois métodos diferentes:

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

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

- Operações em lotes: na versão 1, todas as operações em lotes eram utilitários codificados manualmente. Na versão 2, muitas operações em lotes são operações em lotes geradas automaticamente por meio da API. As interfaces de lote de versão 2 são muito diferentes daquelas da versão 1.

# SDK para exemplos de código Ruby

Os exemplos de código neste tópico mostram como usar o AWS SDK for Ruby with AWS.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Os cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

## Serviços

- [Exemplos de Aurora usando SDK para Ruby](#)
- [Exemplos de Auto Scaling usando SDK para Ruby](#)
- [CloudTrail exemplos de uso SDK para Ruby](#)
- [CloudWatch exemplos de uso SDK para Ruby](#)
- [Exemplos de provedores de identidade Amazon Cognito usando SDK para Ruby](#)
- [Exemplos do Amazon Comprehend usando para Ruby SDK](#)
- [Exemplos do Amazon DocumentDB usando SDK para Ruby](#)
- [Exemplos do DynamoDB usando para Ruby SDK](#)
- [EC2Exemplos da Amazon usando SDK para Ruby](#)
- [Exemplos de uso do Elastic SDK Beanstalk para Ruby](#)
- [EventBridge exemplos de uso SDK para Ruby](#)
- [AWS Glue exemplos de uso SDK para Ruby](#)
- [IAMexemplos de uso SDK para Ruby](#)
- [Exemplos de Kinesis usando SDK para Ruby](#)
- [AWS KMS exemplos de uso SDK para Ruby](#)
- [Exemplos de Lambda usando SDK para Ruby](#)
- [MSKExemplos da Amazon usando SDK para Ruby](#)
- [Exemplos de uso SDK do Amazon Polly para Ruby](#)
- [RDSExemplos da Amazon usando SDK para Ruby](#)



- [Exemplos do Amazon S3 usando SDK para Ruby](#)
- [SESExemplos da Amazon usando SDK para Ruby](#)
- [Exemplos da Amazon SES API v2 usando SDK para Ruby](#)
- [SNSExemplos da Amazon usando SDK para Ruby](#)
- [SQSExemplos da Amazon usando SDK para Ruby](#)
- [AWS STS exemplos de uso SDK para Ruby](#)
- [Exemplos do Amazon Textract usando SDK para Ruby](#)
- [Exemplos do Amazon Translate usando SDK para Ruby](#)
- [WorkDocs Exemplos da Amazon usando SDK para Ruby](#)

## Exemplos de Aurora usando SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Ruby with Aurora.

Cada exemplo inclui um link para o código-fonte completo, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Conceitos básicos

Olá, Aurora

Os exemplos de código a seguir mostram como começar a usar o Aurora.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
end
```

- Para API obter detalhes, consulte [DescreveDBClusters](#) em AWS SDK for Ruby API Referência.

## Exemplos de Auto Scaling usando SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Ruby com Auto Scaling.

Cada exemplo inclui um link para o código-fonte completo, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Conceitos básicos

Olá, Auto Scaling

Os exemplos de código a seguir mostram como começar a usar o Auto Scaling.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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

  end

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

- Para API obter detalhes, consulte [DescribeAutoScalingGroups](#) em AWS SDK for Ruby API Referência.

# CloudTrail exemplos de uso SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Ruby with CloudTrail.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

## Ações

### CreateTrail

O código de exemplo a seguir mostra como usar CreateTrail.

SDK para Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
  })
end
```

```
puts "Successfully created trail: #{trail_name}."
rescue StandardError => e
  puts "Got error trying to create trail #{trail_name}:\n #{e}"
  puts e
  exit 1
end
```

- Para API obter detalhes, consulte [CreateTrail](#) em AWS SDK for Ruby APIReferência.

## DeleteTrail

O código de exemplo a seguir mostra como usar DeleteTrail.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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

- Para API obter detalhes, consulte [DeleteTrail](#) em AWS SDK for Ruby APIReferência.

## ListTrails

O código de exemplo a seguir mostra como usar ListTrails.

## SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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

def describe_trails_example(client)
  resp = client.describe_trails({})
  puts "Found #{resp.trail_list.count} trail(s)."
```

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

- Para API obter detalhes, consulte [ListTrails](#) em AWS SDK for Ruby API Referência.

## LookupEvents

O código de exemplo a seguir mostra como usar LookupEvents.

## SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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

# @param [Object] client
def lookup_events_example(client)
```

```
resp = client.lookup_events
puts "Found #{resp.events.count} events:"
resp.events.each do |e|
  puts "Event name:   #{e.event_name}"
  puts "Event ID:     #{e.event_id}"
  puts "Event time:   #{e.event_time}"
  puts "Resources:"

  e.resources.each do |r|
    puts "  Name:       #{r.resource_name}"
    puts "  Type:       #{r.resource_type}"
    puts ""
  end
end
end
```

- Para API obter detalhes, consulte [LookupEvents](#) em AWS SDK for Ruby API Referência.

## CloudWatch exemplos de uso SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Ruby with CloudWatch.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)


## Ações

### **DescribeAlarms**

O código de exemplo a seguir mostra como usar `DescribeAlarms`.



## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
require "aws-sdk-cloudwatch"


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

- Para API obter detalhes, consulte [DescribeAlarms](#) em AWS SDK for Ruby API Referência.

## DescribeAlarmsForMetric

O código de exemplo a seguir mostra como usar DescribeAlarmsForMetric.

## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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

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

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

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

```
        puts " " + a
      end
    end

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

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

# Example usage:
def run_me
  region = ""

  # Print usage information and then stop.
  if ARGV[0] == "--help" || ARGV[0] == "-h"
    puts "Usage:  ruby cw-ruby-example-show-alarms.rb REGION"
    puts "Example: ruby cw-ruby-example-show-alarms.rb us-east-1"
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    region = "us-east-1"
  # Otherwise, use the values as specified at the command prompt.
  else
    region = ARGV[0]
  end
end
```

```
cloudwatch_client = Aws::CloudWatch::Client.new(region: region)
puts "Available alarms:"
describe_metric_alarms(cloudwatch_client)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para API obter detalhes, consulte [DescribeAlarmsForMetric](#) em AWS SDK for Ruby API Referência.

## DisableAlarmActions

O código de exemplo a seguir mostra como usar `DisableAlarmActions`.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
# Disables an alarm in Amazon CloudWatch.
#
# Prerequisites.
#
# - The alarm to disable.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param alarm_name [String] The name of the alarm to disable.
# @return [Boolean] true if the alarm was disabled; otherwise, false.
# @example
#   exit 1 unless alarm_actions_disabled?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'ObjectsInBucket'
#   )
def alarm_actions_disabled?(cloudwatch_client, alarm_name)
```

```
cloudwatch_client.disable_alarm_actions(alarm_names: [alarm_name])
return true
rescue StandardError => e
  puts "Error disabling alarm actions: #{e.message}"
  return false
end

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

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

  if alarm_created_or_updated?(
    cloudwatch_client,
    alarm_name,
    alarm_description,
    metric_name,
    alarm_actions,
```

```

    namespace,
    statistic,
    dimensions,
    period,
    unit,
    evaluation_periods,
    threshold,
    comparison_operator
  )
  puts "Alarm '#{alarm_name}' created or updated."
else
  puts "Could not create or update alarm '#{alarm_name}'."
end

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

run_me if $PROGRAM_NAME == __FILE__

```

- Para API obter detalhes, consulte [DisableAlarmAction](#) em AWS SDK for Ruby API Referência.

## ListMetrics

O código de exemplo a seguir mostra como usar ListMetrics.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```

# Lists available metrics for a metric namespace in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.

```

```
# @param metric_namespace [String] The namespace of the metric.
# @example
#   list_metrics_for_namespace(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'SITE/TRAFFIC'
#   )
def list_metrics_for_namespace(cloudwatch_client, metric_namespace)
  response = cloudwatch_client.list_metrics(namespace: metric_namespace)

  if response.metrics.count.positive?
    response.metrics.each do |metric|
      puts " Metric name: #{metric.metric_name}"
      if metric.dimensions.count.positive?
        puts "   Dimensions:"
        metric.dimensions.each do |dimension|
          puts "     Name: #{dimension.name}, Value: #{dimension.value}"
        end
      else
        puts "No dimensions found."
      end
    end
  else
    puts "No metrics found for namespace '#{metric_namespace}'. " \
      "Note that it could take up to 15 minutes for recently-added metrics " \
      "to become available."
  end
end

# Example usage:
def run_me
  metric_namespace = "SITE/TRAFFIC"
  # Replace us-west-2 with the AWS Region you're using for Amazon CloudWatch.
  region = "us-east-1"

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

  # Add three datapoints.
  puts "Continuing..." unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    "UniqueVisitors",
    "SiteName",
    "example.com",
    5_885.0,
```

```
    "Count"
  )

  puts "Continuing..." unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    "UniqueVisits",
    "SiteName",
    "example.com",
    8_628.0,
    "Count"
  )

  puts "Continuing..." unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    "PageViews",
    "PageURL",
    "example.html",
    18_057.0,
    "Count"
  )

  puts "Metrics for namespace '#{metric_namespace}':"
  list_metrics_for_namespace(cloudwatch_client, metric_namespace)
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para API obter detalhes, consulte [ListMetrics](#) em AWS SDK for Ruby API Referência.

## PutMetricAlarm

O código de exemplo a seguir mostra como usar PutMetricAlarm.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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: 'doc-example-bucket'
#       },
#       {
#         name: 'StorageType',
#         value: 'AllStorageTypes'
#       }
#     ],
#     86_400,
```

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

- Para API obter detalhes, consulte [PutMetricAlarm](#) em AWS SDK for Ruby API Referência.

## PutMetricData

O código de exemplo a seguir mostra como usar PutMetricData.

### SDKpara Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
require "aws-sdk-cloudwatch"

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

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

- Para API obter detalhes, consulte [PutMetricData](#) em AWS SDK for Ruby API Referência.

## Exemplos de provedores de identidade Amazon Cognito usando SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Ruby Amazon Cognito Identity Provider.

Cada exemplo inclui um link para o código-fonte completo, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

## Conceitos básicos

### Olá, Amazon Cognito

Os exemplos de código a seguir mostram como começar a usar o Amazon Cognito.

### SDKpara Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
```

```
    end
  end
end

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

- Para API obter detalhes, consulte [ListUserPools](#) em AWS SDK for Ruby API Referência.

## Exemplos do Amazon Comprehend usando para Ruby SDK

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Ruby com o Amazon Comprehend.

Os cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Cenários](#)

## Cenários

Criar uma aplicação para analisar o feedback dos clientes

O exemplo de código a seguir mostra como criar uma aplicação que analisa os cartões de comentários dos clientes, os traduz do idioma original, determina seus sentimentos e gera um arquivo de áudio do texto traduzido.

SDK para Ruby

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback

dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter o código-fonte e as instruções de implantação, consulte o projeto em [GitHub](#).

Serviços utilizados neste exemplo

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

## Exemplos do Amazon DocumentDB usando SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Ruby com o Amazon DocumentDB.

Cada exemplo inclui um link para o código-fonte completo, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Exemplos sem servidor](#)

### Exemplos sem servidor

Invocar uma função do Lambda de um acionador do Amazon DocumentDB

O exemplo de código a seguir mostra como implementar uma função Lambda que recebe um evento acionado pelo recebimento de registros de um stream de alterações do DocumentDB. A função recupera a carga útil do DocumentDB e registra em log o conteúdo do registro.

## SDK para Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Consumir um evento do Amazon DocumentDB com o Lambda usando 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
```

## Exemplos do DynamoDB usando para Ruby SDK

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Ruby com o DynamoDB.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.



Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Os cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

## Conceitos básicos

### Olá, DynamoDB

O exemplo de código a seguir mostra como começar a usar o DynamoDB.

### SDKpara Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
require 'aws-sdk-dynamodb'
require 'logger'

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

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

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

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

if table_names.empty?
  @logger.info("You don't have any DynamoDB tables in your account.")
else
  @logger.info("\nFound #{table_names.length} tables.")
end

end

end

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

- Para API obter detalhes, consulte [ListTables](#) em AWS SDK for Ruby API Referência.

## Tópicos

- [Conceitos básicos](#)
- [Ações](#)
- [Cenários](#)
- [Exemplos sem servidor](#)

## Conceitos básicos

Aprender os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar uma tabela que possa conter dados de filmes.
- Colocar, obter e atualizar um único filme na tabela.

- Grave dados do filme na tabela a partir de um JSON arquivo de amostra.
- Consultar filmes que foram lançados em determinado ano.
- Verificar filmes que foram lançados em um intervalo de anos.
- Excluir um filme da tabela e, depois, excluir a tabela.

## SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Crie uma classe que encapsule uma tabela do DynamoDB.

```
# Creates an Amazon DynamoDB table that can be used to store movie data.
# The table uses the release year of the movie as the partition key and the
# title as the sort key.
#
# @param table_name [String] The name of the table to create.
# @return [Aws::DynamoDB::Table] The newly created table.
def create_table(table_name)
  @table = @dynamo_resource.create_table(
    table_name: table_name,
    key_schema: [
      {attribute_name: "year", key_type: "HASH"}, # Partition key
      {attribute_name: "title", key_type: "RANGE"} # Sort key
    ],
    attribute_definitions: [
      {attribute_name: "year", attribute_type: "N"},
      {attribute_name: "title", attribute_type: "S"}
    ],
    provisioned_throughput: {read_capacity_units: 10, write_capacity_units: 10})
  @dynamo_resource.client.wait_until(:table_exists, table_name: table_name)
  @table
rescue Aws::DynamoDB::Errors::ServiceError => e
  @logger.error("Failed create table #{table_name}:\n#{e.code}: #{e.message}")
  raise
end
```

Crie uma função auxiliar para baixar e extrair o JSON arquivo de amostra.

```
# 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
```

Execute um cenário interativo para criar a tabela e executar ações nela.

```
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)
end
```

```
    print "Done!\n".green
  end

  new_step(2, "Add a new record to the DynamoDB table.")
  my_movie = {}
  my_movie[:title] = CLI::UI::Prompt.ask("Enter the title of a movie to add to the
table. E.g. The Matrix")
  my_movie[:year] = CLI::UI::Prompt.ask("What year was it released? E.g. 1989").to_i
  my_movie[:rating] = CLI::UI::Prompt.ask("On a scale of 1 - 10, how do you rate it?
E.g. 7").to_i
  my_movie[:plot] = CLI::UI::Prompt.ask("Enter a brief summary of the plot. E.g. A
man awakens to a new reality.")
  dynamodb_wrapper.add_item(my_movie)
  puts "\nNew record added:"
  puts JSON.pretty_generate(my_movie).green
  print "Done!\n".green

  new_step(3, "Update a record in the DynamoDB table.")
  my_movie[:rating] = CLI::UI::Prompt.ask("Let's update the movie you added with a
new rating, e.g. 3:").to_i
  response = dynamodb_wrapper.update_item(my_movie)
  puts "Updated '#{my_movie[:title]}' with new attributes:"
  puts JSON.pretty_generate(response).green
  print "Done!\n".green

  new_step(4, "Get a record from the DynamoDB table.")
  puts "Searching for #{my_movie[:title]} (#{my_movie[:year]})..."
  response = dynamodb_wrapper.get_item(my_movie[:title], my_movie[:year])
  puts JSON.pretty_generate(response).green
  print "Done!\n".green

  new_step(5, "Write a batch of items into the DynamoDB table.")
  download_file = "moviedata.json"
  puts "Downloading movie database to #{download_file}..."
  movie_data = scaffold.fetch_movie_data(download_file)
  puts "Writing movie data from #{download_file} into your table..."
  scaffold.write_batch(movie_data)
  puts "Records added: #{movie_data.length}."
  print "Done!\n".green

  new_step(5, "Query for a batch of items by key.")
  loop do
    release_year = CLI::UI::Prompt.ask("Enter a year between 1972 and 2018, e.g.
1999:").to_i
```

```
results = dynamodb_wrapper.query_items(release_year)
if results.any?
  puts("There were #{results.length} movies released in #{release_year}")
  results.each do |movie|
    print "\t #{movie["title"]}".green
  end
  break
else
  continue = CLI::UI::Prompt.ask("Found no movies released in #{release_year}!
Try another year? (y/n)")
  break if !continue.eql?("y")
end
end
print "\nDone!\n".green

new_step(6, "Scan for a batch of items using a filter expression.")
years = {}
years[:start] = CLI::UI::Prompt.ask("Enter a starting year between 1972 and
2018:")
years[:end] = CLI::UI::Prompt.ask("Enter an ending year between 1972 and 2018:")
releases = dynamodb_wrapper.scan_items(years)
if !releases.empty?
  puts("Found #{releases.length} movies.")
  count = Question.ask(
    "How many do you want to see? ", method(:is_int), in_range(1,
releases.length))
  puts("Here are your #{count} movies:")
  releases.take(count).each do |release|
    puts("\t#{release["title"]}")
  end
else
  puts("I don't know about any movies released between #{years[:start]} "\
    "and #{years[:end]}".")
end
print "\nDone!\n".green

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

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


- Para API obter detalhes, consulte os tópicos a seguir em AWS SDK for Ruby API Referência.
  - [BatchWriteItem](#)
  - [CreateTable](#)
  - [DeleteItem](#)
  - [DeleteTable](#)
  - [DescribeTable](#)
  - [GetItem](#)
  - [PutItem](#)
  - [Query](#)
  - [Scan](#)
  - [UpdateItem](#)

## Ações

### BatchExecuteStatement

O código de exemplo a seguir mostra como usar BatchExecuteStatement.

## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Leia um lote de itens usando o PartiQL.

```
class DynamoDBPartiQLBatch

  attr_reader :dynamo_resource
  attr_reader :table

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

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

Exclua um lote de itens usando o PartiQL.

```
class DynamoDBPartiQLBatch

  attr_reader :dynamo_resource
  attr_reader :table
```



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

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

- Para API obter detalhes, consulte [BatchExecuteStatement](#) em AWS SDK for Ruby APIReferência.

## BatchWriteItem

O código de exemplo a seguir mostra como usar BatchWriteItem.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
class DynamoDBBasics
  attr_reader :dynamo_resource
  attr_reader :table
end
```

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


# Fills an Amazon DynamoDB table with the specified data. Items are sent in
# batches of 25 until all items are written.
#
# @param movies [Enumerable] The data to put in the table. Each item must contain
at least
#
#           the keys required by the schema that was specified
when the
#
#           table was created.
def write_batch(movies)
  index = 0
  slice_size = 25
  while index < movies.length
    movie_items = []
    movies[index, slice_size].each do |movie|
      movie_items.append({put_request: { item: movie }})
    end
    @dynamo_resource.client.batch_write_item({request_items: { @table.name =>
movie_items }})
    index += slice_size
  end
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts(
      "Couldn't load data into table #{@table.name}. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- Para API obter detalhes, consulte [BatchWriteItem](#) em AWS SDK for Ruby API Referência.

## CreateTable

O código de exemplo a seguir mostra como usar CreateTable.

## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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

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

  # Creates an Amazon DynamoDB table that can be used to store movie data.
  # The table uses the release year of the movie as the partition key and the
  # title as the sort key.
  #
  # @param table_name [String] The name of the table to create.
  # @return [Aws::DynamoDB::Table] The newly created table.
  def create_table(table_name)
    @table = @dynamo_resource.create_table(
      table_name: table_name,
      key_schema: [
        {attribute_name: "year", key_type: "HASH"}, # Partition key
        {attribute_name: "title", key_type: "RANGE"} # Sort key
      ],
      attribute_definitions: [
        {attribute_name: "year", attribute_type: "N"},
        {attribute_name: "title", attribute_type: "S"}
      ],
      provisioned_throughput: {read_capacity_units: 10, write_capacity_units: 10})
    @dynamo_resource.client.wait_until(:table_exists, table_name: table_name)
  end
end
```

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

- Para API obter detalhes, consulte [CreateTable](#) em AWS SDK for Ruby API Referência.

## DeleteItem

O código de exemplo a seguir mostra como usar DeleteItem.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
class DynamoDBBasics
  attr_reader :dynamo_resource
  attr_reader :table

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

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

- Para API obter detalhes, consulte [DeleteItem](#) em AWS SDK for Ruby API Referência.

## DeleteTable

O código de exemplo a seguir mostra como usar DeleteTable.

### SDK para Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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

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

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

- Para API obter detalhes, consulte [DeleteTable](#) em AWS SDK for Ruby API Referência.

## DescribeTable

O código de exemplo a seguir mostra como usar DescribeTable.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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

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

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

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

- Para API obter detalhes, consulte [DescribeTable](#) em AWS SDK for Ruby API Referência.

## ExecuteStatement

O código de exemplo a seguir mostra como usar ExecuteStatement.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Selecione um único item usando o PartiQL.

```
class DynamoDBPartiQLSingle

  attr_reader :dynamo_resource
  attr_reader :table

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

  # Gets a single record from a table using PartiQL.
  # Note: To perform more fine-grained selects,
  # use the Client.query instance method instead.
  #
  # @param title [String] The title of the movie to search.
  # @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
  def select_item_by_title(title)
    request = {
      statement: "SELECT * FROM \"#{@table.name}\" WHERE title=?",
      parameters: [title]
    }
  end
end
```

```

    }
    @dynamodb.client.execute_statement(request)
  end

```

### Atualize um único item usando o PartiQL.

```

class DynamoDBPartiQLSingle

  attr_reader :dynamo_resource
  attr_reader :table

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

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

```

### Adicione um único item usando o PartiQL.

```

class DynamoDBPartiQLSingle

  attr_reader :dynamo_resource
  attr_reader :table

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

```



```

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

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

```

Exclua um único item usando o PartiQL.

```

class DynamoDBPartiQLSingle

  attr_reader :dynamo_resource
  attr_reader :table

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

  # Deletes a single record from a table using PartiQL.
  #
  # @param title [String] The title of the movie to update.
  # @param year [Integer] The year the movie was released.
  # @return [Aws::DynamoDB::Types::ExecuteStatementOutput]
  def delete_item_by_title(title, year)
    request = {
      statement: "DELETE FROM \"#{@table.name}\" WHERE title=? and year=?",
      parameters: [title, year]
    }

```

```
}
  @dynamodb.client.execute_statement(request)
end
```

- Para API obter detalhes, consulte [ExecuteStatement](#) em AWS SDK for Ruby API Referência.

## GetItem

O código de exemplo a seguir mostra como usar GetItem.

### SDK para Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
class DynamoDBBasics
  attr_reader :dynamo_resource
  attr_reader :table

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

  # Gets movie data from the table for a specific movie.
  #
  # @param title [String] The title of the movie.
  # @param year [Integer] The release year of the movie.
  # @return [Hash] The data about the requested movie.
  def get_item(title, year)
    @table.get_item(key: {"year" => year, "title" => title})
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't get movie #{title} (#{year}) from table #{@table.name}:\n")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- Para API obter detalhes, consulte [GetItem](#) em AWS SDK for Ruby API Referência.

## ListTables

O código de exemplo a seguir mostra como usar `ListTables`.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Determine se uma tabela existe.

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

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

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

- Para API obter detalhes, consulte [ListTables](#) em AWS SDK for Ruby API Referência.

## PutItem

O código de exemplo a seguir mostra como usar PutItem.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
class DynamoDBBasics
  attr_reader :dynamo_resource
  attr_reader :table

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

  # Adds a movie to the table.
  #
  # @param movie [Hash] The title, year, plot, and rating of the movie.
  def add_item(movie)
    @table.put_item(
      item: {
        "year" => movie[:year],
```

```

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

```

- Para API obter detalhes, consulte [PutItem](#) em AWS SDK for Ruby API Referência.

## Query

O código de exemplo a seguir mostra como usar Query.

### SDKpara Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```

class DynamoDBBasics
  attr_reader :dynamo_resource
  attr_reader :table

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

  # Queries for movies that were released in the specified year.
  #
  # @param year [Integer] The year to query.
  # @return [Array] The list of movies that were released in the specified year.
  def query_items(year)
    response = @table.query(
      key_condition_expression: "#yr = :year",
      expression_attribute_names: {"#yr" => "year"},

```

```
expression_attribute_values: {":year" => year})
rescue Aws::DynamoDB::Errors::ServiceError => e
  puts("Couldn't query for movies released in #{year}. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  response.items
end
```

- Para API obter detalhes, consulte [Consulta](#) na AWS SDK for Ruby API referência.

## Scan

O código de exemplo a seguir mostra como usar Scan.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
class DynamoDBBasics
  attr_reader :dynamo_resource
  attr_reader :table

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

  # Scans for movies that were released in a range of years.
  # Uses a projection expression to return a subset of data for each movie.
  #
  # @param year_range [Hash] The range of years to retrieve.
  # @return [Array] The list of movies released in the specified years.
  def scan_items(year_range)
    movies = []
```

```

scan_hash = {
  filter_expression: "#yr between :start_yr and :end_yr",
  projection_expression: "#yr, title, info.rating",
  expression_attribute_names: {"#yr" => "year"},
  expression_attribute_values: {
    ":start_yr" => year_range[:start], ":end_yr" => year_range[:end]}
}
done = false
start_key = nil
until done
  scan_hash[:exclusive_start_key] = start_key unless start_key.nil?
  response = @table.scan(scan_hash)
  movies.concat(response.items) unless response.items.empty?
  start_key = response.last_evaluated_key
  done = start_key.nil?
end
rescue Aws::DynamoDB::Errors::ServiceError => e
  puts("Couldn't scan for movies. Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  movies
end

```

- Para API obter detalhes, consulte [Digitalizar](#) em AWS SDK for Ruby APIreferência.

## UpdateItem

O código de exemplo a seguir mostra como usar UpdateItem.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```

class DynamoDBBasics
  attr_reader :dynamo_resource

```

```
attr_reader :table

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

# Updates rating and plot data for a movie in the table.
#
# @param movie [Hash] The title, year, plot, rating of the movie.
def update_item(movie)

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

- Para API obter detalhes, consulte [UpdateItem](#) em AWS SDK for Ruby API Referência.

## Cenários

### Consultar uma tabela usando lotes de instruções PartiQL

O exemplo de código a seguir mostra como:

- Obtenha um lote de itens executando várias SELECT instruções.
- Adicione um lote de itens executando várias INSERT instruções.
- Atualize um lote de itens executando várias UPDATE instruções.
- Exclua um lote de itens executando várias DELETE instruções.



## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Execute um cenário que crie uma tabela e execute consultas do PartiQL em lotes.

```
table_name = "doc-example-table-movies-partiql-#{rand(10**4)}"
scaffold = Scaffold.new(table_name)
sdk = DynamoDBPartiQLBatch.new(table_name)

new_step(1, "Create a new DynamoDB table if none already exists.")
unless scaffold.exists?(table_name)
  puts("\nNo such table: #{table_name}. Creating it...")
  scaffold.create_table(table_name)
  print "Done!\n".green
end

new_step(2, "Populate DynamoDB table with movie data.")
download_file = "moviedata.json"
puts("Downloading movie database to #{download_file}...")
movie_data = scaffold.fetch_movie_data(download_file)
puts("Writing movie data from #{download_file} into your table...")
scaffold.write_batch(movie_data)
puts("Records added: #{movie_data.length}.")
print "Done!\n".green

new_step(3, "Select a batch of items from the movies table.")
puts "Let's select some popular movies for side-by-side comparison."
response = sdk.batch_execute_select([["Mean Girls", 2004], ["Goodfellas", 1977],
["The Prancing of the Lambs", 2005]])
puts("Items selected: #{response['responses'].length}\n")
print "\nDone!\n".green

new_step(4, "Delete a batch of items from the movies table.")
sdk.batch_execute_write([["Mean Girls", 2004], ["Goodfellas", 1977], ["The
Prancing of the Lambs", 2005]])
print "\nDone!\n".green

new_step(5, "Delete the table.")
```

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

- Para API obter detalhes, consulte [BatchExecuteStatement](#) em AWS SDK for Ruby API Referência.

## Consultar uma tabela usando o PartiQL

O exemplo de código a seguir mostra como:

- Obtenha um item executando uma SELECT declaração.
- Adicione um item executando uma INSERT declaração.
- Atualize um item executando uma UPDATE declaração.
- Exclua um item executando uma DELETE declaração.

## SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Execute um cenário que crie uma tabela e execute consultas do PartiQL.

```
table_name = "doc-example-table-movies-partiql-#{rand(10**8)}"
scaffold = Scaffold.new(table_name)
sdk = DynamoDBPartiQLSingle.new(table_name)

new_step(1, "Create a new DynamoDB table if none already exists.")
unless scaffold.exists?(table_name)
  puts("\nNo such table: #{table_name}. Creating it...")
  scaffold.create_table(table_name)
  print "Done!\n".green
end
```

```
new_step(2, "Populate DynamoDB table with movie data.")
download_file = "moviedata.json"
puts("Downloading movie database to #{download_file}...")
movie_data = scaffold.fetch_movie_data(download_file)
puts("Writing movie data from #{download_file} into your table...")
scaffold.write_batch(movie_data)
puts("Records added: #{movie_data.length}.")
print "Done!\n".green

new_step(3, "Select a single item from the movies table.")
response = sdk.select_item_by_title("Star Wars")
puts("Items selected for title 'Star Wars': #{response.items.length}\n")
print "#{response.items.first}".yellow
print "\n\nDone!\n".green

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

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

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

new_step(7, "Delete the table.")
if scaffold.exists?(table_name)
  scaffold.delete_table
end
end
```

- Para API obter detalhes, consulte [ExecuteStatement](#) em AWS SDK for Ruby API Referência.

## Exemplos sem servidor

### Invocar uma função do Lambda em um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma função Lambda que recebe um evento acionado pelo recebimento de registros de um stream do DynamoDB. A função recupera a carga útil do DynamoDB e registra em log o conteúdo do registro.

#### SDKpara Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Como consumir um evento do DynamoDB com o Lambda usando 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
```

### Relatar falhas de itens em lote para funções do Lambda com um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um stream do DynamoDB. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

## SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Como relatar falhas de itens em lote do DynamoDB com o Lambda usando 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 Exemplos da Amazon usando SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Ruby com a AmazonEC2.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Conceitos básicos

## Olá Amazon EC2

Os exemplos de código a seguir mostram como começar a usar a AmazonEC2.

### SDKpara Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
require 'aws-sdk-ec2'
require 'logger'

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

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

    instances = fetch_instances

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

  private

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

```
paginator = @client.describe_instances
instances = []

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

instances

end

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

end

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

- Para API obter detalhes, consulte [DescribeSecurityGroup](#) sem AWS SDK for Ruby API Referência.

## Tópicos

- [Ações](#)

## Ações

### AllocateAddress

O código de exemplo a seguir mostra como usar `AllocateAddress`.

SDKpara Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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


- Para API obter detalhes, consulte [AllocateAddress](#) em AWS SDK for Ruby API Referência.

### AssociateAddress

O código de exemplo a seguir mostra como usar `AssociateAddress`.



## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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,
  )
  return response.association_id
rescue StandardError => e
  puts "Error associating Elastic IP address with instance: #{e.message}"
  return "Error"
end
```

- Para API obter detalhes, consulte [AssociateAddress](#) em AWS SDK for Ruby API Referência.

## CreateKeyPair

O código de exemplo a seguir mostra como usar CreateKeyPair.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
# This code example does the following:
# 1. Creates a key pair in Amazon Elastic Compute Cloud (Amazon EC2).
# 2. Displays information about available key pairs.
# 3. Deletes the key pair.

require "aws-sdk-ec2"

# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param key_pair_name [String] The name for the key pair and private
#   key file.
# @return [Boolean] true if the key pair and private key file were
#   created; otherwise, false.
# @example
#   exit 1 unless key_pair_created?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-key-pair'
#   )
def key_pair_created?(ec2_client, key_pair_name)
  key_pair = ec2_client.create_key_pair(key_name: key_pair_name)
  puts "Created key pair '#{key_pair.key_name}' with fingerprint " \
    "'#{key_pair.key_fingerprint}' and ID '#{key_pair.key_pair_id}'."
  filename = File.join(Dir.home, key_pair_name + ".pem")
  File.open(filename, "w") { |file| file.write(key_pair.key_material) }
  puts "Private key file saved locally as '#{filename}'."
  return true
rescue Aws::EC2::Errors::InvalidKeyPairDuplicate
  puts "Error creating key pair: a key pair named '#{key_pair_name}' " \
```

```

    "already exists."
    return false
  rescue StandardError => e
    puts "Error creating key pair or saving private key file: #{e.message}"
    return false
  end

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

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

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

```

```
rescue StandardError => e
  puts "Error deleting key pair: #{e.message}"
  return false
end

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

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

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

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

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

  puts "-" * 10
  puts "Deleting key pair..."
  unless key_pair_deleted?(ec2_client, key_pair_name)
```

```
puts "Stopping program. You must delete the key pair yourself."
exit 1
end
puts "Key pair deleted."

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

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

run_me if $PROGRAM_NAME == __FILE__
```

- Para API obter detalhes, consulte [CreateKeyPair](#) em AWS SDK for Ruby API Referência.

## CreateRouteTable

O código de exemplo a seguir mostra como usar CreateRouteTable.

### SDKpara Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
require "aws-sdk-ec2"

# Prerequisites:
#
```

```
# - A VPC in Amazon VPC.
# - A subnet in that VPC.
# - A gateway attached to that subnet.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param vpc_id [String] The ID of the VPC for the route table.
# @param subnet_id [String] The ID of the subnet for the route table.
# @param gateway_id [String] The ID of the gateway for the route.
# @param destination_cidr_block [String] The destination CIDR block
#   for the route.
# @param tag_key [String] The key portion of the tag for the route table.
# @param tag_value [String] The value portion of the tag for the route table.
# @return [Boolean] true if the route table was created and associated;
#   otherwise, false.
# @example
#   exit 1 unless route_table_created_and_associated?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     'vpc-0b6f769731EXAMPLE',
#     'subnet-03d9303b57EXAMPLE',
#     'igw-06ca90c011EXAMPLE',
#     '0.0.0.0/0',
#     'my-key',
#     'my-value'
#   )
def route_table_created_and_associated?(
  ec2_resource,
  vpc_id,
  subnet_id,
  gateway_id,
  destination_cidr_block,
  tag_key,
  tag_value
)
  route_table = ec2_resource.create_route_table(vpc_id: vpc_id)
  puts "Created route table with ID '#{route_table.id}'."
  route_table.create_tags(
    tags: [
      {
        key: tag_key,
        value: tag_value
      }
    ]
  )
)
```

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

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

```
    tag_value = "my-value"
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    region = "us-west-2"
# Otherwise, use the values as specified at the command prompt.
else
  vpc_id = ARGV[0]
  subnet_id = ARGV[1]
  gateway_id = ARGV[2]
  destination_cidr_block = ARGV[3]
  tag_key = ARGV[4]
  tag_value = ARGV[5]
  region = ARGV[6]
end

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

if route_table_created_and_associated?(
  ec2_resource,
  vpc_id,
  subnet_id,
  gateway_id,
  destination_cidr_block,
  tag_key,
  tag_value
)
  puts "Route table created and associated."
else
  puts "Route table not created or not associated."
end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para API obter detalhes, consulte [CreateRouteTable](#) em AWS SDK for Ruby API Referência.

## CreateSecurityGroup

O código de exemplo a seguir mostra como usar CreateSecurityGroup.



## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
# This code example does the following:
# 1. Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.
# 2. Adds inbound rules to the security group.
# 3. Displays information about available security groups.
# 4. Deletes the security group.

require "aws-sdk-ec2"

# Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.
#
# Prerequisites:
#
# - A VPC in Amazon Virtual Private Cloud (Amazon VPC).
#
# @param ec2_client [Aws::EC2::Client] An initialized
#   Amazon EC2 client.
# @param group_name [String] A name for the security group.
# @param description [String] A description for the security group.
# @param vpc_id [String] The ID of the VPC for the security group.
# @return [String] The ID of security group that was created.
# @example
#   puts create_security_group(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'my-security-group',
#     'This is my security group.',
#     'vpc-6713dfEX'
#   )
def create_security_group(
  ec2_client,
  group_name,
  description,
  vpc_id
)
```

```
security_group = ec2_client.create_security_group(
  group_name: group_name,
  description: description,
  vpc_id: vpc_id
)
puts "Created security group '#{group_name}' with ID " \
     "'#{security_group.group_id}' in VPC with ID '#{vpc_id}'."
return security_group.group_id
rescue StandardError => e
  puts "Error creating security group: #{e.message}"
  return "Error"
end

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

```

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

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

  unless perm.from_port.nil?

```

```

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

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

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

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

  print "\n"
end

# Displays information about available security groups in
# Amazon Elastic Compute Cloud (Amazon EC2).
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @example
#   describe_security_groups(Aws::EC2::Client.new(region: 'us-west-2'))
def describe_security_groups(ec2_client)
  response = ec2_client.describe_security_groups

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

```

```

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

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

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

# Deletes an Amazon Elastic Compute Cloud (Amazon EC2)
# security group.
#
# Prerequisites:
#
# - The security group.
#
# @param ec2_client [Aws::EC2::Client] An initialized
#   Amazon EC2 client.
# @param security_group_id [String] The ID of the security group to delete.
# @return [Boolean] true if the security group was deleted; otherwise, false.
# @example
#   exit 1 unless security_group_deleted?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'sg-030a858e078f1b9EX'
#   )

```

```
def security_group_deleted?(ec2_client, security_group_id)
  ec2_client.delete_security_group(group_id: security_group_id)
  puts "Deleted security group '#{security_group_id}'."
  return true
rescue StandardError => e
  puts "Error deleting security group: #{e.message}"
  return false
end

# Example usage:
def run_me
  group_name = ""
  description = ""
  vpc_id = ""
  ip_protocol_http = ""
  from_port_http = ""
  to_port_http = ""
  cidr_ip_range_http = ""
  ip_protocol_ssh = ""
  from_port_ssh = ""
  to_port_ssh = ""
  cidr_ip_range_ssh = ""
  region = ""
  # Print usage information and then stop.
  if ARGV[0] == "--help" || ARGV[0] == "-h"
    puts "Usage:  ruby ec2-ruby-example-security-group.rb " \
      "GROUP_NAME DESCRIPTION VPC_ID IP_PROTOCOL_1 FROM_PORT_1 TO_PORT_1 " \
      "CIDR_IP_RANGE_1 IP_PROTOCOL_2 FROM_PORT_2 TO_PORT_2 " \
      "CIDR_IP_RANGE_2 REGION"
    puts "Example: ruby ec2-ruby-example-security-group.rb " \
      "my-security-group 'This is my security group.' vpc-6713dfEX " \
      "tcp 80 80 '0.0.0.0/0' tcp 22 22 '0.0.0.0/0' us-west-2"
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    group_name = "my-security-group"
    description = "This is my security group."
    vpc_id = "vpc-6713dfEX"
    ip_protocol_http = "tcp"
    from_port_http = "80"
    to_port_http = "80"
    cidr_ip_range_http = "0.0.0.0/0"
    ip_protocol_ssh = "tcp"
    from_port_ssh = "22"
```

```
to_port_ssh = "22"
cidr_ip_range_ssh = "0.0.0.0/0"
# Replace us-west-2 with the AWS Region you're using for Amazon EC2.
region = "us-west-2"
# Otherwise, use the values as specified at the command prompt.
else
  group_name = ARGV[0]
  description = ARGV[1]
  vpc_id = ARGV[2]
  ip_protocol_http = ARGV[3]
  from_port_http = ARGV[4]
  to_port_http = ARGV[5]
  cidr_ip_range_http = ARGV[6]
  ip_protocol_ssh = ARGV[7]
  from_port_ssh = ARGV[8]
  to_port_ssh = ARGV[9]
  cidr_ip_range_ssh = ARGV[10]
  region = ARGV[11]
end

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

puts "Attempting to create security group..."
security_group_id = create_security_group(
  ec2_client,
  group_name,
  description,
  vpc_id
)
if security_group_id == "Error"
  puts "Could not create security group. Skipping this step."
else
  security_group_exists = true
end

if security_group_exists
  puts "Attempting to add inbound rules to security group..."
  unless security_group_ingress_authorized?(
    ec2_client,
    security_group_id,
    ip_protocol_http,
    from_port_http,
```

```
    to_port_http,
    cidr_ip_range_http
  )
  puts "Could not add inbound HTTP rule to security group. " \
    "Skipping this step."
end

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```


- Para API obter detalhes, consulte [CreateSecurityGroup](#) em AWS SDK for Ruby API Referência.

## CreateSubnet

O código de exemplo a seguir mostra como usar CreateSubnet.



## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
require "aws-sdk-ec2"

# Creates a subnet within a virtual private cloud (VPC) in
# Amazon Virtual Private Cloud (Amazon VPC) and then tags
# the subnet.
#
# Prerequisites:
#
# - A VPC in Amazon VPC.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param vpc_id [String] The ID of the VPC for the subnet.
# @param cidr_block [String] The IPv4 CIDR block for the subnet.
# @param availability_zone [String] The ID of the Availability Zone
#   for the subnet.
# @param tag_key [String] The key portion of the tag for the subnet.
# @param tag_vlue [String] The value portion of the tag for the subnet.
# @return [Boolean] true if the subnet was created and tagged;
#   otherwise, false.
# @example
#   exit 1 unless subnet_created_and_tagged?(
#     Aws::EC2::Resource.new(region: 'us-west-2'),
#     'vpc-6713dfEX',
#     '10.0.0.0/24',
#     'us-west-2a',
#     'my-key',
#     'my-value'
#   )
def subnet_created_and_tagged?(
  ec2_resource,
  vpc_id,
  cidr_block,
```

```
    availability_zone,
    tag_key,
    tag_value
  )
  subnet = ec2_resource.create_subnet(
    vpc_id: vpc_id,
    cidr_block: cidr_block,
    availability_zone: availability_zone
  )
  subnet.create_tags(
    tags: [
      {
        key: tag_key,
        value: tag_value
      }
    ]
  )
  puts "Subnet created with ID '#{subnet.id}' in VPC with ID '#{vpc_id}' " \
    "and CIDR block '#{cidr_block}' in availability zone " \
    "'#{availability_zone}' and tagged with key '#{tag_key}' and " \
    "value '#{tag_value}'."
  return true
rescue StandardError => e
  puts "Error creating or tagging subnet: #{e.message}"
  return false
end

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

```
elsif ARGV.count.zero?
  vpc_id = "vpc-6713dfEX"
  cidr_block = "10.0.0.0/24"
  availability_zone = "us-west-2a"
  tag_key = "my-key"
  tag_value = "my-value"
  # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
  region = "us-west-2"
# Otherwise, use the values as specified at the command prompt.
else
  vpc_id = ARGV[0]
  cidr_block = ARGV[1]
  availability_zone = ARGV[2]
  tag_key = ARGV[3]
  tag_value = ARGV[4]
  region = ARGV[5]
end

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

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


run_me if $PROGRAM_NAME == __FILE__
```

- Para API obter detalhes, consulte [CreateSubnet](#) em AWS SDK for Ruby API Referência.

## CreateVpc

O código de exemplo a seguir mostra como usar CreateVpc.

## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
require "aws-sdk-ec2"

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

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

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

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

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

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

  if vpc_created_and_tagged?(
    ec2_resource,
    cidr_block,
    tag_key,
```

```
    tag_value
  )
  puts "VPC created and tagged."
else
  puts "VPC not created or not tagged."
end
end
end

run_me if $PROGRAM_NAME == __FILE__
```

- Para API obter detalhes, consulte [CreateVpc](#) em AWS SDK for Ruby APIReferência.

## DescribeInstances

O código de exemplo a seguir mostra como usar DescribeInstances.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
require "aws-sdk-ec2"

# @param ec2_resource [Aws::EC2::Resource] An initialized EC2 resource object.
# @example
# list_instance_ids_states(Aws::EC2::Resource.new(region: 'us-west-2'))
def list_instance_ids_states(ec2_resource)
  response = ec2_resource.instances
  if response.count.zero?
    puts "No instances found."
  else
    puts "Instances -- ID, state:"
    response.each do |instance|
      puts "#{instance.id}, #{instance.state.name}"
    end
  end
end

rescue StandardError => e
```

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

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

run_me if $PROGRAM_NAME == __FILE__
```

- Para API obter detalhes, consulte [DescribeInstances](#) em AWS SDK for Ruby API Referência.

## DescribeRegions

O código de exemplo a seguir mostra como usar DescribeRegions.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
require "aws-sdk-ec2"

# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   list_regions_endpoints(Aws::EC2::Client.new(region: 'us-west-2'))
def list_regions_endpoints(ec2_client)
  result = ec2_client.describe_regions
  # Enable pretty printing.
  max_region_string_length = 16
  max_endpoint_string_length = 33
  # Print header.
  print "Region"
  print " " * (max_region_string_length - "Region".length)
  print "  Endpoint\n"
  print "-" * max_region_string_length
  print " "
  print "-" * max_endpoint_string_length
  print "\n"
  # Print Regions and their endpoints.
  result.regions.each do |region|
    print region.region_name
    print " " * (max_region_string_length - region.region_name.length)
    print " "
    print region.endpoint
    print "\n"
  end
end

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



```

print " Zone"
print " " * (max_zone_string_length - "Zone".length)
print " State\n"
print "-" * max_region_string_length
print " "
print "-" * max_zone_string_length
print " "
print "-" * max_state_string_length
print "\n"
# Print Regions, Availability Zones, and their states.
result.availability_zones.each do |zone|
  print zone.region_name
  print " " * (max_region_string_length - zone.region_name.length)
  print " "
  print zone.zone_name
  print " " * (max_zone_string_length - zone.zone_name.length)
  print " "
  print zone.state
  # Print any messages for this Availability Zone.
  if zone.messages.count.positive?
    print "\n"
    puts " Messages for this zone:"
    zone.messages.each do |message|
      print "   #{message.message}\n"
    end
  end
  print "\n"
end
end

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

```

```

else
  region = ARGV[0]
end

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

puts "AWS Regions for Amazon EC2 that are available to you:"
list_regions_endpoints(ec2_client)
puts "\n\nAmazon EC2 Availability Zones that are available to you for AWS Region
'#{region}':"
list_availability_zones(ec2_client)
end

run_me if $PROGRAM_NAME == __FILE__

```

- Para API obter detalhes, consulte [DescribeRegions](#) em AWS SDK for Ruby API Referência.

## ReleaseAddress

O código de exemplo a seguir mostra como usar ReleaseAddress.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```

# Releases an Elastic IP address from an
# Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - An Amazon EC2 instance with an associated Elastic IP address.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param allocation_id [String] The ID of the allocation corresponding to
# the Elastic IP address.
# @return [Boolean] true if the Elastic IP address was released;
# otherwise, false.

```

```
# @example
#   exit 1 unless elastic_ip_address_released?(
#     Aws::EC2::Client.new(region: 'us-west-2'),
#     'eipalloc-04452e528a66279EX'
#   )
def elastic_ip_address_released?(ec2_client, allocation_id)
  ec2_client.release_address(allocation_id: allocation_id)
  return true
rescue StandardError => e
  puts("Error releasing Elastic IP address: #{e.message}")
  return false
end
```

- Para API obter detalhes, consulte [ReleaseAddress](#) em AWS SDK for Ruby API Referência.

## StartInstances

O código de exemplo a seguir mostra como usar StartInstances.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
end

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

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

```
# Replace us-west-2 with the AWS Region you're using for Amazon EC2.
elsif ARGV.count.zero?
  instance_id = "i-123abc"
  region = "us-west-2"
# Otherwise, use the values as specified at the command prompt.
else
  instance_id = ARGV[0]
  region = ARGV[1]
end

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

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

run_me if $PROGRAM_NAME == __FILE__
```

- Para API obter detalhes, consulte [StartInstances](#) em AWS SDK for Ruby API Referência.

## StopInstances

O código de exemplo a seguir mostra como usar StopInstances.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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."
return true
rescue StandardError => e
  puts "Error stopping instance: #{e.message}"
  return false
end

# Example usage:
def run_me
  instance_id = ""
  region = ""
  # Print usage information and then stop.
  if ARGV[0] == "--help" || ARGV[0] == "-h"
    puts "Usage:  ruby ec2-ruby-example-stop-instance-i-123abc.rb " \
```

```
"INSTANCE_ID REGION "  
# Replace us-west-2 with the AWS Region you're using for Amazon EC2.  
puts "Example: ruby ec2-ruby-example-start-instance-i-123abc.rb " \  
  "i-123abc us-west-2"  
exit 1  
# If no values are specified at the command prompt, use these default values.  
# Replace us-west-2 with the AWS Region you're using for Amazon EC2.  
elsif ARGV.count.zero?  
  instance_id = "i-123abc"  
  region = "us-west-2"  
# Otherwise, use the values as specified at the command prompt.  
else  
  instance_id = ARGV[0]  
  region = ARGV[1]  
end  
  
ec2_client = Aws::EC2::Client.new(region: region)  
  
puts "Attempting to stop instance '#{instance_id}' " \  
  "(this might take a few minutes)..."  
unless instance_stopped?(ec2_client, instance_id)  
  puts "Could not stop instance."  
end  
end  
  
run_me if $PROGRAM_NAME == __FILE__
```

- Para API obter detalhes, consulte [StopInstances](#) em AWS SDK for Ruby API Referência.

## TerminateInstances

O código de exemplo a seguir mostra como usar `TerminateInstances`.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
require "aws-sdk-ec2"

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

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

    puts "The instance is already terminated."
    return true
  end

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

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



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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

- Para API obter detalhes, consulte [TerminateInstances](#) em AWS SDK for Ruby API Referência.

## Exemplos de uso do Elastic SDK Beanstalk para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Ruby com o Elastic Beanstalk.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)

## Ações

### DescribeApplications

O código de exemplo a seguir mostra como usar DescribeApplications.

SDKpara Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
# Class to manage Elastic Beanstalk applications
class ElasticBeanstalkManager
  def initialize(eb_client, logger: Logger.new($stdout))
    @eb_client = eb_client
    @logger = logger
  end

  # Lists applications and their environments
  def list_applications
    @eb_client.describe_applications.applications.each do |application|
      log_application_details(application)
      list_environments(application.application_name)
    end
  rescue Aws::ElasticBeanstalk::Errors::ServiceError => e
    @logger.error("Elastic Beanstalk Service Error: #{e.message}")
  end

  private

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

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

- Para API obter detalhes, consulte [DescribeApplications](#) em AWS SDK for Ruby API Referência.

## ListAvailableSolutionStacks

O código de exemplo a seguir mostra como usar `ListAvailableSolutionStacks`.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
# Manages listing of AWS Elastic Beanstalk solution stacks
# @param [Aws::ElasticBeanstalk::Client] eb_client
# @param [String] filter - Returns subset of results based on match
# @param [Logger] logger
class StackLister
  # Initialize with AWS Elastic Beanstalk client
  def initialize(eb_client, filter, logger: Logger.new($stdout))
    @eb_client = eb_client
    @filter = filter.downcase
    @logger = logger
  end
end
```

```
# Lists and logs Elastic Beanstalk solution stacks
def list_stacks
  stacks = @eb_client.list_available_solution_stacks.solution_stacks
  orig_length = stacks.length
  filtered_length = 0

  stacks.each do |stack|
    if @filter.empty? || stack.downcase.include?(@filter)
      @logger.info(stack)
      filtered_length += 1
    end
  end

  log_summary(filtered_length, orig_length)
rescue Aws::Errors::ServiceError => e
  @logger.error("Error listing solution stacks: #{e.message}")
end

private


# Logs summary of listed stacks
def log_summary(filtered_length, orig_length)
  if @filter.empty?
    @logger.info("Showed #{orig_length} stack(s)")
  else
    @logger.info("Showed #{filtered_length} stack(s) of #{orig_length}")
  end
end
end
```

- Para API obter detalhes, consulte [ListAvailableSolutionStacks](#) em AWS SDK for Ruby API Referência.

## UpdateApplication

O código de exemplo a seguir mostra como usar UpdateApplication.

## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
# Manages deployment of Rails applications to AWS Elastic Beanstalk
class RailsAppDeployer
  def initialize(eb_client, s3_client, app_name, logger: Logger.new($stdout))
    @eb_client = eb_client
    @s3_client = s3_client
    @app_name = app_name
    @logger = logger
  end

  # Deploys the latest application version to Elastic Beanstalk
  def deploy
    create_storage_location
    zip_file_name = create_zip_file
    upload_zip_to_s3(zip_file_name)
    create_and_deploy_new_application_version(zip_file_name)
  end

  private

  # Creates a new S3 storage location for the application
  def create_storage_location
    resp = @eb_client.create_storage_location
    @logger.info("Created storage location in bucket #{resp.s3_bucket}")
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to create storage location: #{e.message}")
  end

  # Creates a ZIP file of the application using git
  def create_zip_file
    zip_file_basename = SecureRandom.urlsafe_base64
    zip_file_name = "#{zip_file_basename}.zip"
    `git archive --format=zip -o #{zip_file_name} HEAD`
    zip_file_name
  end
end
```

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

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

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

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

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

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

- Para API obter detalhes, consulte [UpdateApplication](#) em AWS SDK for Ruby API Referência.

## EventBridge exemplos de uso SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Ruby with EventBridge.

Os cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos


- [Cenários](#)

## Cenários

Criar e acionar uma regra

O exemplo de código a seguir mostra como criar e acionar uma regra na Amazon EventBridge.

## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Chame as funções na ordem correta.

```
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 o tópico especificado do Amazon Simple Notification Service (AmazonSNS) existe entre aqueles fornecidos para essa função.

```
# Checks whether the specified Amazon SNS
# topic exists among those provided to this function.
# This is a helper function that is called by the topic_exists? function.
#
# @param topics [Array] An array of Aws::SNS::Types::Topic objects.
# @param topic_arn [String] The ARN of the topic to find.
# @return [Boolean] true if the topic ARN was found; otherwise, false.
# @example
#   sns_client = Aws::SNS::Client.new(region: 'us-east-1')
#   response = sns_client.list_topics
#   if topic_found?(
#     response.topics,
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
#     puts 'Topic found.'
#   end

def topic_found?(topics, topic_arn)
  topics.each do |topic|
    return true if topic.topic_arn == topic_arn
  end
end
```



```
end
  return false
end
```

Verifica se o tópico especificado existe entre aqueles disponíveis para o chamador na AmazonSNS.

```
# Checks whether the specified topic exists among those available to the
# caller in Amazon SNS.
#
# @param sns_client [Aws::SNS::Client] An initialized Amazon SNS client.
# @param topic_arn [String] The ARN of the topic to find.
# @return [Boolean] true if the topic ARN was found; otherwise, false.
# @example
#   exit 1 unless topic_exists?(
#     Aws::SNS::Client.new(region: 'us-east-1'),
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
def topic_exists?(sns_client, topic_arn)
  puts "Searching for topic with ARN '#{topic_arn}'..."
  response = sns_client.list_topics
  if response.topics.count.positive?
    if topic_found?(response.topics, topic_arn)
      puts "Topic found."
      return true
    end
    while response.next_page? do
      response = response.next_page
      if response.topics.count.positive?
        if topic_found?(response.topics, topic_arn)
          puts "Topic found."
          return true
        end
      end
    end
  end
  puts "Topic not found."
  return false
rescue StandardError => e
  puts "Topic not found: #{e.message}"
  return false
end
```

Crie um tópico na Amazon SNS e depois assine um endereço de e-mail para receber notificações sobre esse tópico.

```
# Creates a topic in Amazon SNS
# and then subscribes an email address to receive notifications to that topic.
#
# @param sns_client [Aws::SNS::Client] An initialized Amazon SNS client.
# @param topic_name [String] The name of the topic to create.
# @param email_address [String] The email address of the recipient to notify.
# @return [String] The ARN of the topic that was created.
# @example
#   puts create_topic(
#     Aws::SNS::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-topic',
#     'mary@example.com'
#   )
def create_topic(sns_client, topic_name, email_address)
  puts "Creating the topic named '#{topic_name}'..."
  topic_response = sns_client.create_topic(name: topic_name)
  puts "Topic created with ARN '#{topic_response.topic_arn}'."
  subscription_response = sns_client.subscribe(
    topic_arn: topic_response.topic_arn,
    protocol: "email",
    endpoint: email_address,
    return_subscription_arn: true
  )
  puts "Subscription created with ARN " \
    "'#{subscription_response.subscription_arn}'. Have the owner of the " \
    "'email address '#{email_address}' check their inbox in a few minutes " \
    "and confirm the subscription to start receiving notification emails."
  return topic_response.topic_arn
rescue StandardError => e
  puts "Error creating or subscribing to topic: #{e.message}"
  return "Error"
end
```

Verifique se a função especificada AWS Identity and Access Management (IAM) existe entre as fornecidas para essa função.

```
# Checks whether the specified AWS Identity and Access Management (IAM)
```

```

# role exists among those provided to this function.
# This is a helper function that is called by the role_exists? function.
#
# @param roles [Array] An array of Aws::IAM::Role objects.
# @param role_arn [String] The ARN of the role to find.
# @return [Boolean] true if the role ARN was found; otherwise, false.
# @example
#   iam_client = Aws::IAM::Client.new(region: 'us-east-1')
#   response = iam_client.list_roles
#   if role_found?(
#     response.roles,
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change'
#   )
#     puts 'Role found.'
#   end
def role_found?(roles, role_arn)
  roles.each do |role|
    return true if role.arn == role_arn
  end
  return false
end
end

```

Verifique se a função especificada existe entre as disponíveis para o chamador em IAM.

```

# Checks whether the specified role exists among those available to the
# caller in AWS Identity and Access Management (IAM).
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_arn [String] The ARN of the role to find.
# @return [Boolean] true if the role ARN was found; otherwise, false.
# @example
#   exit 1 unless role_exists?(
#     Aws::IAM::Client.new(region: 'us-east-1'),
#     'arn:aws:iam::111111111111:role/aws-doc-sdk-examples-ec2-state-change'
#   )
def role_exists?(iam_client, role_arn)
  puts "Searching for role with ARN '#{role_arn}'..."
  response = iam_client.list_roles
  if response.roles.count.positive?
    if role_found?(response.roles, role_arn)
      puts "Role found."
      return true
    end
  end
end

```

```

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

```

### Crie uma função em IAM.

```

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

```

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

Verifica se a EventBridge regra especificada existe entre as fornecidas para essa função.

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

Verifica se a regra especificada existe entre as disponíveis para o chamador. 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."
    end
  end
end
```

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

Crie uma regra em EventBridge.

```
# Creates a rule in Amazon EventBridge.
# This rule is triggered whenever an available instance in
# Amazon EC2 changes to the specified state.
# This rule is designed to be used specifically by this code example.
#
# Prerequisites:
#
# - A role in AWS Identity and Access Management (IAM) that is designed
#   to be used specifically by this code example.
# - A topic in Amazon SNS.
#
# @param cloudwatchevents_client [Aws::CloudWatchEvents::Client]
#   An initialized Amazon EventBridge client.
# @param rule_name [String] The name of the rule to create.
# @param rule_description [String] Some description for this rule.
# @param instance_state [String] The state that available instances in
#   Amazon EC2 must change to, to
#   trigger this rule.
# @param role_arn [String] The Amazon Resource Name (ARN) of the IAM role.
# @param target_id [String] Some identifying string for the rule's target.
# @param topic_arn [String] The ARN of the Amazon SNS topic.
```

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

  put_targets_response = cloudwatchevents_client.put_targets(
    rule: rule_name,
```



```

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

```

Verifique se o grupo de registros especificado existe entre aqueles disponíveis para o chamador no Amazon CloudWatch Logs.

```

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

```

```

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

```

Crie um grupo de CloudWatch registros em Registros.

```

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

```

Grave um evento em um stream de CloudWatch registros em Logs.

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

```

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

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

```

Reinicie uma instância do Amazon Elastic Compute Cloud (AmazonEC2) e adicione informações sobre a atividade relacionada a um stream de log no 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
  )

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

```

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

```

Exibir informações sobre a atividade de uma regra em 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

```

Exibir informações de registro de todos os fluxos de registros em um grupo de CloudWatch registros de registros.

```

# Displays log information for all of the log streams in a log group in
# Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - A log group in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Amazon::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group.
# @example

```

```

# display_log_data(
#   Amazon::CloudWatchLogs::Client.new(region: 'us-east-1'),
#   'aws-doc-sdk-examples-cloudwatch-log'
# )
def display_log_data(cloudwatchlogs_client, log_group_name)
  puts "Attempting to display log stream data for the log group " \
    "named '#{log_group_name}'..."
  describe_log_streams_response = cloudwatchlogs_client.describe_log_streams(
    log_group_name: log_group_name,
    order_by: "LastEventTime",
    descending: true
  )
  if describe_log_streams_response.key?(:log_streams) &&
    describe_log_streams_response.log_streams.count.positive?
    describe_log_streams_response.log_streams.each do |log_stream|
      get_log_events_response = cloudwatchlogs_client.get_log_events(
        log_group_name: log_group_name,
        log_stream_name: log_stream.log_stream_name
      )
      puts "\nLog messages for '#{log_stream.log_stream_name}':"
      puts "-" * (log_stream.log_stream_name.length + 20)
      if get_log_events_response.key?(:events) &&
        get_log_events_response.events.count.positive?
        get_log_events_response.events.each do |event|
          puts event.message
        end
      else
        puts "No log messages for this log stream."
      end
    end
  end
end
rescue StandardError => e
  puts "Error getting information about the log streams or their messages: " \
    "#{e.message}"
end

```

Exiba um lembrete para o chamador limpar manualmente todos AWS os recursos associados dos quais ele não precisa mais.

```

# Displays a reminder to the caller to manually clean up any associated
# AWS resources that they no longer need.

```



```
#
# @param topic_name [String] The name of the Amazon SNS topic.
# @param role_name [String] The name of the IAM role.
# @param rule_name [String] The name of the Amazon EventBridge rule.
# @param log_group_name [String] The name of the Amazon CloudWatch Logs log group.
# @param instance_id [String] The ID of the Amazon EC2 instance.
# @example
#   manual_cleanup_notice(
#     'aws-doc-sdk-examples-topic',
#     'aws-doc-sdk-examples-cloudwatch-events-rule-role',
#     'aws-doc-sdk-examples-ec2-state-change',
#     'aws-doc-sdk-examples-cloudwatch-log',
#     'i-033c48ef067af3dEX'
#   )
def manual_cleanup_notice(
  topic_name, role_name, rule_name, log_group_name, instance_id
)
  puts "-" * 10
  puts "Some of the following AWS resources might still exist in your account."
  puts "If you no longer want to use this code example, then to clean up"
  puts "your AWS account and avoid unexpected costs, you might want to"
  puts "manually delete any of the following resources if they exist:"
  puts "- The Amazon SNS topic named '#{topic_name}'."
  puts "- The IAM role named '#{role_name}'."
  puts "- The Amazon EventBridge rule named '#{rule_name}'."
  puts "- The Amazon CloudWatch Logs log group named '#{log_group_name}'."
  puts "- The Amazon EC2 instance with the ID '#{instance_id}'."
end

# Example usage:
def run_me
  # Properties for the Amazon SNS topic.
  topic_name = "aws-doc-sdk-examples-topic"
  email_address = "mary@example.com"
  # Properties for the IAM role.
  role_name = "aws-doc-sdk-examples-cloudwatch-events-rule-role"
  # Properties for the Amazon EventBridge rule.
  rule_name = "aws-doc-sdk-examples-ec2-state-change"
  rule_description = "Triggers when any available EC2 instance starts."
  instance_state = "running"
  target_id = "sns-topic"
  # Properties for the Amazon EC2 instance.
  instance_id = "i-033c48ef067af3dEX"
  # Properties for displaying the event rule's activity.
```

```
start_time = Time.now - 600 # Go back over the past 10 minutes
                          # (10 minutes * 60 seconds = 600 seconds).

end_time = Time.now
period = 60 # Look back every 60 seconds over the past 10 minutes.
# Properties for the Amazon CloudWatch Logs log group.
log_group_name = "aws-doc-sdk-examples-cloudwatch-log"
# AWS service clients for this code example.
region = "us-east-1"
sts_client = Aws::STS::Client.new(region: region)
sns_client = Aws::SNS::Client.new(region: region)
iam_client = Aws::IAM::Client.new(region: region)
cloudwatchevents_client = Aws::CloudWatchEvents::Client.new(region: region)
ec2_client = Aws::EC2::Client.new(region: region)
cloudwatch_client = Aws::CloudWatch::Client.new(region: region)
cloudwatchlogs_client = Aws::CloudWatchLogs::Client.new(region: region)

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

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

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

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

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

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

# Display how many times the rule was triggered over the past 10 minutes.
display_rule_activity(
```

```
    cloudwatch_client,  
    rule_name,  
    start_time,  
    end_time,  
    period  
  )  
  
  # Display related log data in Amazon CloudWatch Logs.  
  display_log_data(cloudwatchlogs_client, log_group_name)  
  
  # Reminder the caller to clean up any AWS resources that are used  
  # by this code example and are no longer needed.  
  manual_cleanup_notice(  
    topic_name, role_name, rule_name, log_group_name, instance_id  
  )  
end  
  
run_me if $PROGRAM_NAME == __FILE__
```

- Para API obter detalhes, consulte os tópicos a seguir em AWS SDK for Ruby API Referência.
  - [PutEvents](#)
  - [PutRule](#)

## AWS Glue exemplos de uso SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Ruby with AWS Glue.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Conceitos básicos

## Olá AWS Glue

O exemplo de código a seguir mostra como começar a usar o AWS Glue.

### SDKpara Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
```

```
end

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

- Para API obter detalhes, consulte [ListJobs](#) em AWS SDK for Ruby API Referência.

## Tópicos

- [Conceitos básicos](#)
- [Ações](#)

## Conceitos básicos

Aprender os conceitos básicos

O exemplo de código a seguir mostra como:

- Crie um rastreador que rastreie um bucket público do Amazon S3 e gere um banco de dados de metadados formatados. CSV
- Liste informações sobre bancos de dados e tabelas em seu AWS Glue Data Catalog.
- Crie um trabalho para extrair CSV dados do bucket do S3, transformar os dados e carregar a saída JSON formatada em outro bucket do S3.
- Listar informações sobre execuções de tarefas, visualizar dados transformados e limpar recursos.

Para obter mais informações, consulte [Tutorial: Introdução ao AWS Glue Studio](#).

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

## Crie uma classe que envolva as AWS Glue funções usadas no cenário.

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

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

  # Creates a new crawler with the specified configuration.
  #
  # @param name [String] The name of the crawler.
  # @param role_arn [String] The ARN of the IAM role to be used by the crawler.
  # @param db_name [String] The name of the database where the crawler stores its
metadata.
  # @param db_prefix [String] The prefix to be added to the names of tables that the
crawler creates.
  # @param s3_target [String] The S3 path that the crawler will crawl.
  # @return [void]
  def create_crawler(name, role_arn, db_name, db_prefix, s3_target)
    @glue_client.create_crawler(
      name: name,
      role: role_arn,
```

```
        database_name: db_name,
        targets: {
          s3_targets: [
            {
              path: s3_target
            }
          ]
        }
      )
    rescue Aws::Glue::Errors::GlueException => e
      @logger.error("Glue could not create crawler: \n#{e.message}")
      raise
    end

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

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

    # Retrieves information about a specific database.
    #
    # @param name [String] The name of the database to retrieve information about.
    # @return [Aws::Glue::Types::Database, nil] The database object if found, or nil
    if not found.
    def get_database(name)
      response = @glue_client.get_database(name: name)
      response.database
    end
  end
end
```



```
rescue Aws::Glue::Errors::GlueException => e
  @logger.error("Glue could not get database #{name}: \n#{e.message}")
  raise
end

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

# Creates a new job with the specified configuration.
#
# @param name [String] The name of the job.
# @param description [String] The description of the job.
# @param role_arn [String] The ARN of the IAM role to be used by the job.
# @param script_location [String] The location of the ETL script for the job.
# @return [void]
def create_job(name, description, role_arn, script_location)
  @glue_client.create_job(
    name: name,
    description: description,
    role: role_arn,
    command: {
      name: "glueetl",
      script_location: script_location,
      python_version: "3"
    },
    glue_version: "3.0"
  )
rescue Aws::Glue::Errors::GlueException => e
  @logger.error("Glue could not create job #{name}: \n#{e.message}")
  raise
end

# Starts a job run for the specified job.
#
# @param name [String] The name of the job to start the run for.
```

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

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

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

# Retrieves data for a specific job run.
#
# @param job_name [String] The name of the job run to retrieve data for.
# @return [Glue::Types::GetJobRunResponse]
```

```
def get_job_run(job_name, run_id)
  @glue_client.get_job_run(job_name: job_name, run_id: run_id)
rescue Aws::Glue::Errors::GlueException => e
  @logger.error("Glue could not get job runs: \n#{e.message}")
end

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

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

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

# Uploads a job script file to an S3 bucket.
#
# @param file_path [String] The local path of the job script file.
# @param bucket_resource [Aws::S3::Bucket] The S3 bucket resource to upload the
file to.
# @return [void]
```

```
def upload_job_script(file_path, bucket_resource)
  File.open(file_path) do |file|
    bucket_resource.client.put_object({
      body: file,
      bucket: bucket_resource.name,
      key: file_path
    })
  end
rescue Aws::S3::Errors::S3UploadFailedError => e
  @logger.error("S3 could not upload job script: \n#{e.message}")
  raise
end

end
```

Crie uma classe que execute o cenário.

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

  def run(crawler_name, db_name, db_prefix, data_source, job_script, job_name)
    wrapper = GlueWrapper.new(@glue_client, @logger)

    new_step(1, "Create a crawler")
    puts "Checking for crawler #{crawler_name}."
    crawler = wrapper.get_crawler(crawler_name)
    if crawler == false
      puts "Creating crawler #{crawler_name}."
      wrapper.create_crawler(crawler_name, @glue_service_role.arn, db_name,
db_prefix, data_source)
      puts "Successfully created #{crawler_name}:"
      crawler = wrapper.get_crawler(crawler_name)
      puts JSON.pretty_generate(crawler).yellow
    end
    print "\nDone!\n".green

    new_step(2, "Run a crawler to output a database.")
  end
end
```

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

new_step(3, "Query the database.")
database = wrapper.get_database(db_name)
puts "The crawler created database #{db_name}:"
print "#{database}.".yellow
puts "\nThe database contains these tables:"
tables = wrapper.get_tables(db_name)
tables.each_with_index do |table, index|
  print "\t#{index + 1}. #{table['name']}".yellow
end
print "\nDone!\n".green

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

new_step(5, "Start a new job")
job_run_status = nil
job_run_id = wrapper.start_job_run(
  job_name,
  db_name,
  tables[0]["name"],
  @glue_bucket.name
)
puts "Job #{job_name} started. Let's wait for it to run."
until ["SUCCEEDED", "STOPPED", "FAILED", "TIMEOUT"].include?(job_run_status)
```

```
    custom_wait(10)
    job_run = wrapper.get_job_runs(job_name)
    job_run_status = job_run[0]["job_run_state"]
    print "Status check: #{job_name}/#{job_run_id} - #{job_run_status}.".yellow
  end
  print "\nDone!\n".green

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

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

      # Print the first 256 bytes of a run file
      desired_sample_objects = 1
      @glue_bucket.objects.each do |object_summary|
        if object_summary.key.include?("run-")
          if desired_sample_objects > 0
            sample_object = @glue_bucket.object(object_summary.key)
            sample = sample_object.get(range: "bytes=0-255").body.read
            puts "\nSample run file contents:"
            print "#{sample}".yellow
            desired_sample_objects -= 1
          end
        end
      end

      rescue Aws::S3::Errors::ServiceError => e
        logger.error(
          "Couldn't get job run data. Here's why: %s: %s",
          e.response.error.code, e.response.error.message
        )
        raise
      end
    end
    print "\nDone!\n".green

    new_step(7, "Delete job definition and crawler.")
```

```

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

def main

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

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

```

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

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

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

scenario = GlueCrawlerJobScenario.new(
  Aws::Glue::Client.new(region: "us-east-1"),
  iam_role,
  s3_bucket,
  @logger
)

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

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

end
```

Crie um ETL script que seja usado AWS Glue para extrair, transformar e carregar dados durante a execução do trabalho.



```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

"""
These custom arguments must be passed as Arguments to the StartJobRun request.
    --input_database    The name of a metadata database that is contained in your
                        AWS Glue Data Catalog and that contains tables that
describe
                        the data to be processed.
    --input_table       The name of a table in the database that describes the data
to
                        be processed.
    --output_bucket_url An S3 bucket that receives the transformed output data.
"""
args = getResolvedOptions(
    sys.argv, ["JOB_NAME", "input_database", "input_table", "output_bucket_url"]
)
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark_session
job = Job(glueContext)
job.init(args["JOB_NAME"], args)

# Script generated for node S3 Flight Data.
S3FlightData_node1 = glueContext.create_dynamic_frame.from_catalog(
    database=args["input_database"],
    table_name=args["input_table"],
    transformation_ctx="S3FlightData_node1",
)

# This mapping performs two main functions:
# 1. It simplifies the output by removing most of the fields from the data.
# 2. It renames some fields. For example, `fl_date` is renamed to `flight_date`.
ApplyMapping_node2 = ApplyMapping.apply(
    frame=S3FlightData_node1,
    mappings=[
        ("year", "long", "year", "long"),
        ("month", "long", "month", "tinyint"),
        ("day_of_month", "long", "day", "tinyint"),
```

```

    ("fl_date", "string", "flight_date", "string"),
    ("carrier", "string", "carrier", "string"),
    ("fl_num", "long", "flight_num", "long"),
    ("origin_city_name", "string", "origin_city_name", "string"),
    ("origin_state_abr", "string", "origin_state_abr", "string"),
    ("dest_city_name", "string", "dest_city_name", "string"),
    ("dest_state_abr", "string", "dest_state_abr", "string"),
    ("dep_time", "long", "departure_time", "long"),
    ("wheels_off", "long", "wheels_off", "long"),
    ("wheels_on", "long", "wheels_on", "long"),
    ("arr_time", "long", "arrival_time", "long"),
    ("mon", "string", "mon", "string"),
  ],
  transformation_ctx="ApplyMapping_node2",
)

# Script generated for node Revised Flight Data.
RevisedFlightData_node3 = glueContext.write_dynamic_frame.from_options(
  frame=ApplyMapping_node2,
  connection_type="s3",
  format="json",
  connection_options={"path": args["output_bucket_url"], "partitionKeys": []},
  transformation_ctx="RevisedFlightData_node3",
)

job.commit()

```

- Para API obter detalhes, consulte os tópicos a seguir em AWS SDK for Ruby API Referência.
  - [CreateCrawler](#)
  - [CreateJob](#)
  - [DeleteCrawler](#)
  - [DeleteDatabase](#)
  - [DeleteJob](#)
  - [DeleteTable](#)
  - [GetCrawler](#)
  - [GetDatabase](#)
  - [GetDatabases](#)
  - [GetJob](#)

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

## Ações

### CreateCrawler

O código de exemplo a seguir mostra como usar CreateCrawler.

SDKpara Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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

  # Creates a new crawler with the specified configuration.
  #
  # @param name [String] The name of the crawler.
  # @param role_arn [String] The ARN of the IAM role to be used by the crawler.
```

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

- Para API obter detalhes, consulte [CreateCrawler](#) em AWS SDK for Ruby API Referência.

## CreateJob

O código de exemplo a seguir mostra como usar CreateJob.

### SDKpara Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
# The `GlueWrapper` class serves as a wrapper around the AWS Glue API, providing a
simplified interface for common operations.
```

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


  # Creates a new job with the specified configuration.
  #
  # @param name [String] The name of the job.
  # @param description [String] The description of the job.
  # @param role_arn [String] The ARN of the IAM role to be used by the job.
  # @param script_location [String] The location of the ETL script for the job.
  # @return [void]
  def create_job(name, description, role_arn, script_location)
    @glue_client.create_job(
      name: name,
      description: description,
      role: role_arn,
      command: {
        name: "glueetl",
        script_location: script_location,
        python_version: "3"
      },
      glue_version: "3.0"
    )
  rescue Aws::Glue::Errors::GlueException => e
    @logger.error("Glue could not create job #{name}: \n#{e.message}")
    raise
  end
end
```

- Para API obter detalhes, consulte [CreateJob](#) em AWS SDK for Ruby API Referência.

## DeleteCrawler

O código de exemplo a seguir mostra como usar DeleteCrawler.

## SDK para Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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


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

- Para API obter detalhes, consulte [DeleteCrawler](#) em AWS SDK for Ruby API Referência.

## DeleteDatabase

O código de exemplo a seguir mostra como usar DeleteDatabase.

## SDK para Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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

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

- Para API obter detalhes, consulte [DeleteDatabase](#) em AWS SDK for Ruby API Referência.

## DeleteJob

O código de exemplo a seguir mostra como usar DeleteJob.

## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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

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


- Para API obter detalhes, consulte [DeleteJob](#) em AWS SDK for Ruby APIReferência.

## DeleteTable

O código de exemplo a seguir mostra como usar DeleteTable.



## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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


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

- Para API obter detalhes, consulte [DeleteTable](#) em AWS SDK for Ruby APIReferência.

## GetCrawler

O código de exemplo a seguir mostra como usar GetCrawler.

## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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

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

- Para API obter detalhes, consulte [GetCrawler](#) em AWS SDK for Ruby APIReferência.

## GetDatabase

O código de exemplo a seguir mostra como usar GetDatabase.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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

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

- Para API obter detalhes, consulte [GetDatabase](#) em AWS SDK for Ruby API Referência.

## GetJobRun

O código de exemplo a seguir mostra como usar GetJobRun.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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

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

- Para API obter detalhes, consulte [GetJobRun](#) em AWS SDK for Ruby API Referência.

## GetJobRuns

O código de exemplo a seguir mostra como usar GetJobRuns.

## SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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

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

- Para API obter detalhes, consulte [GetJobRuns](#) em AWS SDK for Ruby API Referência.

## GetTables

O código de exemplo a seguir mostra como usar GetTables.

## SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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

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

- Para API obter detalhes, consulte [GetTables](#) em AWS SDK for Ruby API Referência.

## ListJobs

O código de exemplo a seguir mostra como usar ListJobs.

## SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
```

- Para API obter detalhes, consulte [ListJobs](#) em AWS SDK for Ruby APIReferência.

## StartCrawler

O código de exemplo a seguir mostra como usar StartCrawler.

## SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
```


- Para API obter detalhes, consulte [StartCrawler](#) em AWS SDK for Ruby API Referência.

## StartJobRun

O código de exemplo a seguir mostra como usar StartJobRun.



## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
```

- Para API obter detalhes, consulte [StartJobRun](#) em AWS SDK for Ruby API Referência.

## IAM exemplos de uso SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Ruby with IAM.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Os cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Conceitos básicos

Olá, IAM

Os exemplos de código a seguir mostram como começar a usar IAM.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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

# IAMManager is a class responsible for managing IAM operations
# such as listing all IAM policies in the current AWS account.
```

```
class IAMManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

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

    paginator = @client.list_policies
    policies = []

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

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

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

- Para API obter detalhes, consulte [ListPolicies](#) em AWS SDK for Ruby API Referência.

## Tópicos

- [Ações](#)
- [Cenários](#)

# Ações

## AttachRolePolicy

O código de exemplo a seguir mostra como usar `AttachRolePolicy`.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Este exemplo de módulo lista, cria, anexa e desconecta políticas de perfis.

```
# 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
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
```

```
end

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

- Para API obter detalhes, consulte [AttachRolePolicy](#) em AWS SDK for Ruby API Referência.

## AttachUserPolicy

O código de exemplo a seguir mostra como usar AttachUserPolicy.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
# Attaches a policy to a user
#
# @param user_name [String] The name of the user
# @param policy_arn [String] The Amazon Resource Name (ARN) of the policy
# @return [Boolean] true if successful, false otherwise
def attach_policy_to_user(user_name, policy_arn)
  @iam_client.attach_user_policy(
    user_name: user_name,
```

```

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

```

- Para API obter detalhes, consulte [AttachUserPolicy](#) em AWS SDK for Ruby API Referência.

## CreateAccessKey

O código de exemplo a seguir mostra como usar `CreateAccessKey`.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Este exemplo de módulo lista, cria, desativa e exclui chaves de acesso.

```

# 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
  end
end

```

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

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

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



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

- Para API obter detalhes, consulte [CreateAccessKey](#) em AWS SDK for Ruby API Referência.

## CreateAccountAlias

O código de exemplo a seguir mostra como usar CreateAccountAlias.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Listar, criar e excluir aliases da conta.

```
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
end
```

```
end

# Lists available AWS account aliases.
def list_aliases
  response = @iam_client.list_account_aliases

  if response.account_aliases.count.positive?
    @logger.info("Account aliases are:")
    response.account_aliases.each { |account_alias| @logger.info("#{account_alias}") }
  else
    @logger.info("No account aliases found.")
  end
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error listing account aliases: #{e.message}")
end

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

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

- Para API obter detalhes, consulte [CreateAccountAlias](#) em AWS SDK for Ruby API Referência.

## CreatePolicy

O código de exemplo a seguir mostra como usar CreatePolicy.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Este exemplo de módulo lista, cria, anexa e desconecta políticas de perfis.

```
# 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
end
```

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

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

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

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

- Para API obter detalhes, consulte [CreatePolicy](#) em AWS SDK for Ruby API Referência.

## CreateRole

O código de exemplo a seguir mostra como usar CreateRole.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
# Creates a role and attaches policies to it.
#
# @param role_name [String] The name of the role.
# @param assume_role_policy_document [Hash] The trust relationship policy
document.
# @param policy_arns [Array<String>] The ARNs of the policies to attach.
# @return [String, nil] The ARN of the new role if successful, or nil if an error
occurred.
def create_role(role_name, assume_role_policy_document, policy_arns)
```

```

response = @iam_client.create_role(
  role_name: role_name,
  assume_role_policy_document: assume_role_policy_document.to_json
)
role_arn = response.role.arn

policy_arns.each do |policy_arn|
  @iam_client.attach_role_policy(
    role_name: role_name,
    policy_arn: policy_arn
  )
end

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

```

- Para API obter detalhes, consulte [CreateRole](#) em AWS SDK for Ruby API Referência.

## CreateServiceLinkedRole

O código de exemplo a seguir mostra como usar CreateServiceLinkedRole.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```

# Creates a service-linked role
#
# @param service_name [String] The service name to create the role for.
# @param description [String] The description of the service-linked role.
# @param suffix [String] Suffix for customizing role name.
# @return [String] The name of the created role
def create_service_linked_role(service_name, description, suffix)
  response = @iam_client.create_service_linked_role(

```

```

    aws_service_name: service_name, description: description, custom_suffix:
suffix,)
    role_name = response.role.role_name
    @logger.info("Created service-linked role #{role_name}.")
    role_name
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't create service-linked role for #{service_name}. Here's
why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
end

```

- Para API obter detalhes, consulte [CreateServiceLinkedRole](#) em AWS SDK for Ruby API Referência.

## CreateUser

O código de exemplo a seguir mostra como usar CreateUser.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```

# Creates a user and their login profile
#
# @param user_name [String] The name of the user
# @param initial_password [String] The initial password for the user
# @return [String, nil] The ID of the user if created, or nil if an error occurred
def create_user(user_name, initial_password)
  response = @iam_client.create_user(user_name: user_name)
  @iam_client.wait_until(:user_exists, user_name: user_name)
  @iam_client.create_login_profile(
    user_name: user_name,
    password: initial_password,
    password_reset_required: true
  )
)

```

```
@logger.info("User '#{user_name}' created successfully.")
response.user.user_id
rescue Aws::IAM::Errors::EntityAlreadyExists
  @logger.error("Error creating user '#{user_name}': user already exists.")
  nil
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error creating user '#{user_name}': #{e.message}")
  nil
end
```

- Para API obter detalhes, consulte [CreateUser](#) em AWS SDK for Ruby API Referência.

## DeleteAccessKey

O código de exemplo a seguir mostra como usar DeleteAccessKey.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Este exemplo de módulo lista, cria, desativa e exclui chaves de acesso.

```
# 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}'.")
    end
  end
end
```



```
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity => e
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

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

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

```
end

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

- Para API obter detalhes, consulte [DeleteAccessKey](#) em AWS SDK for Ruby API Referência.

## DeleteAccountAlias

O código de exemplo a seguir mostra como usar DeleteAccountAlias.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Listar, criar e excluir aliases da conta.

```
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
  end
end
```

```
@logger = logger
end

# Lists available AWS account aliases.
def list_aliases
  response = @iam_client.list_account_aliases

  if response.account_aliases.count.positive?
    @logger.info("Account aliases are:")
    response.account_aliases.each { |account_alias| @logger.info("#{account_alias}") }
  else
    @logger.info("No account aliases found.")
  end
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error listing account aliases: #{e.message}")
end

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

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

- Para API obter detalhes, consulte [DeleteAccountAlias](#)em AWS SDK for Ruby APIReferência.

## DeleteRole

O código de exemplo a seguir mostra como usar DeleteRole.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
# Deletes a role and its attached policies.
#
# @param role_name [String] The name of the role to delete.
def delete_role(role_name)
  begin
    # Detach and delete attached policies
    @iam_client.list_attached_role_policies(role_name: role_name).each do |
response|
      response.attached_policies.each do |policy|
        @iam_client.detach_role_policy({
          role_name: role_name,
          policy_arn: policy.policy_arn
        })

        # Check if the policy is a customer managed policy (not AWS managed)
        unless policy.policy_arn.include?("aws:policy/")
          @iam_client.delete_policy({ policy_arn: policy.policy_arn })
          @logger.info("Deleted customer managed policy #{policy.policy_name}.")
        end
      end
    end
  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}")
end
```

```
    raise
  end
end
```

- Para API obter detalhes, consulte [DeleteRole](#) em AWS SDK for Ruby API Referência.

## DeleteServerCertificate

O código de exemplo a seguir mostra como usar DeleteServerCertificate.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Listar, atualizar e excluir certificados de servidor.

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

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
      server_certificate_name: name,
      certificate_body: certificate_body,
      private_key: private_key,
    })

    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
  end
end
```

```
    false
  end

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

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

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

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

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

- Para API obter detalhes, consulte [DeleteServerCertificate](#) em AWS SDK for Ruby API Referência.

## DeleteServiceLinkedRole

O código de exemplo a seguir mostra como usar DeleteServiceLinkedRole.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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)
  end
end
```

```
        sleep(3)
      end
    end

    # Handles deletion error
    #
    # @param e [Aws::Errors::ServiceError] The error encountered during deletion
    # @param role_name [String] The name of the role attempted to delete
    def handle_deletion_error(e, role_name)
      unless e.code == "NoSuchEntity"
        @logger.error("Couldn't delete #{role_name}. Here's why:")
        @logger.error("\t#{e.code}: #{e.message}")
        raise
      end
    end
  end
end
```

- Para API obter detalhes, consulte [DeleteServiceLinkedRole](#) em AWS SDK for Ruby API Referência.

## DeleteUser

O código de exemplo a seguir mostra como usar DeleteUser.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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 })
  end
end
```



```

    @logger.info("Deleted access key #{key.access_key_id} for user
'#{user_name}'.")
  end

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

```

- Para API obter detalhes, consulte [DeleteUser](#) em AWS SDK for Ruby API Referência.

## DeleteUserPolicy

O código de exemplo a seguir mostra como usar DeleteUserPolicy.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```

# Deletes a user and their associated resources
#
# @param user_name [String] The name of the user to delete
def delete_user(user_name)
  user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
  user.each do |key|
    @iam_client.delete_access_key({ access_key_id: key.access_key_id, user_name:
user_name })
    @logger.info("Deleted access key #{key.access_key_id} for user
'#{user_name}'.")
  end

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

```

- Para API obter detalhes, consulte [DeleteUserPolicy](#) em AWS SDK for Ruby APIReferência.

## DetachRolePolicy

O código de exemplo a seguir mostra como usar DetachRolePolicy.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Este exemplo de módulo lista, cria, anexa e desconecta políticas de perfis.

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

```
    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)
  end
end
```

```
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error listing policies attached to role: #{e.message}")
  []
end

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

- Para API obter detalhes, consulte [DetachRolePolicy](#) em AWS SDK for Ruby API Referência.

## DetachUserPolicy

O código de exemplo a seguir mostra como usar DetachUserPolicy.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
# Detaches a policy from a user
#
# @param user_name [String] The name of the user
# @param policy_arn [String] The ARN of the policy to detach
```

```
# @return [Boolean] true if the policy was successfully detached, false otherwise
def detach_user_policy(user_name, policy_arn)
  @iam_client.detach_user_policy(
    user_name: user_name,
    policy_arn: policy_arn
  )
  @logger.info("Policy '#{policy_arn}' detached from user '#{user_name}'
successfully.")
  true
rescue Aws::IAM::Errors::NoSuchEntity
  @logger.error("Error detaching policy: Policy or user does not exist.")
  false
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error detaching policy from user '#{user_name}': #{e.message}")
  false
end
```

- Para API obter detalhes, consulte [DetachUserPolicy](#) em AWS SDK for Ruby API Referência.

## GetAccountPasswordPolicy

O código de exemplo a seguir mostra como usar `GetAccountPasswordPolicy`.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
end
```

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

- Para API obter detalhes, consulte [GetAccountPasswordPolicy](#) em AWS SDK for Ruby API Referência.

## GetPolicy

O código de exemplo a seguir mostra como usar GetPolicy.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
end
```

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

- Para API obter detalhes, consulte [GetPolicy](#) em AWS SDK for Ruby API Referência.

## GetRole

O código de exemplo a seguir mostra como usar GetRole.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
# Gets data about a role.
#
# @param name [String] The name of the role to look up.
# @return [Aws::IAM::Role] The retrieved role.
def get_role(name)
  role = @iam_client.get_role({
    role_name: name,
  }).role
  puts("Got data for role '#{role.role_name}'. Its ARN is '#{role.arn}'.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't get data for role '#{name}' Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  role
end
```

- Para API obter detalhes, consulte [GetRole](#) em AWS SDK for Ruby API Referência.

## GetUser

O código de exemplo a seguir mostra como usar GetUser.

### SDK para Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
# Retrieves a user's details
#
# @param user_name [String] The name of the user to retrieve
# @return [Aws::IAM::Types::User, nil] The user object if found, or nil if an
error occurred
def get_user(user_name)
  response = @iam_client.get_user(user_name: user_name)
  response.user
rescue Aws::IAM::Errors::NoSuchEntity
  @logger.error("User '#{user_name}' not found.")
  nil
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error retrieving user '#{user_name}': #{e.message}")
  nil
end
```

- Para API obter detalhes, consulte [GetUser](#) em AWS SDK for Ruby API Referência.

## ListAccessKeys

O código de exemplo a seguir mostra como usar ListAccessKeys.



## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Este exemplo de módulo lista, cria, desativa e exclui chaves de acesso.

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "AccessKeyManager"
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity => e
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

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

```
@logger.info("Access key created for user '#{user_name}':
#{access_key.access_key_id}")
  access_key
rescue Aws::IAM::Errors::LimitExceeded => e
  @logger.error("Error creating access key: limit exceeded. Cannot create more.")
  nil
rescue StandardError => e
  @logger.error("Error creating access key: #{e.message}")
  nil
end

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

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

- Para API obter detalhes, consulte [ListAccessKeys](#) em AWS SDK for Ruby API Referência.

## ListAccountAliases

O código de exemplo a seguir mostra como usar ListAccountAliases.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Listar, criar e excluir aliases da conta.

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info("Account aliases are:")
      response.account_aliases.each { |account_alias| @logger.info("#{account_alias}") }
    else
      @logger.info("No account aliases found.")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end
end
```

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

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

- Para API obter detalhes, consulte [ListAccountAliases](#) em AWS SDK for Ruby API Referência.

## ListAttachedRolePolicies

O código de exemplo a seguir mostra como usar `ListAttachedRolePolicies`.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Este exemplo de módulo lista, cria, anexa e desconecta políticas de perfis.

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

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

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

```
end

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

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

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

```
end
end
```

- Para API obter detalhes, consulte [ListAttachedRolePolicies](#) em AWS SDK for Ruby API Referência.

## ListGroups

O código de exemplo a seguir mostra como usar ListGroups.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
```

```
end
end
```

- Para API obter detalhes, consulte [ListGroupsem AWS SDK for Ruby APIReferência](#).

## ListPolicies

O código de exemplo a seguir mostra como usar ListPolicies.

### SDKpara Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Este exemplo de módulo lista, cria, anexa e desconecta políticas de perfis.

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

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  end
end
```



```

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

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

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

# Lists policy ARNs attached to a role
#
# @param role_name [String] The name of the role
# @return [Array<String>] List of policy ARNs
def list_attached_policy_arns(role_name)

```

```

    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

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

```

- Para API obter detalhes, consulte [ListPolicies](#) em AWS SDK for Ruby API Referência.

## ListRolePolicies

O código de exemplo a seguir mostra como usar ListRolePolicies.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```

# Lists policy ARNs attached to a role
#
# @param role_name [String] The name of the role

```

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

- Para API obter detalhes, consulte [ListRolePolicies](#) em AWS SDK for Ruby API Referência.

## ListRoles

O código de exemplo a seguir mostra como usar ListRoles.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
# Lists IAM roles up to a specified count.
# @param count [Integer] the maximum number of roles to list.
# @return [Array<String>] the names of the roles.
def list_roles(count)
  role_names = []
  roles_counted = 0

  @iam_client.list_roles.each_page do |page|
    page.roles.each do |role|
      break if roles_counted >= count
      @logger.info("\t#{roles_counted + 1}: #{role.role_name}")
      role_names << role.role_name
      roles_counted += 1
    end
    break if roles_counted >= count
  end

  role_names
```

```
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Couldn't list roles for the account. Here's why:")
  @logger.error("\t#{e.code}: #{e.message}")
  raise
end
```

- Para API obter detalhes, consulte [ListRoles](#) em AWS SDK for Ruby API Referência.

## ListSAMLProviders

O código de exemplo a seguir mostra como usar ListSAMLProviders.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
  end

  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't list SAML providers. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
  end
end
```

```
    raise
  end
end
```

- Para API obter detalhes, consulte [ListSAMLProviders](#) em AWS SDK for Ruby API Referência.

## ListServerCertificates

O código de exemplo a seguir mostra como usar ListServerCertificates.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Listar, atualizar e excluir certificados de servidor.

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

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
      server_certificate_name: name,
      certificate_body: certificate_body,
      private_key: private_key,
    })

    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
  end
end
```

```
    false
  end

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

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

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

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

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

- Para API obter detalhes, consulte [ListServerCertificates](#) em AWS SDK for Ruby API Referência.

## ListUsers

O código de exemplo a seguir mostra como usar ListUsers.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
# Lists all users in the AWS account
#
# @return [Array<Aws::IAM::Types::User>] An array of user objects
def list_users
  users = []
  @iam_client.list_users.each_page do |page|
    page.users.each do |user|
      users << user
    end
  end
  users
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error listing users: #{e.message}")
  []
end
```

- Para API obter detalhes, consulte [ListUsers](#) em AWS SDK for Ruby API Referência.

## PutUserPolicy

O código de exemplo a seguir mostra como usar PutUserPolicy.

## SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
# Creates an inline policy for a specified user.
# @param username [String] The name of the IAM user.
# @param policy_name [String] The name of the policy to create.
# @param policy_document [String] The JSON policy document.
# @return [Boolean]
def create_user_policy(username, policy_name, policy_document)
  @iam_client.put_user_policy({
    user_name: username,
    policy_name: policy_name,
    policy_document: policy_document
  })
  @logger.info("Policy #{policy_name} created for user #{username}.")
  true
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Couldn't create policy #{policy_name} for user #{username}.
Here's why:")
  @logger.error("\t#{e.code}: #{e.message}")
  false
end
```

- Para API obter detalhes, consulte [PutUserPolicy](#) em AWS SDK for Ruby APIReferência.

## UpdateServerCertificate

O código de exemplo a seguir mostra como usar UpdateServerCertificate.



## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Listar, atualizar e excluir certificados de servidor.

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

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
      server_certificate_name: name,
      certificate_body: certificate_body,
      private_key: private_key,
    })

    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

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

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

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

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

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

- Para API obter detalhes, consulte [UpdateServerCertificate](#) em AWS SDK for Ruby API Referência.

## UpdateUser

O código de exemplo a seguir mostra como usar UpdateUser.

## SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
# Updates an IAM user's name
#
# @param current_name [String] The current name of the user
# @param new_name [String] The new name of the user
def update_user_name(current_name, new_name)
  @iam_client.update_user(user_name: current_name, new_user_name: new_name)
  true
rescue StandardError => e
  @logger.error("Error updating user name from '#{current_name}' to '#{new_name}':
#{e.message}")
  false
end
```

- Para API obter detalhes, consulte [UpdateUser](#) em AWS SDK for Ruby APIReferência.

## Cenários

### Criar um usuário e assumir uma função

O exemplo de código a seguir mostra como criar um usuário e assumir um perfil.

### Warning

Para evitar riscos de segurança, não use IAM usuários para autenticação ao desenvolver software específico ou trabalhar com dados reais. Em vez disso, use federação com um provedor de identidade, como [AWS IAM Identity Center](#).

- Crie um usuário sem permissões.
- Crie uma função que conceda permissão para listar os buckets do Amazon S3 para a conta.

- Adicione uma política para permitir que o usuário assuma a função.
- Assuma o perfil e liste buckets do S3 usando credenciais temporárias, depois limpe os recursos.

## SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Crie um IAM usuário e uma função que concedam permissão para listar buckets do Amazon S3. O usuário só tem direitos para assumir a função. Após assumir a função, use credenciais temporárias para listar os buckets para a conta.

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

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

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

  # Creates a user.
  #
  # @param user_name [String] The name to give the user.
  # @return [Aws::IAM::User] The newly created user.
  def create_user(user_name)
    user = @iam_client.create_user(user_name: user_name).user
    @logger.info("Created demo user named #{user.user_name}.")
    rescue Aws::Errors::ServiceError => e
  end
end
```

```
@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
  end
end

```

```
        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
  end

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

  # Gets temporary credentials that can be used to assume a role.
  #
  # @param role_arn [String] The ARN of the role that is assumed when these
  # credentials
  #
  #           are used.
  # @param sts_client [AWS::STS::Client] An AWS STS client.
  # @return [Aws::AssumeRoleCredentials] The credentials that can be used to assume
  # the role.
  def assume_role(role_arn, sts_client)
    credentials = Aws::AssumeRoleCredentials.new(
      client: sts_client,
      role_arn: role_arn,
      role_session_name: "create-use-assume-role-scenario"
    )
    @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
    credentials
  end

  # Deletes a role. If the role has policies attached, they are detached and
  # deleted before the role is deleted.
```



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

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

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

# Runs the IAM create a user and assume a role scenario.
def run_scenario(scenario)
  puts("-" * 88)
  puts("Welcome to the IAM create a user and assume a role demo!")
  puts("-" * 88)
```

```

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

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

```

- Para API obter detalhes, consulte os tópicos a seguir em AWS SDK for Ruby API Referência.
  - [AttachRolePolicy](#)
  - [CreateAccessKey](#)
  - [CreatePolicy](#)
  - [CreateRole](#)
  - [CreateUser](#)
  - [DeleteAccessKey](#)

- [DeletePolicy](#)
- [DeleteRole](#)
- [DeleteUser](#)
- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

## Exemplos de Kinesis usando SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Ruby with Kinesis.

Cada exemplo inclui um link para o código-fonte completo, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Exemplos sem servidor](#)

## Exemplos sem servidor

Invocar uma função do Lambda em um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um stream do Kinesis. A função recupera a carga útil do Kinesis, decodifica do Base64 e registra o conteúdo do registro em log.

### SDKpara Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Consumir um evento do Kinesis com o Lambda usando 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
```

## Relatando falhas de itens em lote para funções do Lambda com um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um stream do Kinesis. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

### SDKpara Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Relatar falhas de item em lote do Kinesis com o Lambda usando 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 exemplos de uso SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Ruby with AWS KMS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

## Ações

### CreateKey

O código de exemplo a seguir mostra como usar CreateKey.

SDKpara Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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

# Create a AWS KMS key.
# As long we are only encrypting small amounts of data (4 KiB or less) directly,
# a KMS key is fine for our purposes.
# For larger amounts of data,
# use the KMS key to encrypt a data encryption key (DEK).

client = Aws::KMS::Client.new

resp = client.create_key({
  tags: [
    {
      tag_key: "CreatedBy",
      tag_value: "ExampleUser"
    }
  ]
})
```

```
        }
      ]
    })

puts resp.key_metadata.key_id
```

- Para API obter detalhes, consulte [CreateKey](#) em AWS SDK for Ruby API Referência.

## Decrypt

O código de exemplo a seguir mostra como usar Decrypt.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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

# Decrypted blob

blob =
  "01020200785d68faeec386af1057904926253051eb2919d3c16078badf65b808b26dd057c101747cadf3593596"
blob_packed = [blob].pack("H*")

client = Aws::KMS::Client.new(region: "us-west-2")

resp = client.decrypt({
  ciphertext_blob: blob_packed
})

puts "Raw text: "
puts resp.plaintext
```

- Para API obter detalhes, consulte [Decrypt](#) in Reference.AWS SDK for Ruby API

## Encrypt

O código de exemplo a seguir mostra como usar Encrypt.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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

# ARN of the AWS KMS key.
#
# Replace the fictitious key ARN with a valid key ID

keyId = "arn:aws:kms:us-
west-2:111122223333:key/1234abcd-12ab-34cd-56ef-1234567890ab"

text = "1234567890"

client = Aws::KMS::Client.new(region: "us-west-2")

resp = client.encrypt({
  key_id: keyId,
  plaintext: text,
})

# Display a readable version of the resulting encrypted blob.
puts "Blob:"
puts resp.ciphertext_blob.unpack("H*")
```


- Para API obter detalhes, consulte [Criptografar](#) em AWS SDK for Ruby APIreferência.

## ReEncrypt

O código de exemplo a seguir mostra como usar ReEncrypt.



## SDK para Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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

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

blob =
  "01020200785d68faeec386af1057904926253051eb2919d3c16078badf65b808b26dd057c101747cadf3593596"
sourceCiphertextBlob = [blob].pack("H*")

# Replace the fictitious key ARN with a valid key ID

destinationKeyId = "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321"

client = Aws::KMS::Client.new(region: "us-west-2")

resp = client.re_encrypt({
  ciphertext_blob: sourceCiphertextBlob,
  destination_key_id: destinationKeyId
})

# Display a readable version of the resulting re-encrypted blob.
puts "Blob:"
puts resp.ciphertext_blob.unpack("H*")
```

- Para API obter detalhes, consulte [ReEncrypt](#) em AWS SDK for Ruby API Referência.

## Exemplos de Lambda usando SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Ruby com o Lambda.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Os cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

## Conceitos básicos

### Olá, Lambda

Os exemplos de código a seguir mostram como começar a usar o Lambda.

## SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
end
```

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

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

list_lambda_functions if __FILE__ == $PROGRAM_NAME
```

- Para API obter detalhes, consulte [ListFunctions](#) em AWS SDK for Ruby API Referência.

## Tópicos

- [Ações](#)
- [Cenários](#)
- [Exemplos sem servidor](#)

## Ações

### CreateFunction

O código de exemplo a seguir mostra como usar CreateFunction.

#### SDKpara Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
class LambdaWrapper
  attr_accessor :lambda_client
```

```
def initialize
  @lambda_client = Aws::Lambda::Client.new
  @logger = Logger.new($stdout)
  @logger.level = Logger::WARN
end

# Deploys a Lambda function.
#
# @param function_name: The name of the Lambda function.
# @param handler_name: The fully qualified name of the handler function. This
#                       must include the file name and the function name.
# @param role_arn: The IAM role to use for the function.
# @param deployment_package: The deployment package that contains the function
#                             code in .zip format.
# @return: The Amazon Resource Name (ARN) of the newly created function.
def create_function(function_name, handler_name, role_arn, deployment_package)
  response = @lambda_client.create_function({
    role: role_arn.to_s,
    function_name: function_name,
    handler: handler_name,
    runtime: "ruby2.7",
    code: {
      zip_file: deployment_package
    },
    environment: {
      variables: {
        "LOG_LEVEL" => "info"
      }
    }
  })
  @lambda_client.wait_until(:function_active_v2, { function_name: function_name})
do |w|
  w.max_attempts = 5
  w.delay = 5
end
  response
rescue Aws::Lambda::Errors::ServiceException => e
  @logger.error("There was an error creating #{function_name}:\n #{e.message}")
rescue Aws::Waiters::Errors::WaiterFailed => e
  @logger.error("Failed waiting for #{function_name} to activate:\n #{e.message}")
end
end
```

- Para API obter detalhes, consulte [CreateFunction](#) em AWS SDK for Ruby API Referência.

## DeleteFunction

O código de exemplo a seguir mostra como usar DeleteFunction.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Deletes a Lambda function.
  # @param function_name: The name of the function to delete.
  def delete_function(function_name)
    print "Deleting function: #{function_name}..."
    @lambda_client.delete_function(
      function_name: function_name
    )
    print "Done!".green
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error deleting #{function_name}:\n #{e.message}")
  end
end
```

- Para API obter detalhes, consulte [DeleteFunction](#) em AWS SDK for Ruby API Referência.

## GetFunction

O código de exemplo a seguir mostra como usar GetFunction.

## SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end


  # Gets data about a Lambda function.
  #
  # @param function_name: The name of the function.
  # @return response: The function data, or nil if no such function exists.
  def get_function(function_name)
    @lambda_client.get_function(
      {
        function_name: function_name
      }
    )
  rescue Aws::Lambda::Errors::ResourceNotFoundException => e
    @logger.debug("Could not find function: #{function_name}:\n #{e.message}")
    nil
  end
end
```

- Para API obter detalhes, consulte [GetFunction](#) em AWS SDK for Ruby API Referência.

## Invoke

O código de exemplo a seguir mostra como usar Invoke.

## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end


  # Invokes a Lambda function.
  # @param function_name [String] The name of the function to invoke.
  # @param payload [nil] Payload containing runtime parameters.
  # @return [Object] The response from the function invocation.
  def invoke_function(function_name, payload = nil)
    params = { function_name: function_name }
    params[:payload] = payload unless payload.nil?
    @lambda_client.invoke(params)
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error executing #{function_name}:\n #{e.message}")
  end
end
```

- Para API obter detalhes, consulte [Invoke](#) in AWS SDK for Ruby APIReference.

## ListFunctions

O código de exemplo a seguir mostra como usar ListFunctions.

## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

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


- Para API obter detalhes, consulte [ListFunctions](#) em AWS SDK for Ruby API Referência.

## UpdateFunctionCode

O código de exemplo a seguir mostra como usar UpdateFunctionCode.



## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Updates the code for a Lambda function by submitting a .zip archive that
  # contains
  # the code for the function.

  # @param function_name: The name of the function to update.
  # @param deployment_package: The function code to update, packaged as bytes in
  #                               .zip format.
  # @return: Data about the update, including the status.
  def update_function_code(function_name, deployment_package)
    @lambda_client.update_function_code(
      function_name: function_name,
      zip_file: deployment_package
    )
    @lambda_client.wait_until(:function_updated_v2, { function_name: function_name})
  do |w|
    w.max_attempts = 5
    w.delay = 5
  end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating function code for: #{function_name}:
  \n #{e.message}")
    nil
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to update:\n #{e.message}")
  end
end
```

- Para API obter detalhes, consulte [UpdateFunctionCode](#) em AWS SDK for Ruby APIReferência.

## UpdateFunctionConfiguration

O código de exemplo a seguir mostra como usar UpdateFunctionConfiguration.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

  # Updates the environment variables for a Lambda function.
  # @param function_name: The name of the function to update.
  # @param log_level: The log level of the function.
  # @return: Data about the update, including the status.
  def update_function_configuration(function_name, log_level)
    @lambda_client.update_function_configuration({
      function_name: function_name,
      environment: {
        variables: {
          "LOG_LEVEL" => log_level
        }
      }
    })
    @lambda_client.wait_until(:function_updated_v2, { function_name: function_name})
  end
end

w = LambdaWrapper.new
w.max_attempts = 5
w.delay = 5
```

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

- Para API obter detalhes, consulte [UpdateFunctionConfiguration](#) em AWS SDK for Ruby API Referência.

## Cenários

### Criar uma aplicação para analisar o feedback dos clientes

O exemplo de código a seguir mostra como criar uma aplicação que analisa os cartões de comentários dos clientes, os traduz do idioma original, determina seus sentimentos e gera um arquivo de áudio do texto traduzido.

### SDK para Ruby

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter o código-fonte e as instruções de implantação, consulte o projeto em [GitHub](#).

### Serviços utilizados neste exemplo

- Amazon Comprehend
- Lambda

- Amazon Polly
- Amazon Textract
- Amazon Translate

## Conceitos básicos de funções

O exemplo de código a seguir mostra como:

- Criar uma IAM função e uma função Lambda e, em seguida, fazer o upload do código do manipulador.
- Invocar essa função com um único parâmetro e receber resultados.
- Atualizar o código de função e configurar usando uma variável de ambiente.
- Invocar a função com novos parâmetros e receber resultados. Exibir o log de execução retornado.
- Listar as funções para sua conta e limpar os recursos.

Para obter mais informações, consulte [Criar uma função do Lambda no console](#).

## SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Configure IAM as permissões de pré-requisito para uma função Lambda capaz de gravar registros.

```
# Get an AWS Identity and Access Management (IAM) role.
#
# @param iam_role_name: The name of the role to retrieve.
# @param action: Whether to create or destroy the IAM apparatus.
# @return: The IAM role.
def manage_iam(iam_role_name, action)
  role_policy = {
    'Version': "2012-10-17",
    'Statement': [
      {
```

```

        'Effect': "Allow",
        'Principal': {
          'Service': "lambda.amazonaws.com"
        },
        'Action': "sts:AssumeRole"
      }
    ]
  }
  case action
  when "create"
    role = $iam_client.create_role(
      role_name: iam_role_name,
      assume_role_policy_document: role_policy.to_json
    )
    $iam_client.attach_role_policy(
      {
        policy_arn: "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole",
        role_name: iam_role_name
      }
    )
    $iam_client.wait_until(:role_exists, { role_name: iam_role_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
    @logger.debug("Successfully created IAM role: #{role['role']['arn']}")
    @logger.debug("Enforcing a 10-second sleep to allow IAM role to activate
fully.")
    sleep(10)
    return role, role_policy.to_json
  when "destroy"
    $iam_client.detach_role_policy(
      {
        policy_arn: "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole",
        role_name: iam_role_name
      }
    )
    $iam_client.delete_role(
      role_name: iam_role_name
    )
    @logger.debug("Detached policy & deleted IAM role: #{iam_role_name}")
  else
    raise "Incorrect action provided. Must provide 'create' or 'destroy'"
  end
end

```

```
end
rescue Aws::Lambda::Errors::ServiceException => e
  @logger.error("There was an error creating role or attaching policy:\n
#{e.message}")
end
```

Defina um manipulador do Lambda que incremente um número fornecido como um parâmetro de invocação.

```
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
```

Compacte a função do Lambda em um pacote de implantação.

```
# Creates a Lambda deployment package in .zip format.
```

```

# This zip can be passed directly as a string to Lambda when creating the
function.
#
# @param source_file: The name of the object, without suffix, for the Lambda file
and zip.
# @return: The deployment package.
def create_deployment_package(source_file)
  Dir.chdir(File.dirname(__FILE__))
  if File.exist?("lambda_function.zip")
    File.delete("lambda_function.zip")
    @logger.debug("Deleting old zip: lambda_function.zip")
  end
  Zip::File.open("lambda_function.zip", create: true) {
    |zipfile|
    zipfile.add("lambda_function.rb", "#{source_file}.rb")
  }
  @logger.debug("Zipping #{source_file}.rb into: lambda_function.zip.")
  File.read("lambda_function.zip").to_s
rescue StandardError => e
  @logger.error("There was an error creating deployment package:\n #{e.message}")
end

```

## Crie uma nova função do Lambda.

```

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

```

```

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

```

Invoke a função do Lambda com parâmetros de runtime opcionais.

```

# 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

```

Atualize a configuração da função do Lambda para injetar uma nova variável de 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
end

```

Atualize o código da função do Lambda com um pacote de implantação diferente que contenha um código diferente.

```

# Updates the code for a Lambda function by submitting a .zip archive that
contains
# the code for the function.

# @param function_name: The name of the function to update.
# @param deployment_package: The function code to update, packaged as bytes in
#                             .zip format.
# @return: Data about the update, including the status.
def update_function_code(function_name, deployment_package)
  @lambda_client.update_function_code(
    function_name: function_name,
    zip_file: deployment_package
  )
  @lambda_client.wait_until(:function_updated_v2, { function_name: function_name})
do |w|
  w.max_attempts = 5
  w.delay = 5
end
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error updating function code for: #{function_name}:
\n #{e.message}")
  end
end

```

```
nil
rescue Aws::Waiters::Errors::WaiterFailed => e
  @logger.error("Failed waiting for #{function_name} to update:\n #{e.message}")
end
```

Liste todas as funções do Lambda existentes usando o paginador integrado.

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

Exclua uma função do Lambda específica.

```
# Deletes a Lambda function.
# @param function_name: The name of the function to delete.
def delete_function(function_name)
  print "Deleting function: #{function_name}..."
  @lambda_client.delete_function(
    function_name: function_name
  )
  print "Done!".green
rescue Aws::Lambda::Errors::ServiceException => e
  @logger.error("There was an error deleting #{function_name}:\n #{e.message}")
end
```

- Para API obter detalhes, consulte os tópicos a seguir em AWS SDK for Ruby API Referência.
  - [CreateFunction](#)
  - [DeleteFunction](#)
  - [GetFunction](#)

- [Invoke](#)
- [ListFunctions](#)
- [UpdateFunctionCode](#)
- [UpdateFunctionConfiguration](#)

## Exemplos sem servidor

Conectando-se a um RDS banco de dados da Amazon em uma função Lambda

O exemplo de código a seguir mostra como implementar uma função Lambda que se conecta a um RDS banco de dados. A função faz uma solicitação simples ao banco de dados e exibe o resultado.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Conectando-se a um RDS banco de dados da Amazon em uma função Lambda usando Ruby.

```
# Ruby code here.

require 'aws-sdk-rds'
require 'json'
require 'mysql2'

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

  credentials = Aws::Credentials.new(
    ENV['AWS_ACCESS_KEY_ID'],
    ENV['AWS_SECRET_ACCESS_KEY'],
    ENV['AWS_SESSION_TOKEN']
  )
```

```
rds_client = Aws::RDS::AuthTokenGenerator.new(
  region: region,
  credentials: credentials
)

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

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

## Invocar uma função do Lambda em um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo recebimento de mensagens de um stream do Kinesis. A função recupera a carga útil do Kinesis, decodifica do Base64 e registra o conteúdo do registro em log.

## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Consumir um evento do Kinesis com o Lambda usando 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
```

## Invocar uma função do Lambda em um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma função Lambda que recebe um evento acionado pelo recebimento de registros de um stream do DynamoDB. A função recupera a carga útil do DynamoDB e registra em log o conteúdo do registro.

### SDKpara Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Como consumir um evento do DynamoDB com o Lambda usando 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
```

## Invocar uma função do Lambda de um acionador do Amazon DocumentDB

O exemplo de código a seguir mostra como implementar uma função Lambda que recebe um evento acionado pelo recebimento de registros de um stream de alterações do DocumentDB. A função recupera a carga útil do DocumentDB e registra em log o conteúdo do registro.

## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Consumir um evento do Amazon DocumentDB com o Lambda usando Ruby.

```
require 'json'

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


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

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

Invocar uma função Lambda a partir de um gatilho da Amazon MSK

O exemplo de código a seguir mostra como implementar uma função Lambda que recebe um evento acionado pelo recebimento de registros de um cluster da AmazonMSK. A função recupera a MSK carga e registra o conteúdo do registro.

## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Consumindo um MSK evento da Amazon com Lambda usando 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
```

Invocar uma função do Lambda em um acionador do Amazon S3

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo upload de um objeto para um bucket do S3. A função recupera o nome do bucket do S3 e a chave do objeto do parâmetro de evento e chama o Amazon API S3 para recuperar e registrar o tipo de conteúdo do objeto.



## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Como consumir um evento do S3 com o Lambda usando Ruby.

```
require 'json'
require 'uri'
require 'aws-sdk'

puts 'Loading function'

def lambda_handler(event:, context:)
  s3 = Aws::S3::Client.new(region: 'region') # Your AWS region
  # puts "Received event: #{JSON.dump(event)}"

  # Get the object from the event and show its content type
  bucket = event['Records'][0]['s3']['bucket']['name']
  key = URI.decode_www_form_component(event['Records'][0]['s3']['object']['key'],
  Encoding::UTF_8)
  begin
    response = s3.get_object(bucket: bucket, key: key)
    puts "CONTENT TYPE: #{response.content_type}"
    return response.content_type
  rescue StandardError => e
    puts e.message
    puts "Error getting object #{key} from bucket #{bucket}. Make sure they exist
    and your bucket is in the same region as this function."
    raise e
  end
end
```

## Invocar uma função Lambda a partir de um gatilho da Amazon SNS

O exemplo de código a seguir mostra como implementar uma função Lambda que recebe um evento acionado pelo recebimento de mensagens de um SNS tópico. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

### SDKpara Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Consumir um SNS evento com Lambda usando Ruby.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].map { |record| process_message(record) }
end

def process_message(record)
  message = record['Sns']['Message']
  puts("Processing message: #{message}")
rescue StandardError => e
  puts("Error processing message: #{e}")
  raise
end
```

## Invocar uma função Lambda a partir de um gatilho da Amazon SQS

O exemplo de código a seguir mostra como implementar uma função Lambda que recebe um evento acionado pelo recebimento de mensagens de uma SQS fila. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Consumir um SQS evento com Lambda usando 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
```

Relatando falhas de itens em lote para funções do Lambda com um trigger do Kinesis

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um stream do Kinesis. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Relatar falhas de item em lote do Kinesis com o Lambda usando 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
```

## Relatar falhas de itens em lote para funções do Lambda com um gatilho do DynamoDB

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções do Lambda que recebem eventos de um stream do DynamoDB. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

### SDKpara Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

## Como relatar falhas de itens em lote do DynamoDB com o Lambda usando 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
```

## Relatando falhas de itens em lote para funções Lambda com um gatilho da Amazon SQS

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções Lambda que recebem eventos de uma SQS fila. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

### SDKpara Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Relatar falhas SQS de itens em lote com o Lambda usando 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
```

# MSK Exemplos da Amazon usando SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Ruby com a AmazonMSK.

Cada exemplo inclui um link para o código-fonte completo, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

## Tópicos

- [Exemplos sem servidor](#)

## Exemplos sem servidor

Invocar uma função Lambda a partir de um gatilho da Amazon MSK

O exemplo de código a seguir mostra como implementar uma função Lambda que recebe um evento acionado pelo recebimento de registros de um cluster da AmazonMSK. A função recupera a MSK carga e registra o conteúdo do registro.

## SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Consumindo um MSK evento da Amazon com Lambda usando 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
```

## Exemplos de uso SDK do Amazon Polly para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Ruby com o Amazon Polly.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Os cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)
- [Cenários](#)

## Ações

### **DescribeVoices**

O código de exemplo a seguir mostra como usar DescribeVoices.

### SDK para Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).



```
require "aws-sdk-polly" # In v2: require 'aws-sdk'

begin
  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  # Get US English voices
  resp = polly.describe_voices(language_code: "en-US")

  resp.voices.each do |v|
    puts v.name
    puts "  " + v.gender
    puts
  end
rescue StandardError => ex
  puts "Could not get voices"
  puts "Error message:"
  puts ex.message
end
```

- Para API obter detalhes, consulte [DescribeVoices](#) em AWS SDK for Ruby API Referência.

## ListLexicons

O código de exemplo a seguir mostra como usar ListLexicons.

### SDK para Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
require "aws-sdk-polly" # In v2: require 'aws-sdk'
```

```
begin
  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  resp = polly.list_lexicons

  resp.lexicons.each do |l|
    puts l.name
    puts "  Alphabet:" + l.attributes.alphabet
    puts "  Language:" + l.attributes.language
    puts
  end
rescue StandardError => ex
  puts "Could not get lexicons"
  puts "Error message:"
  puts ex.message
end
```

- Para API obter detalhes, consulte [ListLexicons](#) em AWS SDK for Ruby API Referência.

## SynthesizeSpeech

O código de exemplo a seguir mostra como usar SynthesizeSpeech.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
require "aws-sdk-polly" # In v2: require 'aws-sdk'

begin
  # Get the filename from the command line
  if ARGV.empty?
    puts "You must supply a filename"
```

```
    exit 1
  end

  filename = ARGV[0]

  # Open file and get the contents as a string
  if File.exist?(filename)
    contents = IO.read(filename)
  else
    puts "No such file: " + filename
    exit 1
  end

  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  resp = polly.synthesize_speech({
    output_format: "mp3",
    text: contents,
    voice_id: "Joanna",
  })

  # Save output
  # Get just the file name
  # abc/xyz.txt -> xyx.txt
  name = File.basename(filename)

  # Split up name so we get just the xyz part
  parts = name.split(".")
  first_part = parts[0]
  mp3_file = first_part + ".mp3"

  IO.copy_stream(resp.audio_stream, mp3_file)

  puts "Wrote MP3 content to: " + mp3_file
  rescue StandardError => ex
    puts "Got error:"
    puts "Error message:"
    puts ex.message
  end
end
```

- Para API obter detalhes, consulte [SynthesizeSpeech](#) em AWS SDK for Ruby API Referência.

## Cenários

Criar uma aplicação para analisar o feedback dos clientes

O exemplo de código a seguir mostra como criar uma aplicação que analisa os cartões de comentários dos clientes, os traduz do idioma original, determina seus sentimentos e gera um arquivo de áudio do texto traduzido.

### SDK para Ruby

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter o código-fonte e as instruções de implantação, consulte o projeto em [GitHub](#).

Serviços utilizados neste exemplo

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

## RDS Exemplos da Amazon usando SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Ruby com a Amazon RDS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.


Cada exemplo inclui um link para o código-fonte completo, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Conceitos básicos

Olá Amazon RDS

Os exemplos de código a seguir mostram como começar a usar a AmazonRDS.

SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
  end
end
```

```
end

if instances.empty?
  @logger.info('No instances found.')
else
  @logger.info("Found #{instances.count} instance(s):")
  instances.each do |instance|
    @logger.info(" * #{instance.db_instance_identifier}
  (#{instance.db_instance_status})")
  end
end
end
end

if $PROGRAM_NAME == __FILE__
  rds_client = Aws::RDS::Client.new(region: 'us-west-2')
  manager = RDSManager.new(rds_client)
  manager.list_db_instances
end
```

- Para API obter detalhes, consulte [Descreva as instâncias de banco de dados](#) em AWS SDK for Ruby API Referência.

## Tópicos

- [Ações](#)
- [Exemplos sem servidor](#)

## Ações

### CreateDBSnapshot

O código de exemplo a seguir mostra como usar CreateDBSnapshot.

## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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


# Create a snapshot for an Amazon Relational Database Service (Amazon RDS)
# DB instance.
#
# @param rds_resource [Aws::RDS::Resource] The resource containing SDK logic.
# @param db_instance_name [String] The name of the Amazon RDS DB instance.
# @return [Aws::RDS::DBSnapshot, nil] The snapshot created, or nil if error.
def create_snapshot(rds_resource, db_instance_name)
  id = "snapshot-#{rand(10**6)}"
  db_instance = rds_resource.db_instance(db_instance_name)
  db_instance.create_snapshot({
    db_snapshot_identifier: id
  })
rescue Aws::Errors::ServiceError => e
  puts "Couldn't create DB instance snapshot #{id}:\n #{e.message}"
end
```

- Para API obter detalhes, consulte [C reateDBSnapshot](#) em AWS SDK for Ruby APIReferência.

## DescribeDBInstances

O código de exemplo a seguir mostra como usar DescribeDBInstances.

## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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

# List all Amazon Relational Database Service (Amazon RDS) DB instances.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return [Array, nil] List of all DB instances, or nil if error.
def list_instances(rds_resource)
  db_instances = []
  rds_resource.db_instances.each do |i|
    db_instances.append({
      "name": i.id,
      "status": i.db_instance_status
    })
  end
  db_instances
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list instances:\n#{e.message}"
end
```

- Para API obter detalhes, consulte [DescrevaDBInstances](#) em AWS SDK for Ruby APIReferência.

## DescribeDBParameterGroups

O código de exemplo a seguir mostra como usar DescribeDBParameterGroups.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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

# List all Amazon Relational Database Service (Amazon RDS) parameter groups.
#
# @param rds_resource [Aws::RDS::Resource] An SDK for Ruby Amazon RDS resource.
# @return [Array, nil] List of all parameter groups, or nil if error.
```



```
def list_parameter_groups(rds_resource)
  parameter_groups = []
  rds_resource.db_parameter_groups.each do |p|
    parameter_groups.append({
      "name": p.db_parameter_group_name,
      "description": p.description
    })
  end
  parameter_groups
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list parameter groups:\n #{e.message}"
end
```

- Para API obter detalhes, consulte [DescrevaDBParameter Groups](#) in AWS SDK for Ruby APIReference.

## DescribeDBParameters

O código de exemplo a seguir mostra como usar DescribeDBParameters.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
end
```

```

    })
  end
  parameter_groups
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list parameter groups:\n #{e.message}"
end

```

- Para API obter detalhes, consulte [DescrevaDBParameters](#) em AWS SDK for Ruby API Referência.

## DescribeDBSnapshots

O código de exemplo a seguir mostra como usar DescribeDBSnapshots.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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

```

```
end
```

- Para API obter detalhes, consulte [DescreveDBSnapshots](#) em AWS SDK for Ruby APIReferência.

## Exemplos sem servidor

Conectando-se a um RDS banco de dados da Amazon em uma função Lambda

O exemplo de código a seguir mostra como implementar uma função Lambda que se conecta a um RDS banco de dados. A função faz uma solicitação simples ao banco de dados e exibe o resultado.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Conectando-se a um RDS banco de dados da Amazon em uma função Lambda usando 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
```

## Exemplos do Amazon S3 usando SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Ruby com o Amazon S3.

As noções básicas são exemplos de código que mostram como realizar as operações essenciais em um serviço.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Os cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

## Conceitos básicos

### Olá, Amazon S3

O exemplo de código a seguir mostra como começar a usar o Amazon S3.

## SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
# frozen_string_literal: true

# S3Manager is a class responsible for managing S3 operations
# such as listing all S3 buckets in the current AWS account.
class S3Manager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all S3 buckets in the current AWS account.
  def list_buckets
    @logger.info('Here are the buckets in your account:')
  end
end
```

```
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

if $PROGRAM_NAME == __FILE__
  s3_client = Aws::S3::Client.new
  manager = S3Manager.new(s3_client)
  manager.list_buckets
end
```

- Para API obter detalhes, consulte [ListBuckets](#) em AWS SDK for Ruby API Referência.

## Tópicos

- [Conceitos básicos](#)
- [Ações](#)
- [Cenários](#)
- [Exemplos sem servidor](#)

## Conceitos básicos

Aprender os conceitos básicos

O exemplo de código a seguir mostra como:

- Criar um bucket e fazer upload de um arquivo para ele.
- Baixar um objeto de um bucket.
- Copiar um objeto em uma subpasta em um bucket.

- Listar os objetos em um bucket.
- Excluir os objetos do bucket e o bucket.

## SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
require "aws-sdk-s3"

# Wraps the getting started scenario actions.
class ScenarioGettingStarted
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end

  # Creates a bucket with a random name in the currently configured account and
  # AWS Region.
  #
  # @return [Aws::S3::Bucket] The newly created bucket.
  def create_bucket
    bucket = @s3_resource.create_bucket(
      bucket: "doc-example-bucket-#{Random.uuid}",
      create_bucket_configuration: {
        location_constraint: "us-east-1" # Note: only certain regions permitted
      }
    )
    puts("Created demo bucket named #{bucket.name}.")
  rescue Aws::Errors::ServiceError => e
    puts("Tried and failed to create demo bucket.")
    puts("\t#{e.code}: #{e.message}")
    puts("\nCan't continue the demo without a bucket!")
    raise
  else
    bucket
  end
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

# 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!
      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__
end
```

- Para API obter detalhes, consulte os tópicos a seguir em AWS SDK for Ruby API Referência.
  - [CopyObject](#)
  - [CreateBucket](#)
  - [DeleteBucket](#)
  - [DeleteObjects](#)
  - [GetObject](#)

- [ListObjectsV2](#)
- [PutObject](#)

## Ações

### CopyObject

O código de exemplo a seguir mostra como usar CopyObject.

SDKpara Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Copie um objeto.

```
require "aws-sdk-s3"

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

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

  # Copy the source object to the specified target bucket and rename it with the
  # target key.
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
  # object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
  # nil.
  def copy_object(target_bucket, target_object_key)
```

```

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

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

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

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

run_demo if $PROGRAM_NAME == __FILE__

```

Copie um objeto e adicione criptografia do lado do servidor ao objeto de destino.

```

require "aws-sdk-s3"

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

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

```

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

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

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

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

run_demo if $PROGRAM_NAME == __FILE__
```

- Para API obter detalhes, consulte [CopyObject](#) em AWS SDK for Ruby API Referência.

## CreateBucket

O código de exemplo a seguir mostra como usar CreateBucket.

### SDKpara Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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("doc-example-bucket-
#{Random.uuid}"))
  return unless wrapper.create?(region)

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

run_demo if $PROGRAM_NAME == __FILE__

```

- Para API obter detalhes, consulte [CreateBucket](#) em AWS SDK for Ruby API Referência.

## DeleteBucket

O código de exemplo a seguir mostra como usar DeleteBucket.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```

# Deletes the objects in an Amazon S3 bucket and deletes the bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
def delete_bucket(bucket)
  puts("\nDo you want to delete all of the objects as well as the bucket (y/n)? ")

```

```

    answer = gets.chomp.downcase
    if answer == "y"
      bucket.objects.batch_delete!
      bucket.delete
      puts("Emptied and deleted bucket #{bucket.name}.\n")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't empty and delete bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
end

```

- Para API obter detalhes, consulte [DeleteBucket](#) em AWS SDK for Ruby API Referência.

## DeleteBucketCors

O código de exemplo a seguir mostra como usar DeleteBucketCors.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```

require "aws-sdk-s3"

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with an
  # existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Deletes the CORS configuration of a bucket.
  #

```



```

# @return [Boolean] True if the CORS rules were deleted; otherwise, false.
def delete_cors
  @bucket_cors.delete
  true
rescue Aws::Errors::ServiceError => e
  puts "Couldn't delete CORS rules for #{@bucket_cors.bucket.name}. Here's why:
#{e.message}"
  false
end
end

```

- Para API obter detalhes, consulte [DeleteBucketCors](#) em AWS SDK for Ruby API Referência.

## DeleteBucketPolicy

O código de exemplo a seguir mostra como usar DeleteBucketPolicy.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```

# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object configured
  with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  def delete_policy
    @bucket_policy.delete
    true
  rescue Aws::Errors::ServiceError => e

```

```

    puts "Couldn't delete the policy from #{@bucket_policy.bucket.name}. Here's why:
#{e.message}"
    false
  end
end
end

```

- Para API obter detalhes, consulte [DeleteBucketPolicy](#) em AWS SDK for Ruby API Referência.

## DeleteObjects

O código de exemplo a seguir mostra como usar DeleteObjects.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```

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

```

- Para API obter detalhes, consulte [DeleteObjects](#) em AWS SDK for Ruby API Referência.

## GetBucketCors

O código de exemplo a seguir mostra como usar GetBucketCors.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
require "aws-sdk-s3"

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with an
  # existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Gets the CORS configuration of a bucket.
  #
  # @return [Aws::S3::Type::GetBucketCorsOutput, nil] The current CORS configuration
  # for the bucket.
  def get_cors
    @bucket_cors.data
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get CORS configuration for #{@bucket_cors.bucket.name}. Here's
  why: #{e.message}"
    nil
  end
end
```

- Para API obter detalhes, consulte [GetBucketCorsem AWS SDK for Ruby APIReferência](#).

## GetBucketPolicy

O código de exemplo a seguir mostra como usar `GetBucketPolicy`.

### SDK para Ruby

#### Note

Tem mais sobre `GetBucketPolicy` no GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object configured
  # with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Gets the policy of a bucket.
  #
  # @return [Aws::S3::GetBucketPolicyOutput, nil] The current bucket policy.
  def get_policy
    policy = @bucket_policy.data.policy
    policy.respond_to?(:read) ? policy.read : policy
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get the policy for #{@bucket_policy.bucket.name}. Here's why:
    #{e.message}"
    nil
  end
end
```

- Para API obter detalhes, consulte [GetBucketPolicy](#) em AWS SDK for Ruby API Referência.

## GetObject

O código de exemplo a seguir mostra como usar `GetObject`.

## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Obtenha um objeto.

```
require "aws-sdk-s3"

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

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

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

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object.txt"
  target_path = "my-object-as-file.txt"

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

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

run_demo if $PROGRAM_NAME == __FILE__
```

Obtenha um objeto e relate seu estado de criptografia do lado do servidor.

```
require "aws-sdk-s3"

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

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

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

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object.txt"

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

  encryption = obj_data.server_side_encryption.nil? ? "no" :
  obj_data.server_side_encryption
end
```

```
puts "Object #{object_key} uses #{encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- Para API obter detalhes, consulte [GetObject](#) em AWS SDK for Ruby API Referência.

## HeadObject

O código de exemplo a seguir mostra como usar `HeadObject`.

### SDK para Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
require "aws-sdk-s3"

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

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

  # Checks whether the object exists.
  #
  # @return [Boolean] True if the object exists; otherwise false.
  def exists?
    @object.exists?
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't check existence of object #{@object.bucket.name}:#{@object.key}.
    Here's why: #{e.message}"
    false
  end
end
```

```
# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object.txt"

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

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

run_demo if $PROGRAM_NAME == __FILE__
```

- Para API obter detalhes, consulte [HeadObject](#) em AWS SDK for Ruby API Referência.

## ListBuckets

O código de exemplo a seguir mostra como usar ListBuckets.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
require "aws-sdk-s3"

# Wraps Amazon S3 resource actions.
class BucketListWrapper
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end

  # Lists buckets for the current account.
  #
```



```
# @param count [Integer] The maximum number of buckets to list.
def list_buckets(count)
  puts "Found these buckets:"
  @s3_resource.buckets.each do |bucket|
    puts "\t#{bucket.name}"
    count -= 1
    break if count.zero?
  end
  true
rescue Aws::Errors::ServiceError => e
  puts "Couldn't list buckets. Here's why: #{e.message}"
  false
end
end

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

run_demo if $PROGRAM_NAME == __FILE__
```

- Para API obter detalhes, consulte [ListBuckets](#) em AWS SDK for Ruby API Referência.

## ListObjectsV2

O código de exemplo a seguir mostra como usar ListObjectsV2.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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 = "doc-example-bucket"

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


run_demo if $PROGRAM_NAME == __FILE__
```

- Para API obter detalhes, consulte [ListObjectsV2](#) em AWS SDK for Ruby API Referência.

## PutBucketCors

O código de exemplo a seguir mostra como usar PutBucketCors.

## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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
```

- Para API obter detalhes, consulte [PutBucketCors](#) em AWS SDK for Ruby APIReferência.

## PutBucketPolicy

O código de exemplo a seguir mostra como usar `PutBucketPolicy`.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object configured
  # with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Sets a policy on a bucket.
  #
  def set_policy(policy)
    @bucket_policy.put(policy: policy)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't set the policy for #{@bucket_policy.bucket.name}. Here's why:
#{e.message}"
    false
  end
end
```

- Para API obter detalhes, consulte [PutBucketPolicy](#) em AWS SDK for Ruby APIReferência.

## PutBucketWebsite

O código de exemplo a seguir mostra como usar PutBucketWebsite.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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 = "doc-example-bucket"
  index_document = "index.html"
  error_document = "404.html"

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

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

run_demo if $PROGRAM_NAME == __FILE__
```

- Para API obter detalhes, consulte [PutBucketWebsite](#) em AWS SDK for Ruby API Referência.

## PutObject

O código de exemplo a seguir mostra como usar PutObject.

### SDKpara Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Carregue um arquivo usando um carregador gerenciado (Object.upload\_file).

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
```

```
class ObjectUploadFileWrapper
  attr_reader :object

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

  # Uploads a file to an Amazon S3 object by using a managed uploader.
  #
  # @param file_path [String] The path to the file to upload.
  # @return [Boolean] True when the file is uploaded; otherwise false.
  def upload_file(file_path)
    @object.upload_file(file_path)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't upload file #{file_path} to #{@object.key}. Here's why:
#{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-uploaded-file"
  file_path = "object_upload_file.rb"

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

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

run_demo if $PROGRAM_NAME == __FILE__
```

Carregue um arquivo usando Object.put.

```
require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
```

```

class ObjectPutWrapper
  attr_reader :object

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

  def put_object(source_file_path)
    File.open(source_file_path, "rb") do |file|
      @object.put(body: file)
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put #{source_file_path} to #{object.key}. Here's why:
#{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object-key"
  file_path = "my-local-file.txt"

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

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

run_demo if $PROGRAM_NAME == __FILE__

```

Carregue um arquivo usando `Object.put` e adicione criptografia do lado do servidor.

```

require "aws-sdk-s3"

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

```



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

def put_object_encrypted(object_content, encryption)
  @object.put(body: object_content, server_side_encryption: encryption)
  true
rescue Aws::Errors::ServiceError => e
  puts "Couldn't put your content to #{object.key}. Here's why: #{e.message}"
  false
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-encrypted-content"
  object_content = "This is my super-secret content."
  encryption = "AES256"

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

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

run_demo if $PROGRAM_NAME == __FILE__
```


- Para API obter detalhes, consulte [PutObject](#) em AWS SDK for Ruby API Referência.

## Cenários

### Crie um pré-assinado URL

O exemplo de código a seguir mostra como criar um pré-assinado URL para o Amazon S3 e fazer o upload de um objeto.

## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
require "aws-sdk-s3"
require "net/http"

# Creates a presigned URL that can be used to upload content to an object.
#
# @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
# @param object_key [String] The key to give the uploaded object.
# @return [URI, nil] The parsed URI if successful; otherwise nil.
def get_presigned_url(bucket, object_key)
  url = bucket.object(object_key).presigned_url(:put)
  puts "Created presigned URL: #{url}"
  URI(url)
rescue Aws::Errors::ServiceError => e
  puts "Couldn't create presigned URL for #{bucket.name}:#{object_key}. Here's why:
  #{e.message}"
end

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

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

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

  case response
  when Net::HTTPSuccess
```

```
    puts "Content uploaded!"
  else
    puts response.value
  end
end
end

run_demo if $PROGRAM_NAME == __FILE__
```

## Exemplos sem servidor

### Invocar uma função do Lambda em um acionador do Amazon S3

O exemplo de código a seguir mostra como implementar uma função do Lambda que recebe um evento acionado pelo upload de um objeto para um bucket do S3. A função recupera o nome do bucket do S3 e a chave do objeto do parâmetro de evento e chama o Amazon API S3 para recuperar e registrar o tipo de conteúdo do objeto.

### SDKpara Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Como consumir um evento do S3 com o Lambda usando 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
```

## SESExemplos da Amazon usando SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Ruby com a AmazonSES.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos


- [Ações](#)

## Ações

### **GetIdentityVerificationAttributes**

O código de exemplo a seguir mostra como usar `GetIdentityVerificationAttributes`.

## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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

# Create client in us-west-2 region
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
client = Aws::SES::Client.new(region: "us-west-2")

# Get up to 1000 identities
ids = client.list_identities({
  identity_type: "EmailAddress"
})

ids.identities.each do |email|
  attrs = client.get_identity_verification_attributes({
    identities: [email]
  })

  status = attrs.verification_attributes[email].verification_status


  # Display email addresses that have been verified
  if status == "Success"
    puts email
  end
end
```

- Para API obter detalhes, consulte [GetIdentityVerificationAttributes](#) em AWS SDK for Ruby APIReferência.

## ListIdentities

O código de exemplo a seguir mostra como usar `ListIdentities`.

## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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

# Create client in us-west-2 region
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
client = Aws::SES::Client.new(region: "us-west-2")

# Get up to 1000 identities
ids = client.list_identities({
  identity_type: "EmailAddress"
})

ids.identities.each do |email|
  attrs = client.get_identity_verification_attributes({
    identities: [email]
  })

  status = attrs.verification_attributes[email].verification_status


  # Display email addresses that have been verified
  if status == "Success"
    puts email
  end
end
```

- Para API obter detalhes, consulte [ListIdentities](#) em AWS SDK for Ruby API Referência.

## SendEmail

O código de exemplo a seguir mostra como usar SendEmail.

## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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

# Replace sender@example.com with your "From" address.
# This address must be verified with Amazon SES.
sender = "sender@example.com"

# Replace recipient@example.com with a "To" address. If your account
# is still in the sandbox, this address must be verified.
recipient = "recipient@example.com"

# Specify a configuration set. To use a configuration
# set, uncomment the next line and line 74.
# configsetname = "ConfigSet"

# The subject line for the email.
subject = "Amazon SES test (AWS SDK for Ruby)"

# The HTML body of the email.
htmlbody =
  "<h1>Amazon SES test (AWS SDK for Ruby)</h1>\"
  '<p>This email was sent with <a href="https://aws.amazon.com/ses/">\'
  'Amazon SES</a> using the <a href="https://aws.amazon.com/sdk-for-ruby/">\'
  "AWS SDK for Ruby</a>."

# The email body for recipients with non-HTML email clients.
textbody = "This email was sent with Amazon SES using the AWS SDK for Ruby."

# Specify the text encoding scheme.
encoding = "UTF-8"

# Create a new SES client in the us-west-2 region.
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
ses = Aws::SES::Client.new(region: "us-west-2")
```

```
# Try to send the email.
begin
  # Provide the contents of the email.
  ses.send_email(
    destination: {
      to_addresses: [
        recipient
      ]
    },
    message: {
      body: {
        html: {
          charset: encoding,
          data: htmlbody
        },
        text: {
          charset: encoding,
          data: textbody
        }
      },
      subject: {
        charset: encoding,
        data: subject
      }
    },
    source: sender,
    # Uncomment the following line to use a configuration set.
    # configuration_set_name: configsetname,
  )

  puts "Email sent to " + recipient

# If something goes wrong, display an error message.
rescue Aws::SES::Errors::ServiceError => error
  puts "Email not sent. Error message: #{error}"
end
```

- Para API obter detalhes, consulte [SendEmail](#) em AWS SDK for Ruby API Referência.



## VerifyEmailIdentity

O código de exemplo a seguir mostra como usar VerifyEmailIdentity.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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

# Replace recipient@example.com with a "To" address.
recipient = "recipient@example.com"

# Create a new SES resource in the us-west-2 region.
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
ses = Aws::SES::Client.new(region: "us-west-2")

# Try to verify email address.
begin
  ses.verify_email_identity({
    email_address: recipient
  })

  puts "Email sent to " + recipient

# If something goes wrong, display an error message.
rescue Aws::SES::Errors::ServiceError => error
  puts "Email not sent. Error message: #{error}"
end
```

- Para API obter detalhes, consulte [VerifyEmailIdentity](#) em AWS SDK for Ruby API Referência.

# Exemplos da Amazon SES API v2 usando SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Ruby com a Amazon SES API v2.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

## Ações

### SendEmail

O código de exemplo a seguir mostra como usar SendEmail.

SDKpara Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
require "aws-sdk-sesv2"
require_relative "config" # Recipient and sender email addresses.

# Set up the SESv2 client.
client = Aws::SESV2::Client.new(region: AWS_REGION)

def send_email(client, sender_email, recipient_email)
  response = client.send_email(
    {
      from_email_address: sender_email,
```

```
destination: {
  to_addresses: [recipient_email]
},
content: {
  simple: {
    subject: {
      data: "Test email subject"
    },
    body: {
      text: {
        data: "Test email body"
      }
    }
  }
}
)
puts "Email sent from #{SENDER_EMAIL} to #{RECIPIENT_EMAIL} with message ID:
#{response.message_id}"
end

send_email(client, SENDER_EMAIL, RECIPIENT_EMAIL)
```

- Para API obter detalhes, consulte [SendEmail](#) em AWS SDK for Ruby API Referência.

## SNSExemplos da Amazon usando SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Ruby com a AmazonSNS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Ações](#)
- [Exemplos sem servidor](#)

# Ações

## CreateTopic

O código de exemplo a seguir mostra como usar `CreateTopic`.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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 = SNSTopicCreator.new

puts "Creating the topic '#{topic_name}'..."
unless sns_topic_creator.create_topic(topic_name)
  puts "The topic was not created. Stopping program."
  exit 1
end
end
```

- Para obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK for Ruby](#).
- Para API obter detalhes, consulte [CreateTopic](#) em AWS SDK for Ruby APIReferência.

## ListSubscriptions

O código de exemplo a seguir mostra como usar ListSubscriptions.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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

# Example usage:
if $PROGRAM_NAME == __FILE__
  sns_client = Aws::SNS::Client.new
  topic_arn = "SNS_TOPIC_ARN" # Replace with your SNS topic ARN
  lister = SnsSubscriptionLister.new(sns_client)

  begin
    lister.list_subscriptions(topic_arn)
  rescue StandardError => e
    puts "Failed to list subscriptions: #{e.message}"
    exit 1
  end
end
```

- Para obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK for Ruby](#).
- Para API obter detalhes, consulte [ListSubscriptions](#) em AWS SDK for Ruby API Referência.

## ListTopics

O código de exemplo a seguir mostra como usar ListTopics.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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

```
def list_topics?(sns_client)
  sns_client.topics.each do |topic|
    puts topic.arn
  rescue StandardError => e
    puts "Error while listing the topics: #{e.message}"
  end
end

def run_me

  region = "REGION"
  sns_client = Aws::SNS::Resource.new(region: region)

  puts "Listing the topics."

  if list_topics?(sns_client)
  else
    puts "The bucket was not created. Stopping program."
    exit 1
  end
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Para obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK for Ruby](#).
- Para API obter detalhes, consulte [ListTopics](#) em AWS SDK for Ruby API Referência.

## Publish

O código de exemplo a seguir mostra como usar Publish.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
# Service class for sending messages using Amazon Simple Notification Service (SNS)
class SnsMessageSender
  # Initializes the SnsMessageSender with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Sends a message to a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param message [String] The message to send
  # @return [Boolean] true if message was successfully sent, false otherwise
  def send_message(topic_arn, message)
    @sns_client.publish(topic_arn: topic_arn, message: message)
    @logger.info("Message sent successfully to #{topic_arn}.")
    true
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error while sending the message: #{e.message}")
    false
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = "SNS_TOPIC_ARN" # Should be replaced with a real topic ARN
  message = "MESSAGE"        # Should be replaced with the actual message content

  sns_client = Aws::SNS::Client.new
  message_sender = SnsMessageSender.new(sns_client)

  @logger.info("Sending message.")
  unless message_sender.send_message(topic_arn, message)
    @logger.error("Message sending failed. Stopping program.")
    exit 1
  end
end
```

- Para obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK for Ruby](#).
- Para API obter detalhes, consulte [Publicar](#) em AWS SDK for Ruby API referência.



## SetTopicAttributes

O código de exemplo a seguir mostra como usar SetTopicAttributes.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
# Service class to enable an SNS resource with a specified policy
class SnsResourceEnabler
  # Initializes the SnsResourceEnabler with an SNS resource client
  #
  # @param sns_resource [Aws::SNS::Resource] The SNS resource client
  def initialize(sns_resource)
    @sns_resource = sns_resource
    @logger = Logger.new($stdout)
  end

  # Sets a policy on a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource to include in the policy
  # @param policy_name [String] The name of the policy attribute to set
  def enable_resource(topic_arn, resource_arn, policy_name)
    policy = generate_policy(topic_arn, resource_arn)
    topic = @sns_resource.topic(topic_arn)

    topic.set_attributes({
      attribute_name: policy_name,
      attribute_value: policy
    })
    @logger.info("Policy #{policy_name} set successfully for topic #{topic_arn}.")
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Failed to set policy: #{e.message}")
  end

  private

  # Generates a policy string with dynamic resource ARNs
```

```

#
# @param topic_arn [String] The ARN of the SNS topic
# @param resource_arn [String] The ARN of the resource
# @return [String] The policy as a JSON string
def generate_policy(topic_arn, resource_arn)
  {
    Version: "2008-10-17",
    Id: "__default_policy_ID",
    Statement: [{
      Sid: "__default_statement_ID",
      Effect: "Allow",
      Principal: { "AWS": "*" },
      Action: ["SNS:Publish"],
      Resource: topic_arn,
      Condition: {
        ArnEquals: {
          "AWS:SourceArn": resource_arn
        }
      }
    }]
  }.to_json
end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = "MY_TOPIC_ARN" # Should be replaced with a real topic ARN
  resource_arn = "MY_RESOURCE_ARN" # Should be replaced with a real resource ARN
  policy_name = "POLICY_NAME" # Typically, this is "Policy"

  sns_resource = Aws::SNS::Resource.new
  enabler = SnsResourceEnabler.new(sns_resource)

  enabler.enable_resource(topic_arn, resource_arn, policy_name)
end

```

- Para obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK for Ruby](#).
- Para API obter detalhes, consulte [SetTopicAttributes](#) em AWS SDK for Ruby API Referência.

## Subscribe

O código de exemplo a seguir mostra como usar Subscribe.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

Inscreva um endereço de e-mail em um tópico.

```
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
```

- Para obter mais informações, consulte o [Guia do desenvolvedor do AWS SDK for Ruby](#).
- Para API obter detalhes, consulte [Inscrever-se](#) na AWS SDK for Ruby API referência.

## Exemplos sem servidor

Invocar uma função Lambda a partir de um gatilho da Amazon SNS

O exemplo de código a seguir mostra como implementar uma função Lambda que recebe um evento acionado pelo recebimento de mensagens de um SNS tópico. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Consumir um SNS evento com Lambda usando 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
```

## SQSExemplos da Amazon usando SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Ruby com a AmazonSQS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos


- [Ações](#)
- [Exemplos sem servidor](#)

## Ações

### **ChangeMessageVisibility**

O código de exemplo a seguir mostra como usar `ChangeMessageVisibility`.

## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
require "aws-sdk-sqs" # v2: require 'aws-sdk'
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
sqs = Aws::SQS::Client.new(region: "us-west-2")

begin
  queue_name = "my-queue"
  queue_url = sqs.get_queue_url(queue_name: queue_name).queue_url

  receive_message_result_before = sqs.receive_message({
    queue_url: queue_url,
    max_number_of_messages: 10 # Receive up to 10 messages, if there are that many.
  })

  puts "Before attempting to change message visibility timeout: received
#{receive_message_result_before.messages.count} message(s)."

  receive_message_result_before.messages.each do |message|
    sqs.change_message_visibility({
      queue_url: queue_url,
      receipt_handle: message.receipt_handle,
      visibility_timeout: 30 # This message will not be visible for 30 seconds after
first receipt.
    })
  end

  # Try to retrieve the original messages after setting their visibility timeout.
  receive_message_result_after = sqs.receive_message({
    queue_url: queue_url,
    max_number_of_messages: 10
  })

  puts "\nAfter attempting to change message visibility timeout: received
#{receive_message_result_after.messages.count} message(s)."
```

```
rescue Aws::SQS::Errors::NonExistentQueue
  puts "Cannot receive messages for a queue named '#{receive_queue_name}', as it
  does not exist."
end
```

- Para API obter detalhes, consulte [ChangeMessageVisibility](#) em AWS SDK for Ruby APIReferência.

## CreateQueue

O código de exemplo a seguir mostra como usar CreateQueue.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
# This code example demonstrates how to create a queue in Amazon Simple Queue
Service (Amazon SQS).

require "aws-sdk-sqs"

# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_name [String] The name of the queue.
# @return [Boolean] true if the queue was created; otherwise, false.
# @example
#   exit 1 unless queue_created?(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'my-queue'
#   )
def queue_created?(sqs_client, queue_name)
  sqs_client.create_queue(queue_name: queue_name)
  true
rescue StandardError => e
  puts "Error creating queue: #{e.message}"
  false
```

```
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = "us-west-2"
  queue_name = "my-queue"
  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Creating the queue named '#{queue_name}'..."

  if queue_created?(sqs_client, queue_name)
    puts "Queue created."
  else
    puts "Queue not created."
  end
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Para API obter detalhes, consulte [CreateQueue](#) em AWS SDK for Ruby API Referência.

## DeleteQueue

O código de exemplo a seguir mostra como usar DeleteQueue.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
require "aws-sdk-sqs" # v2: require 'aws-sdk'
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
sqs = Aws::SQS::Client.new(region: "us-west-2")

sqs.delete_queue(queue_url: URL)
```



- Para API obter detalhes, consulte [DeleteQueue](#) em AWS SDK for Ruby APIReferência.

## ListQueues

O código de exemplo a seguir mostra como usar ListQueues.

### SDKpara Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
require "aws-sdk-sqs"
require "aws-sdk-sts"

# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @example
# list_queue_urls(Aws::SQS::Client.new(region: 'us-west-2'))
def list_queue_urls(sqs_client)
  queues = sqs_client.list_queues

  queues.queue_urls.each do |url|
    puts url
  end
rescue StandardError => e
  puts "Error listing queue URLs: #{e.message}"
end

# Lists the attributes of a queue in Amazon Simple Queue Service (Amazon SQS).
#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @example
# list_queue_attributes(
#   Aws::SQS::Client.new(region: 'us-west-2'),
#   'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
# )
```

```
def list_queue_attributes(sqs_client, queue_url)
  attributes = sqs_client.get_queue_attributes(
    queue_url: queue_url,
    attribute_names: ["All"]
  )

  attributes.attributes.each do |key, value|
    puts "#{key}: #{value}"
  end

rescue StandardError => e
  puts "Error getting queue attributes: #{e.message}"
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = "us-west-2"
  queue_name = "my-queue"

  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Listing available queue URLs..."
  list_queue_urls(sqs_client)

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
  queue_url = "https://sqs." + region + ".amazonaws.com/" +
    sts_client.get_caller_identity.account + "/" + queue_name


  puts "\nGetting information about queue '#{queue_name}'..."
  list_queue_attributes(sqs_client, queue_url)
end
```

- Para API obter detalhes, consulte [ListQueues](#) em AWS SDK for Ruby API Referência.

## ReceiveMessage

O código de exemplo a seguir mostra como usar `ReceiveMessage`.

## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da 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__
```

- Para API obter detalhes, consulte [ReceiveMessage](#) em AWS SDK for Ruby API Referência.

## SendMessage

O código de exemplo a seguir mostra como usar SendMessage.

## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
require "aws-sdk-sqs"
require "aws-sdk-sts"

# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param message_body [String] The contents of the message to be sent.
# @return [Boolean] true if the message was sent; otherwise, false.
# @example
#   exit 1 unless message_sent?(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue',
#     'This is my message.'
#   )
def message_sent?(sqs_client, queue_url, message_body)
  sqs_client.send_message(
    queue_url: queue_url,
    message_body: message_body
  )
  true
rescue StandardError => e
  puts "Error sending message: #{e.message}"
  false
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = "us-west-2"
  queue_name = "my-queue"
  message_body = "This is my message."

  sts_client = Aws::STS::Client.new(region: region)
```

```

# For example:
# 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
queue_url = "https://sqs." + region + ".amazonaws.com/" +
  sts_client.get_caller_identity.account + "/" + queue_name

sqs_client = Aws::SQS::Client.new(region: region)

puts "Sending a message to the queue named '#{queue_name}'..."

if message_sent?(sqs_client, queue_url, message_body)
  puts "Message sent."
else
  puts "Message not sent."
end
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__

```

- Para API obter detalhes, consulte [SendMessage](#) em AWS SDK for Ruby API Referência.

## SendMessageBatch

O código de exemplo a seguir mostra como usar SendMessageBatch.

SDKpara Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```

require "aws-sdk-sqs"
require "aws-sdk-sts"

#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param entries [Hash] The contents of the messages to be sent,

```

```
# in the correct format.
# @return [Boolean] true if the messages were sent; otherwise, false.
# @example
#   exit 1 unless messages_sent?(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue',
#     [
#       {
#         id: 'Message1',
#         message_body: 'This is the first message.'
#       },
#       {
#         id: 'Message2',
#         message_body: 'This is the second message.'
#       }
#     ]
#   )
def messages_sent?(sqs_client, queue_url, entries)
  sqs_client.send_message_batch(
    queue_url: queue_url,
    entries: entries
  )
  true
rescue StandardError => e
  puts "Error sending messages: #{e.message}"
  false
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = "us-west-2"
  queue_name = "my-queue"
  entries = [
    {
      id: "Message1",
      message_body: "This is the first message."
    },
    {
      id: "Message2",
      message_body: "This is the second message."
    }
  ]
]
```

```
sts_client = Aws::STS::Client.new(region: region)

# For example:
# 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
queue_url = "https://sqs." + region + ".amazonaws.com/" +
  sts_client.get_caller_identity.account + "/" + queue_name

sqs_client = Aws::SQS::Client.new(region: region)

puts "Sending messages to the queue named '#{queue_name}'..."

if messages_sent?(sqs_client, queue_url, entries)
  puts "Messages sent."
else
  puts "Messages not sent."
end
end
```

- Para API obter detalhes, consulte [SendMessageBatch](#) em AWS SDK for Ruby API Referência.

## Exemplos sem servidor

Invocar uma função Lambda a partir de um gatilho da Amazon SQS

O exemplo de código a seguir mostra como implementar uma função Lambda que recebe um evento acionado pelo recebimento de mensagens de uma SQS fila. A função recupera as mensagens do parâmetro event e registra o conteúdo de cada mensagem.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Consumir um SQS evento com Lambda usando 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
```

## Relatando falhas de itens em lote para funções Lambda com um gatilho da Amazon SQS

O exemplo de código a seguir mostra como implementar uma resposta parcial em lote para funções Lambda que recebem eventos de uma SQS fila. A função relata as falhas do item em lote na resposta, sinalizando para o Lambda tentar novamente essas mensagens posteriormente.

### SDKpara Ruby

#### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no repositório dos [Exemplos sem servidor](#).

Relatar falhas SQS de itens em lote com o Lambda usando 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 exemplos de uso SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Ruby with AWS STS.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

## Ações

### **AssumeRole**

O código de exemplo a seguir mostra como usar `AssumeRole`.

## SDK para Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

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

# Gets temporary credentials that can be used to assume a role.
#
# @param role_arn [String] The ARN of the role that is assumed when these
credentials
#           are used.
# @param sts_client [Aws::STS::Client] An AWS STS client.
# @return [Aws::AssumeRoleCredentials] The credentials that can be used to assume
the role.
def assume_role(role_arn, sts_client)
  credentials = Aws::AssumeRoleCredentials.new(
    client: sts_client,
    role_arn: role_arn,
    role_session_name: "create-use-assume-role-scenario"
  )
  @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
  credentials
end
```

- Para API obter detalhes, consulte [AssumeRole](#) em AWS SDK for Ruby API Referência.

# Exemplos do Amazon Textract usando SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Ruby com o Amazon Textract.

Os cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

## Tópicos

- [Cenários](#)

## Cenários

### Criar uma aplicação para analisar o feedback dos clientes

O exemplo de código a seguir mostra como criar uma aplicação que analisa os cartões de comentários dos clientes, os traduz do idioma original, determina seus sentimentos e gera um arquivo de áudio do texto traduzido.

### SDKpara Ruby

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter o código-fonte e as instruções de implantação, consulte o projeto em [GitHub](#).

## Serviços utilizados neste exemplo

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

## Exemplos do Amazon Translate usando SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Ruby com o Amazon Translate.

Os cenários são exemplos de código que mostram como realizar tarefas específicas chamando várias funções dentro de um serviço ou combinadas com outros Serviços da AWS.

Cada exemplo inclui um link para o código-fonte completo, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

### Tópicos

- [Cenários](#)

## Cenários

### Criar uma aplicação para analisar o feedback dos clientes

O exemplo de código a seguir mostra como criar uma aplicação que analisa os cartões de comentários dos clientes, os traduz do idioma original, determina seus sentimentos e gera um arquivo de áudio do texto traduzido.

### SDK para Ruby

Esta aplicação de exemplo analisa e armazena cartões de feedback de clientes. Especificamente, ela atende à necessidade de um hotel fictício na cidade de Nova York. O hotel recebe feedback dos hóspedes em vários idiomas na forma de cartões de comentários físicos. Esse feedback é enviado para a aplicação por meio de um cliente web. Depois de fazer upload da imagem de um cartão de comentário, ocorrem as seguintes etapas:

- O texto é extraído da imagem usando o Amazon Textract.
- O Amazon Comprehend determina o sentimento do texto extraído e o idioma.
- O texto extraído é traduzido para o inglês com o Amazon Translate.
- O Amazon Polly sintetiza um arquivo de áudio do texto extraído.

A aplicação completa pode ser implantada com o AWS CDK. Para obter o código-fonte e as instruções de implantação, consulte o projeto em [GitHub](#).

Serviços utilizados neste exemplo

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

## WorkDocs Exemplos da Amazon usando SDK para Ruby

Os exemplos de código a seguir mostram como realizar ações e implementar cenários comuns usando o AWS SDK for Ruby com a Amazon WorkDocs.

Ações são trechos de código de programas maiores e devem ser executadas em contexto. Embora as ações mostrem como chamar funções de serviço individuais, você pode ver as ações no contexto em seus cenários relacionados.

Cada exemplo inclui um link para o código-fonte completo, onde você pode encontrar instruções sobre como configurar e executar o código no contexto.

Tópicos

- [Ações](#)

### Ações

#### **DescribeRootFolders**

O código de exemplo a seguir mostra como usar `DescribeRootFolders`.

## SDKpara Ruby

 Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
# Retrieves the root folder for a user by email
# @param users [Array<Types::User>] A list of users selected from API response
# @param user_email [String] The email of the user.
def get_user_folder(users, user_email)
  user = users.find { |user| user.email_address == user_email }
  if user
    user.root_folder_id
  else
    @logger.error "Could not get root folder for user with email address
#{user_email}"
    exit(1)
  end
end

# Describes the contents of a folder
# @param [String] folder_id - The Id of the folder to describe.
def describe_folder_contents(folder_id)
  resp = @client.describe_folder_contents({
                                     folder_id: folder_id, # required
                                     sort: "NAME", # accepts DATE, NAME
                                     order: "ASCENDING", # accepts
ASCENDING, DESCENDING
                                     })

  resp.documents.each do |doc|
    md = doc.latest_version_metadata
    @logger.info "Name:          #{md.name}"
    @logger.info "Size (bytes):  #{md.size}"
    @logger.info "Last modified: #{doc.modified_timestamp}"
    @logger.info "Doc ID:        #{doc.id}"
    @logger.info "Version ID:    #{md.id}"
    @logger.info ""
  end
rescue Aws::WorkDocs::Errors::ServiceError => e
  @logger.error "Error listing folder contents: #{e.message}"
end
```

```
    exit(1)
  end
```

- Para API obter detalhes, consulte [DescribeRootFolders](#) em AWS SDK for Ruby API Referência.

## DescribeUsers

O código de exemplo a seguir mostra como usar DescribeUsers.

SDK para Ruby

### Note

Tem mais sobre GitHub. Encontre o exemplo completo e saiba como configurar e executar no [Repositório de exemplos de código da AWS](#).

```
# Describes users within an organization
# @param [String] org_id: The ID of the org.
def describe_users(org_id)
  resp = @client.describe_users({
    organization_id: org_id,
    include: "ALL", # accepts ALL, ACTIVE_PENDING
    order: "ASCENDING", # accepts ASCENDING,
DESCENDING
    sort: "USER_NAME", # accepts USER_NAME,
FULL_NAME, STORAGE_LIMIT, USER_STATUS, STORAGE_USED
  })

  resp.users.each do |user|
    @logger.info "First name: #{user.given_name}"
    @logger.info "Last name:  #{user.surname}"
    @logger.info "Email:      #{user.email_address}"
    @logger.info "Root folder: #{user.root_folder_id}"
    @logger.info ""
  end
  resp.users
rescue Aws::WorkDocs::Errors::ServiceError => e
  @logger.error "AWS WorkDocs Service Error: #{e.message}"
  exit(1)
end
```



- Para API obter detalhes, consulte [DescribeUsers](#) em AWS SDK for Ruby API Referência.

# Segurança AWS SDK para Ruby

A segurança da nuvem na Amazon Web Services (AWS) é a nossa maior prioridade. Como cliente da AWS, você contará com um data center e uma arquitetura de rede criados para atender aos requisitos das organizações com as maiores exigências de segurança. A segurança é uma responsabilidade compartilhada entre você AWS e você. O [modelo de responsabilidade compartilhada](#) descreve isso como a Segurança da nuvem e a Segurança na nuvem.

Segurança da nuvem — AWS é responsável por proteger a infraestrutura que executa todos os serviços oferecidos na AWS nuvem e fornecer serviços que você possa usar com segurança. Nossa responsabilidade de segurança é a maior prioridade em AWS, e a eficácia de nossa segurança é regularmente testada e verificada por auditores terceirizados como parte dos [Programas de AWS Conformidade](#).

Segurança na nuvem — Sua responsabilidade é determinada pelo Serviço da AWS que você está usando e por outros fatores, incluindo a sensibilidade de seus dados, os requisitos da sua organização e as leis e regulamentos aplicáveis.

## Tópicos

- [Proteção de dados AWS SDK para Ruby](#)
- [Identity and Access Management AWS SDK para Ruby](#)
- [Validação de conformidade AWS SDK para Ruby](#)
- [Resiliência AWS SDK para Ruby](#)
- [Segurança de infraestrutura AWS SDK para Ruby](#)
- [Aplicando uma TLS versão mínima no AWS SDK para Ruby](#)
- [Migração do cliente de criptografia do Amazon S3](#)

## Proteção de dados AWS SDK para Ruby

O modelo de [responsabilidade AWS compartilhada modelo](#) se aplica à proteção de dados em. Conforme descrito neste modelo, AWS é responsável por proteger a infraestrutura global que executa todos os Nuvem AWS. Você é responsável por manter o controle sobre seu conteúdo hospedado nessa infraestrutura. Você também é responsável pelas tarefas de configuração e gerenciamento de segurança dos Serviços da AWS que usa. Para obter mais informações sobre

privacidade de dados, consulte [Privacidade de dados FAQ](#). Para obter informações sobre proteção de dados na Europa, consulte o [Modelo de Responsabilidade AWS Compartilhada e GDPR](#) a postagem no blog AWS de segurança.

Para fins de proteção de dados, recomendamos que você proteja Conta da AWS as credenciais e configure usuários individuais com AWS IAM Identity Center ou AWS Identity and Access Management (IAM). Dessa maneira, cada usuário receberá apenas as permissões necessárias para cumprir suas obrigações de trabalho. Recomendamos também que você proteja seus dados das seguintes formas:

- Use a autenticação multifator (MFA) com cada conta.
- Use SSL/TLS para se comunicar com AWS os recursos. Exigimos TLS 1,2 e recomendamos TLS 1,3.
- Configure API e registre as atividades do usuário com AWS CloudTrail.
- Use soluções de AWS criptografia, juntamente com todos os controles de segurança padrão Serviços da AWS.
- Use serviços gerenciados de segurança avançada, como o Amazon Macie, que ajuda a localizar e proteger dados sigilosos armazenados no Amazon S3.
- Se você precisar de FIPS 140-3 módulos criptográficos validados ao acessar AWS por meio de uma interface de linha de comando ou uma API, use um endpoint. FIPS Para obter mais informações sobre os FIPS endpoints disponíveis, consulte [Federal Information Processing Standard \(FIPS\) 140-3](#).

É altamente recomendável que nunca sejam colocadas informações de identificação confidenciais, como endereços de e-mail dos seus clientes, em marcações ou campos de formato livre, como um campo Nome. Isso inclui quando você trabalha com ou Serviços da AWS usa o console, API, AWS CLI, ou AWS SDKs. Quaisquer dados inseridos em tags ou campos de texto de formato livre usados para nomes podem ser usados para logs de faturamento ou de diagnóstico. Se você fornecer um URL para um servidor externo, é altamente recomendável que você não inclua informações de credenciais no URL para validar sua solicitação para esse servidor.

## Identity and Access Management AWS SDK para Ruby

AWS Identity and Access Management (IAM) é um serviço da Amazon Web Services (AWS) que ajuda o administrador a controlar com segurança o acesso aos AWS recursos. IAM os administradores controlam quem pode ser autenticado (conectado) e autorizado (tem permissões)

a usar os recursos. Serviços da AWS IAM é um Serviço da AWS que você pode usar sem custo adicional.

Para usar o Ruby AWS SDK para acessar AWS, você precisa de uma AWS conta e AWS credenciais. Para aumentar a segurança da sua AWS conta, recomendamos que você use um IAM usuário para fornecer credenciais de acesso em vez de usar as credenciais da sua AWS conta.

Para obter detalhes sobre como trabalhar com IAM, consulte [IAM](#).

Para uma visão geral dos IAM usuários e por que eles são importantes para a segurança da sua conta, consulte [Credenciais de AWS segurança](#) na [Referência geral da Amazon Web Services](#).

AWS SDK for Ruby segue o [modelo de responsabilidade compartilhada](#) por meio dos serviços específicos da Amazon Web Services (AWS) que ele suporta. Para obter informações sobre Serviço da AWS segurança, consulte a [página Serviço da AWS de documentação](#) de segurança e [Serviços da AWS quais estão no escopo dos esforços de AWS conformidade do programa de conformidade](#).

## Validação de conformidade AWS SDK para Ruby

AWS SDK for Ruby segue o [modelo de responsabilidade compartilhada](#) por meio dos serviços específicos da Amazon Web Services (AWS) que ele suporta. Para obter informações sobre Serviço da AWS segurança, consulte a [página Serviço da AWS de documentação](#) de segurança e [Serviços da AWS quais estão no escopo dos esforços de AWS conformidade do programa de conformidade](#).

A segurança e a conformidade dos serviços da Amazon Web Services (AWS) são avaliadas por auditores terceirizados como parte de vários programas de AWS conformidade. Isso inclui SOC, PCI RAMPHIPAA, Fed e outros. AWS fornece uma lista frequentemente atualizada do escopo de Serviços da AWS programas de conformidade específicos em [AWS Services in Scope by Compliance Program](#).

Relatórios de auditoria de terceiros estão disponíveis para você baixar usando AWS Artifact. Para obter mais informações, consulte [Baixando relatórios no AWS Artifact](#).

Para obter mais informações sobre programas de AWS conformidade, consulte [Programas de AWS conformidade](#).

Sua responsabilidade de conformidade ao usar o AWS SDK Ruby para acessar um Serviço da AWS é determinada pela confidencialidade de seus dados, pelos objetivos de conformidade de sua organização e pelas leis e regulamentos aplicáveis. Se o uso de um Serviço da AWS estiver sujeito à conformidade com padrões como HIPAA, PCI ou o FedRAMP, AWS fornece recursos para ajudar a:

- Guias de [início rápido sobre segurança e conformidade — guias](#) de implantação que discutem considerações arquitetônicas e fornecem etapas para a implantação de ambientes básicos focados na segurança e na conformidade em AWS.
- Documento técnico [sobre arquitetura para HIPAA segurança e conformidade — Um whitepaper](#) que descreve como as empresas podem usar AWS para criar aplicativos compatíveis. HIPAA
- [AWS Recursos de conformidade](#) — Uma coleção de pastas de trabalho e guias que podem ser aplicados ao seu setor e localização.
- [AWS Config](#) — um serviço que avalia o quão bem suas configurações de recursos estão em conformidade com as práticas internas, as diretrizes do setor e os regulamentos.
- [AWS Security Hub](#) — Uma visão abrangente do seu estado de segurança interno AWS que ajuda você a verificar sua conformidade com os padrões e as melhores práticas do setor de segurança.

## Resiliência AWS SDK para Ruby

A infraestrutura global da Amazon Web Services (AWS) é construída em torno Regiões da AWS de zonas de disponibilidade.

Regiões da AWS fornecem várias zonas de disponibilidade fisicamente separadas e isoladas, conectadas a redes de baixa latência, alta taxa de transferência e alta redundância.

Com as Zonas de Disponibilidade, é possível projetar e operar aplicações e bancos de dados que executem o failover automaticamente entre as Zonas de Disponibilidade sem interrupção. As zonas de disponibilidade são mais altamente disponíveis, tolerantes a falhas e escaláveis que uma ou várias infraestruturas de datacenter tradicionais.

Para obter mais informações sobre zonas de disponibilidade Regiões da AWS e zonas de disponibilidade, consulte [Infraestrutura AWS global](#).

AWS SDK for Ruby segue o [modelo de responsabilidade compartilhada](#) por meio dos serviços específicos da Amazon Web Services (AWS) que ele suporta. Para obter informações sobre Serviço da AWS segurança, consulte a [página Serviço da AWS de documentação](#) de segurança e [Serviços da AWS quais estão no escopo dos esforços de AWS conformidade do programa de conformidade](#).

## Segurança de infraestrutura AWS SDK para Ruby

AWS SDK for Ruby segue o [modelo de responsabilidade compartilhada](#) por meio dos serviços específicos da Amazon Web Services (AWS) que ele suporta. Para obter informações sobre Serviço

da AWS segurança, consulte a [página Serviço da AWS de documentação](#) de segurança e [Serviços da AWS quais estão no escopo dos esforços de AWS conformidade do programa de conformidade](#).

Para obter informações sobre processos AWS de segurança, consulte o whitepaper [AWS: Visão geral dos processos de segurança](#).

## Aplicando uma TLS versão mínima no AWS SDK para Ruby

A comunicação entre o AWS SDK for Ruby e AWS é protegida usando Secure Sockets Layer (SSL) ou Transport Layer Security (TLS). Todas as versões SSL e versões TLS anteriores à 1.2 têm vulnerabilidades que podem comprometer a segurança de sua comunicação com AWS. Por esse motivo, você deve se certificar de que está usando o AWS SDK for Ruby com uma versão do Ruby compatível com a TLS versão 1.2 ou posterior.

O Ruby usa a SSL biblioteca Open para proteger HTTP as conexões. As versões suportadas do Ruby (1.9.3 e posteriores) instaladas por meio de [gerenciadores de pacotes do sistema](#) (yum, apt, e outros), de um [instalador oficial](#) ou de [gerenciadores](#) Ruby (rbenv e outros) normalmente incorporam o Open SSL 1.0.1 ou posterior RVM, que suporta 1.2. TLS

Quando usado com uma versão compatível do Ruby com Open SSL 1.0.1 ou posterior, o AWS SDK for Ruby prefere TLS 1.2 e usa a versão mais recente SSL ou TLS suportada pelo cliente e pelo servidor. Isso é sempre pelo menos TLS 1,2 para Serviços da AWS. (SDKEle usa a `Net::HTTP` classe Ruby com `use_ssl=true`.)

## Verificando a SSL versão aberta

Para garantir que sua instalação do Ruby esteja usando o Open SSL 1.0.1 ou posterior, digite o seguinte comando.

```
ruby -r openssl -e 'puts OpenSSL::OPENSSL_VERSION'
```

Uma forma alternativa de obter a SSL versão Open é consultar o `openssl` executável diretamente. Primeiro, localize o executável apropriado usando o comando a seguir.

```
ruby -r rbconfig -e 'puts RbConfig::CONFIG["configure_args"]'
```

A saída deve `--with-openssl-dir=/path/to/openssl` indicar a localização da SSL instalação aberta. Anote esse caminho. Para verificar a versão do OpenSSL, digite os seguintes comandos.

```
cd /path/to/openssl
bin/openssl version
```

Esse último método pode não funcionar com todas as instalações do Ruby.

## Suporte de atualização TLS

[Se a versão do Open SSL usada pela sua instalação do Ruby for anterior à 1.0.1, atualize sua instalação do Ruby ou do Open usando o gerenciador de pacotes do sistema, o instalador do Ruby ou o gerenciador do Ruby, conforme descrito no guia de SSL instalação do Ruby.](#) Se você estiver instalando o Ruby [a partir do código-fonte](#), instale SSL primeiro o [Open mais recente](#) e depois passe `--with-openssl-dir=/path/to/upgraded/openssl` durante a execução `./configure`.

## Migração do cliente de criptografia do Amazon S3

Este tópico mostra como migrar as aplicações da versão 1 (V1) do cliente de criptografia do Amazon Simple Storage Service (Amazon S3) para a versão 2 (V2) e garantir a disponibilidade das aplicações durante todo o processo de migração.

### Visão geral da migração

Essa migração acontece em duas fases:

1. Atualize os clientes existentes para ler novos formatos. Primeiro, implante uma versão atualizada do AWS SDK for Ruby em seu aplicativo. Isso permitirá que os clientes de criptografia existentes da V1 descriptografem objetos gravados pelos novos clientes da V2. Se seu aplicativo usa vários AWS SDKs, você deve atualizar cada um SDK separadamente.
2. Migrar clientes de criptografia e descriptografia para a V2. Depois que todos os seus clientes de criptografia da V1 puderem ler os novos formatos, você poderá migrar seus clientes de criptografia e descriptografia existentes para suas respectivas versões da V2.

### Atualizar os clientes existentes para ler novos formatos

O cliente de criptografia da V2 usa algoritmos de criptografia incompatíveis com as versões mais antigas do cliente. A primeira etapa da migração é atualizar seus clientes de decodificação V1 para a versão mais recente. SDK Depois de concluir essa etapa, os clientes da V1 da sua aplicação poderão descriptografar objetos criptografados por clientes de criptografia da V2. Veja os detalhes abaixo para cada versão principal do AWS SDK para Ruby.

## Atualização AWS SDK para a versão 3 do Ruby

A versão 3 é a versão mais recente do AWS SDK For Ruby. Para concluir essa migração, você precisa utilizar a versão 1.76.0 ou posterior do gem `aws-sdk-s3`.

Instalação a partir da linha de comando

Para projetos que instalam o gem `aws-sdk-s3`, use a opção de versão para verificar se a versão mínima 1.76.0 está instalada.

```
gem install aws-sdk-s3 -v '>= 1.76.0'
```

Uso de Gemfiles

Para projetos que usam um Gemfile para gerenciar dependências, defina a versão mínima do gem `aws-sdk-s3` como 1.76.0. Por exemplo:

```
gem 'aws-sdk-s3', '>= 1.76.0'
```

1. Modificar seu Gemfile.
2. Executar `bundle update aws-sdk-s3`. Para verificar sua versão, execute o `bundle info aws-sdk-s3`.

## Atualização AWS SDK para Ruby versão 2

A versão 2 do AWS SDK para Ruby entrará no [modo de manutenção](#) em 21 de novembro de 2021. Para concluir essa migração, você precisa utilizar a versão 2.11.562 ou posterior do gem `aws-sdk`.

Instalação a partir da linha de comando

Para projetos que instalam o gem `aws-sdk`, a partir da linha de comando, use a opção `version` para verificar se a versão mínima 2.11.562 está instalada.

```
gem install aws-sdk -v '>= 2.11.562'
```

Uso de Gemfiles

Para projetos que usam um Gemfile para gerenciar dependências, defina a versão mínima do gem `aws-sdk` como 2.11.562. Por exemplo:



```
gem 'aws-sdk', '>= 2.11.562'
```

1. Modificar seu Gemfile. Se você tiver um arquivo Gemfile.lock, exclua-o ou atualize-o.
2. Executar `bundle update aws-sdk`. Para verificar sua versão, execute o `bundle info aws-sdk`.

## Migrar clientes de criptografia e descriptografia para a V2

Depois de atualizar seus clientes para ler os novos formatos de criptografia, você pode atualizar suas aplicações para os clientes de criptografia e descriptografia da V2. As etapas a seguir mostram como migrar com sucesso seu código da V1 para a V2.

Antes de atualizar seu código para usar o cliente de criptografia V2, verifique se você seguiu as etapas anteriores e está usando o `gem aws-sdk-s3` versão 2.11.562 ou posterior.

### Note

Ao descriptografar com AES -GCM, leia o objeto inteiro até o final antes de começar a usar os dados descriptografados. Isso é para verificar se o objeto não foi modificado desde que foi criptografado.

## Configuração de clientes de criptografia V2

O `EncryptionV2::Client` requer configuração adicional. Para obter informações detalhadas da configuração, consulte a [documentação EncryptionV2::Client](#) ou os exemplos fornecidos posteriormente neste tópico.

1. O método de chave e o algoritmo de criptografia de conteúdo devem ser especificados durante a construção do cliente. Ao criar um novo `EncryptionV2::Client`, você precisa fornecer valores para `key_wrap_schema` e `content_encryption_schema`.

`key_wrap_schema`- Se você estiver usando AWS KMS, isso deve ser definido como `:kms_context`. Se você estiver usando uma tecla simétrica (AES), ela deverá ser definida como `:aes_gcm`. Se você estiver usando uma chave assimétrica (RSA), ela deverá ser definida como `:rsa_oaep_sha1`.

`content_encryption_schema`: isso deve ser definido como `:aes_gcm_no_padding`.

2. `security_profile` deve ser especificado durante a construção do cliente. Ao criar um novo `EncryptionV2::Client`, você precisa fornecer um valor para `security_profile`. O parâmetro `security_profile` determina o suporte para leitura de objetos gravados usando o `Encryption::Client` da V1 mais antiga. Há dois valores: `:v2` e `:v2_and_legacy`. Para oferecer suporte à migração, defina o `security_profile` como `:v2_and_legacy`. Use `:v2` somente para desenvolvimento de novo aplicativo.

3. AWS KMS key O ID é aplicado por padrão. Na V1, `Encryption::Client`, o `kms_key_id` usado para criar o cliente não foi fornecido ao AWS KMS `Decrypt` call. AWS KMS pode obter essas informações dos metadados e adicioná-las ao blob de texto cifrado simétrico. Na V2, `EncryptionV2::Client`, o `kms_key_id` é passado para a chamada `Decrypt`, e a chamada falhará se não corresponder à chave usada para AWS KMS criptografar o objeto. Se o seu código anteriormente dependia da não definição de um `kms_key_id` específico, defina `kms_key_id: :kms_allow_decrypt_with_any_cmk` na criação do cliente ou defina `kms_allow_decrypt_with_any_cmk: true` nas chamadas do `get_object`.

## Exemplo: usando uma chave simétrica (AES)

### Pré-migração

```
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)
```

### Pós-migração

```
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
```

## Exemplo: usando AWS KMS com `kms_key_id`

### Pré-migração

```
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)
```

## Pós-migração

```
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
```

## Exemplo: usando AWS KMS sem kms\_key\_id

### Pré-migração

```
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)
```

### Pós-migração

```
client = Aws::S3::EncryptionV2::Client.new(
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context, # the key_wrap_schema must be kms_context for KMS keys
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v2_and_legacy # to allow reading/decrypting objects encrypted by
  the V1 encryption client
)
client.put_object(bucket: bucket, key: key, body: secret_data) # No changes
resp = client.get_object(bucket: bucket, key: key, kms_allow_decrypt_with_any_cmk:
  true) # To allow decrypting with any cmk
```

## Alternativa pós-migração

Se você apenas lê e descriptografa (nunca grava e criptografa) objetos usando o cliente de criptografia S2, use este código.

```
client = Aws::S3::EncryptionV2::Client.new(
  kms_key_id: :kms_allow_decrypt_with_any_cmk, # set kms_key_id to allow all get_object
  requests to use any cmk
  key_wrap_schema: :kms_context, # the key_wrap_schema must be kms_context for KMS keys
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v2_and_legacy # to allow reading/decrypting objects encrypted by
  the V1 encryption client
)
resp = client.get_object(bucket: bucket, key: key) # No change
```

# Histórico do documento

A tabela a seguir descreve as alterações importantes neste guia. Para receber notificações sobre atualizações desta documentação, você pode assinar um [RSSfeed](#).

Alteração	Descrição	Data
<a href="#">Índice e exemplos guiados</a>	Os exemplos guiados foram removidos para serem transferidos para o repositório de exemplos de código mais abrangente.	10 de julho de 2024
<a href="#">Índice</a>	Índice atualizado para tornar os exemplos de código mais acessíveis.	1.º de junho de 2023
<a href="#">IAM atualizações de melhores práticas</a>	Guia atualizado para se alinhar às IAM melhores práticas. Para obter mais informações, consulte <a href="#">Melhores práticas de segurança em IAM</a> . Atualizações dos Conceitos básicos.	8 de maio de 2023
<a href="#">Atualizações gerais</a>	Atualização da página de boas-vindas para recursos externos relevantes. Versão mínima exigida do Ruby atualizada para v2.3. AWS Key Management Service Seções atualizadas para refletir as atualizações de terminologia. Informações de uso atualizadas sobre o REPL utilitário para maior clareza.	8 de agosto de 2022

## [Corrigindo links quebrados](#)

Links de exemplos quebrados corrigidos. Página redundante de dicas e truques removida; redirecionando para o conteúdo de exemplo da AmazonEC2. Listas incluídas dos exemplos de código que estão disponíveis GitHub no repositório de exemplos de código.

3 de agosto de 2022

## [SDK Métricas](#)

Foram removidas as informações sobre a ativação do SDK Metrics for Enterprise Support, que se tornou obsoleto.

28 de janeiro de 2022

As traduções são geradas por tradução automática. Em caso de conflito entre o conteúdo da tradução e da versão original em inglês, a versão em inglês prevalecerá.