

Developer Guide

AWS SDK for Ruby



AWS SDK for Ruby: Developer Guide

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

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What is the AWS SDK for Ruby?	1
Additional documentation and resources	1
Deploying to the AWS Cloud	2
Maintenance and support for SDK major versions	2
Get started	3
SDK authentication with AWS	3
Start an AWS access portal session	4
More authentication information	5
Install the SDK	5
Prerequisites	6
Installing the SDK	6
Hello tutorial	7
Write the code	7
Running the program	8
Note for Windows users	8
Next steps	9
Use AWS Cloud9 with the SDK	9
Step 1: Set up your AWS account to use AWS Cloud9	10
Step 2: Set up your AWS Cloud9 development environment	10
Step 3: Set up the AWS SDK for Ruby	11
Step 4: Download example code	12
Step 5: Run example code	13
Configure the SDK	14
Credential provider chain	14
Creating an AWS STS access token	15
Setting a Region	16
Setting the Region using the shared config file	17
Setting the Region using environment variables	17
Setting the Region with <code>Aws.config</code>	17
Setting the Region in a client or resource object	18
Setting a nonstandard endpoint	18
Use the SDK	19
Use the REPL utility	19
Prerequisites	19

Bundler setup	20
Running REPL	20
Use the SDK with Ruby on Rails	21
Debugging tip: Get wire trace information from a client	21
Stub client responses and errors	22
Stubbing client responses	22
Stubbing client errors	24
Pagination	24
Paged responses are enumerable	24
Handling paged responses manually	25
Paged data classes	25
Waiters	25
Invoking a waiter	26
Wait failures	26
Configuring a waiter	27
Extending a waiter	27
Specify client retry behavior	28
Migrate from version 1 or 2 to version 3 of the AWS SDK for Ruby	28
Side-by-side usage	29
General differences	29
Client differences	30
Resource differences	31
Work with AWS services	32
Code examples with guidance	32
AWS CloudTrail Examples	33
Amazon CloudWatch Examples	39
AWS CodeBuild Examples	72
Amazon EC2 Examples	75
AWS Elastic Beanstalk Examples	129
AWS Identity and Access Management (IAM) Examples	133
AWS KMS Examples	168
AWS Lambda Examples	171
Amazon Polly Examples	177
Amazon RDS Examples	181
Amazon SES Examples	188
Amazon SNS Examples	194

Amazon SQS Examples	199
Amazon WorkDocs Examples	226
Code examples	230
Actions and scenarios	230
CloudTrail	231
CloudWatch	236
DynamoDB	248
Amazon EC2	275
Elastic Beanstalk	310
EventBridge	315
AWS Glue	337
IAM	365
Kinesis	420
AWS KMS	423
Lambda	427
Amazon Polly	450
Amazon RDS	454
Amazon S3	458
Amazon SES	488
Amazon SES API v2	494
Amazon SNS	495
Amazon SQS	505
AWS STS	518
Amazon WorkDocs	520
Cross-service examples	522
Create an application to analyze customer feedback	523
Security	524
Data Protection	524
Identity and Access Management	525
Compliance Validation	526
Resilience	527
Infrastructure Security	527
Enforcing a minimum TLS version	527
Checking the OpenSSL version	528
Upgrading TLS support	528
S3 Encryption Client Migration	529

Migration Overview	529
Update Existing Clients to Read New Formats	529
Migrate Encryption and Decryption Clients to V2	530
Document History	534

What is the AWS SDK for Ruby?

Welcome to the AWS SDK for Ruby Developer Guide. The AWS SDK for Ruby provides support libraries for almost all AWS services, including Amazon Simple Storage Service (Amazon S3), Amazon Elastic Compute Cloud (Amazon EC2), and Amazon DynamoDB.

The AWS SDK for Ruby Developer Guide provides information about how to install, set up, and use the AWS SDK for Ruby to create Ruby applications that use AWS services.

[Get started with the AWS SDK for Ruby](#)

Additional documentation and resources

For more resources for AWS SDK for Ruby developers, see the following:

- [AWS SDKs and Tools Reference Guide](#) – Contains settings, features, and other foundational concepts common among AWS SDKs
- [AWS SDK for Ruby API Reference - Version 3](#)
- [AWS Code Examples Repository](#) on GitHub
- [RubyGems.org](#) – Latest version of SDK is modularized into service-specific gems available here
 - [Supported Services](#) – Lists all gems that the AWS SDK for Ruby supports
- AWS SDK for Ruby source on GitHub:
 - [Source](#) and [README](#)
 - [Change logs under each gem](#)
 - [Moving from v2 to v3](#)
 - [Issues](#)
 - [Core upgrade notes](#)
- [Developer blog](#)
- [Gitter channel](#)
- [@awsforruby](#) on Twitter

Deploying to the AWS Cloud

You can use AWS services such as AWS Elastic Beanstalk, AWS OpsWorks, and AWS CodeDeploy to deploy your application to the AWS Cloud. For deploying Ruby applications with Elastic Beanstalk, see [Deploying Elastic Beanstalk Applications in Ruby Using EB CLI and Git](#) in the AWS Elastic Beanstalk Developer Guide. For deploying a Ruby on Rails application with AWS OpsWorks, see [Deploying Ruby on Rails Applications to AWS OpsWorks](#). For an overview of AWS deployment services, see [Overview of Deployment Options on AWS](#).

Maintenance and support for SDK major versions

For information about maintenance and support for SDK major versions and their underlying dependencies, see the following in the [AWS SDKs and Tools Reference Guide](#):

- [AWS SDKs and Tools Maintenance Policy](#)
- [AWS SDKs and Tools Version Support Matrix](#)

Get started with the AWS SDK for Ruby

Learn how to install, set up, and use the SDK to create a Ruby application to access an AWS resource programmatically.

Topics

- [SDK authentication with AWS](#)
- [Install the AWS SDK for Ruby](#)
- [Hello tutorial for the AWS SDK for Ruby](#)
- [Use AWS Cloud9 with the AWS SDK for Ruby](#)

SDK authentication with AWS

You must establish how your code authenticates with AWS when developing with AWS services. You can configure programmatic access to AWS resources in different ways depending on the environment and the AWS access available to you.

To choose your method of authentication and configure it for the SDK, see [Authentication and access](#) in the *AWS SDKs and Tools Reference Guide*.

We recommend that new users who are developing locally and are not given a method of authentication by their employer should set up AWS IAM Identity Center. This method includes installing the AWS CLI for ease of configuration and for regularly signing in to the AWS access portal. If you choose this method, your environment should contain the following elements after you complete the procedure for [IAM Identity Center authentication](#) in the *AWS SDKs and Tools Reference Guide*:

- The AWS CLI, which you use to start an AWS access portal session before you run your application.
- A [shared AWSconfig file](#) having a [default] profile with a set of configuration values that can be referenced from the SDK. To find the location of this file, see [Location of the shared files](#) in the *AWS SDKs and Tools Reference Guide*.
- The shared config file sets the [region](#) setting. This sets the default AWS Region that the SDK uses for AWS requests. This Region is used for SDK service requests that aren't specified with a Region to use.

- The SDK uses the profile's [SSO token provider configuration](#) to acquire credentials before sending requests to AWS. The `sso_role_name` value, which is an IAM role connected to an IAM Identity Center permission set, allows access to the AWS services used in your application.

The following sample config file shows a default profile set up with SSO token provider configuration. The profile's `sso_session` setting refers to the named [sso-session section](#). The `sso-session` section contains settings to initiate an AWS access portal session.

```
[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
```

The AWS SDK for Ruby does not need additional packages (such as SSO and SSO0IDC) to be added to your application to use IAM Identity Center authentication.

Start an AWS access portal session

Before running an application that accesses AWS services, you need an active AWS access portal session for the SDK to use IAM Identity Center authentication to resolve credentials. Depending on your configured session lengths, your access will eventually expire and the SDK will encounter an authentication error. To sign in to the AWS access portal, run the following command in the AWS CLI.

```
aws sso login
```

If you followed the guidance and have a default profile setup, you do not need to call the command with a `--profile` option. If your SSO token provider configuration is using a named profile, the command is `aws sso login --profile named-profile`.

To optionally test if you already have an active session, run the following AWS CLI command.

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

If your session is active, the response to this command reports the IAM Identity Center account and permission set configured in the shared config file.

Note

If you already have an active AWS access portal session and run `aws sso login`, you will not be required to provide credentials.

The sign-in process might prompt you to allow the AWS CLI access to your data. Because the AWS CLI is built on top of the SDK for Python, permission messages might contain variations of the `botocore` name.

More authentication information

Human users, also known as *human identities*, are the people, administrators, developers, operators, and consumers of your applications. They must have an identity to access your AWS environments and applications. Human users that are members of your organization - that means you, the developer - are known as *workforce identities*.

Use temporary credentials when accessing AWS. You can use an identity provider for your human users to provide federated access to AWS accounts by assuming roles, which provide temporary credentials. For centralized access management, we recommend that you use AWS IAM Identity Center (IAM Identity Center) to manage access to your accounts and permissions within those accounts. For more alternatives, see the following:

- To learn more about best practices, see [Security best practices in IAM](#) in the *IAM User Guide*.
- To create short-term AWS credentials, see [Temporary Security Credentials](#) in the *IAM User Guide*.
- To learn about other AWS SDK for Ruby credential providers, see [Standardized credential providers](#) in the *AWS SDKs and Tools Reference Guide*.

Install the AWS SDK for Ruby

This section includes prerequisites and installation instructions for the AWS SDK for Ruby.

Prerequisites

Before you use the AWS SDK for Ruby, you must authenticate with AWS. For information about setting up authentication, see [SDK authentication with AWS](#).

Installing the SDK

You can install the AWS SDK for Ruby as you would any Ruby gem. The gems are available at [RubyGems](#). The AWS SDK for Ruby is designed to be modular and is separated by AWS service. Installing the entire `aws-sdk` gem is large and may take over an hour.

We recommend only installing the gems for the AWS services you use. These are named like `aws-sdk-service_abbreviation` and the complete list is found in the [Supported Services](#) table of the AWS SDK for Ruby README file. For example, the gem for interfacing with the Amazon S3 service is directly available at [aws-sdk-s3](#).

Ruby version manager

Instead of using system Ruby, we recommend using a Ruby version manager such as the following:

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

For example, if you are using an Amazon Linux 2 operating system, the following commands can be used to update RVM, list the available Ruby versions, then choose the version you want to use for development with the AWS SDK for Ruby. The minimum required Ruby version is 2.3.

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

Bundler

If you use [Bundler](#), the following commands install the AWS SDK for Ruby gem for Amazon S3:

1. Install Bundler and create the Gemfile:

```
$ gem install bundler
```

```
$ bundle init
```

2. Open the created Gemfile and add a gem line for each AWS service gem your code will use. To follow along with the Amazon S3 example, add the following line to the bottom of the file:

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

3. Save the Gemfile.
4. Install the dependencies specified in your Gemfile:

```
$ bundle install
```

Hello tutorial for the AWS SDK for Ruby

Say hello to Amazon S3 using the AWS SDK for Ruby. The following example displays a list of your Amazon S3 buckets.

Write the code

Copy and paste the following code into a new source file. Name the file `hello-s3.rb`.

```
require "aws-sdk-s3"

# Wraps Amazon S3 resource actions.
class BucketListWrapper
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end

  # Lists buckets for the current account.
  #
  # @param count [Integer] The maximum number of buckets to list.
  def list_buckets(count)
    puts "Found these buckets:"
    @s3_resource.buckets.each do |bucket|
      puts "\t#{bucket.name}"
      count -= 1
    end
  end
end
```

```
        break if count.zero?
      end
      true
    rescue Aws::Errors::ServiceError => e
      puts "Couldn't list buckets. Here's why: #{e.message}"
      false
    end
  end
end

# Example usage:
def run_demo
  wrapper = BucketListWrapper.new(Aws::S3::Resource.new)
  wrapper.list_buckets(25)
end

run_demo if $PROGRAM_NAME == __FILE__
```

AWS SDK for Ruby is designed to be modular and is separated by AWS service. After the gem is installed, the `require` statement at the top of your Ruby source file imports the AWS SDK classes and methods for the Amazon S3 service. For a complete list of available AWS service gems, see the [Supported Services](#) table of the AWS SDK for Ruby README file.

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

Running the program

Open a command prompt to run your Ruby program. The typical command syntax to run a Ruby program is:

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

This sample code uses no arguments. To run this code, enter the following into the command prompt:

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

Note for Windows users

When you use SSL certificates on Windows and run your Ruby code, you might see an error similar to the following.

```
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'
...

```

To fix this issue, add the following line to your Ruby source file, somewhere before your first AWS call.

```
Aws.use_bundled_cert!
```

If you're using only the `aws-sdk-s3` gem in your Ruby program and you want to use the bundled certificate, you also need to add the `aws-sdk-core` gem.

Next steps

To test out many other Amazon S3 operations, check out the [AWS Code Examples Repository](#) on GitHub.

Use AWS Cloud9 with the AWS SDK for Ruby

AWS Cloud9 is a web-based integrated development environment (IDE) that contains a collection of tools that you use to code, build, run, test, debug, and release software in the cloud. You can use AWS Cloud9 with the AWS SDK for Ruby to write and run your Ruby code by using a browser. AWS Cloud9 includes tools such as a code editor and terminal. Because the AWS Cloud9 IDE is cloud based, you can work on your projects from your office, home, or anywhere by using an internet-connected machine. For general information about AWS Cloud9, see the [AWS Cloud9 User Guide](#).

Follow these instructions to set up AWS Cloud9 with the AWS SDK for Ruby:

- [Step 1: Set up your AWS account to use AWS Cloud9](#)
- [Step 2: Set up your AWS Cloud9 development environment](#)

- [Step 3: Set up the AWS SDK for Ruby](#)
- [Step 4: Download example code](#)
- [Step 5: Run example code](#)

Step 1: Set up your AWS account to use AWS Cloud9

To use AWS Cloud9, sign in to the AWS Cloud9 console from the AWS Management Console.

Note

If you are using AWS IAM Identity Center to authenticate, you might need to add the required permission of `iam:ListInstanceProfilesForRole` to the user-attached policy in the IAM console.

To set up an IAM entity in your AWS account to access AWS Cloud9 and sign in to the AWS Cloud9 console, see [Team Setup for AWS Cloud9](#) in the *AWS Cloud9 User Guide*.

Step 2: Set up your AWS Cloud9 development environment

After you sign in to the AWS Cloud9 console, use the console to create an AWS Cloud9 development environment. After you create the environment, AWS Cloud9 opens the IDE for that environment.

For details, see [Creating an Environment in AWS Cloud9](#) in the *AWS Cloud9 User Guide*.

Note

As you create your environment in the console for the first time, we recommend that you choose the option to **Create a new instance for environment (EC2)**. This option tells AWS Cloud9 to create an environment, launch an Amazon EC2 instance, and then connect the new instance to the new environment. This is the fastest way to begin using AWS Cloud9.

If the terminal isn't already open in the IDE, open it. On the menu bar in the IDE, choose **Window, New Terminal**. You can use the terminal window to install tools and build your applications.

Step 3: Set up the AWS SDK for Ruby

After AWS Cloud9 opens the IDE for your development environment, use the terminal window to set up the AWS SDK for Ruby in your environment.

You can install the AWS SDK for Ruby as you would any Ruby gem. The gems are available at [RubyGems](#). The AWS SDK for Ruby is designed to be modular and is separated by AWS service. Installing the entire `aws-sdk` gem is large and may take over an hour.

We recommend only installing the gems for the AWS services you use. These are named like `aws-sdk-service_abbreviation` and the complete list is found in the [Supported Services](#) table of the AWS SDK for Ruby README file. For example, the gem for interfacing with the Amazon S3 service is directly available at [aws-sdk-s3](#).

Ruby version manager

Instead of using system Ruby, we recommend using a Ruby version manager such as the following:

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

For example, if you are using an Amazon Linux 2 operating system, the following commands can be used to update RVM, list the available Ruby versions, then choose the version you want to use for development with the AWS SDK for Ruby. The minimum required Ruby version is 2.3.

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

Bundler

If you use [Bundler](#), the following commands install the AWS SDK for Ruby gem for Amazon S3:

1. Install Bundler and create the Gemfile:

```
$ gem install bundler
```

```
$ bundle init
```

2. Open the created Gemfile and add a gem line for each AWS service gem your code will use. To follow along with the Amazon S3 example, add the following line to the bottom of the file:

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

3. Save the Gemfile.
4. Install the dependencies specified in your Gemfile:

```
$ bundle install
```

Step 4: Download example code

Use the terminal window to download example code for the AWS SDK for Ruby into the AWS Cloud9 development environment.

To download a copy of all the code examples used in the official AWS SDK documentation into your environment's root directory, run the following command:

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

The code examples for the AWS SDK for Ruby are located at `ENVIRONMENT_NAME/aws-doc-sdk-examples/ruby` directory, where `ENVIRONMENT_NAME` is the name of your development environment.

To follow along using an Amazon S3 example, we recommend starting with code example `ENVIRONMENT_NAME/aws-doc-sdk-examples/ruby/example_code/s3/bucket_list.rb`. Use the terminal window to navigate to the `s3` directory and list the files.

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

To open the file in AWS Cloud9, you can click on the `bucket_list.rb` directly in the terminal window.

For more support in understanding code examples, see [AWS SDK for Ruby Code Examples](#).

Step 5: Run example code

To run code in your AWS Cloud9 development environment, choose the **Run** button in the top menu bar. AWS Cloud9 will automatically detect the `.rb` file extension and use the Ruby runner to run the code. For more information about running code in AWS Cloud9, see [Run Your Code](#) in the *AWS Cloud9 User Guide*.

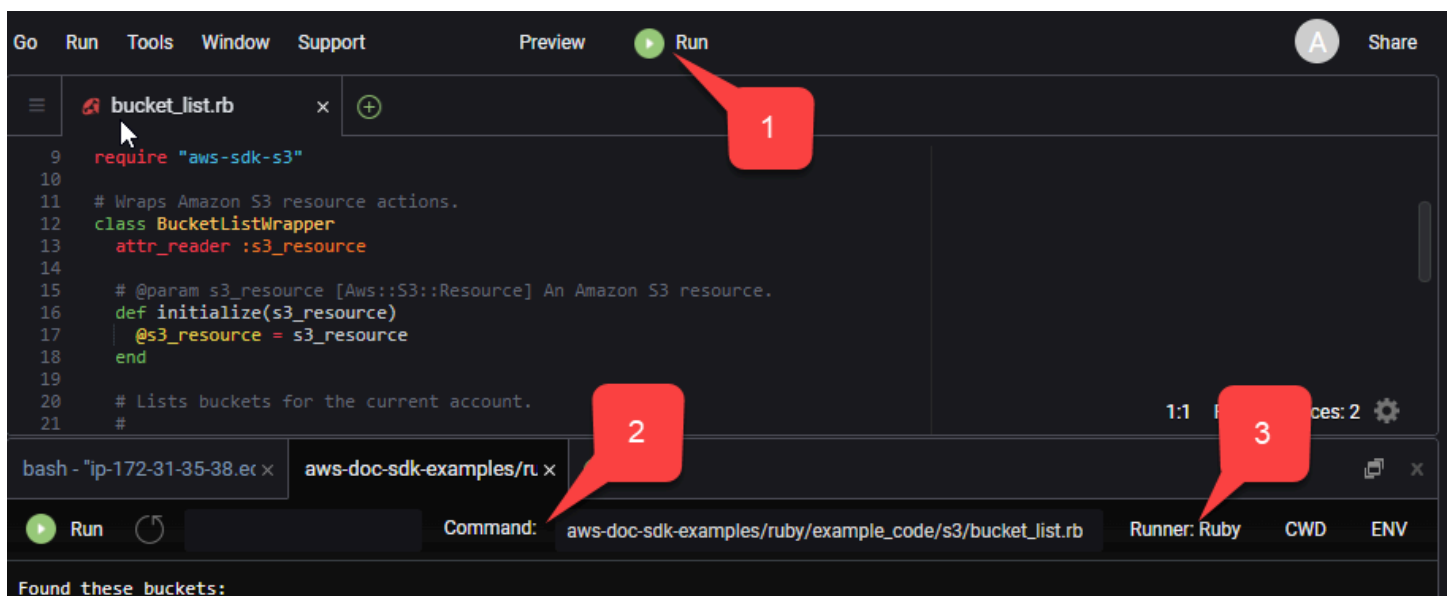
In the following screenshot, note these basic areas:

- 1: Run. The **Run** button is located on the top menu bar. This opens a new tab for your results.

Note

You can also manually create new run configurations. On the menu bar, choose **Run, Run Configurations, New Run Configuration**.

- 2: Command. AWS Cloud9 populates the **Command** text box with the path and file name to the file you run. If your code expects any command line parameters to be passed in, these can be added to the command line in the same way that you would when running the code through a terminal window.
- 3: Runner. AWS Cloud9 detects that your file extension is `.rb` and selects the Ruby Runner to run your code.



Any output generated from the running code displays in the tab.

Configure the AWS SDK for Ruby

Learn how to configure the AWS SDK for Ruby. You must establish how your code authenticates with AWS when you develop with AWS services. You must also set the AWS Region you want to use.

Credential provider chain

All SDKs have a series of places (or sources) that they check in order to get valid credentials to use to make a request to an AWS service. After valid credentials are found, the search is stopped. This systematic search is called the default credential provider chain.

For each step in the chain, there are different ways to set the values. Setting values directly in code always takes precedence, followed by setting as environment variables, and then in the shared AWS config file. For more information, see [Precedence of settings](#) in the *AWS SDKs and Tools Reference Guide*.

The *AWS SDKs and Tools Reference Guide* has information on SDK configuration settings used by all AWS SDKs and the AWS CLI. To learn more about how to configure the SDK through the shared AWS config file, see [Shared config and credentials files](#). To learn more about how to configure the SDK through setting environment variables, see [Environment variables support](#).

To authenticate with AWS, the AWS SDK for Ruby checks the credential providers in the order listed in the following table.

Credential provider by precedence	<i>AWS SDKs and Tools Reference Guide</i>	<i>AWS SDK for Ruby API Reference</i>
Static credentials	AWS access keys	Aws::Credentials Aws::SharedCredentials
Web identity token from AWS Security Token Service (AWS STS)	Assume role credential provider Using <code>role_arn</code> , <code>role_session_name</code> , and <code>web_identity_token_file</code>	Aws::AssumeRoleWebIdentityCredentials

Credential provider by precedence	<i>AWS SDKs and Tools Reference Guide</i>	<i>AWS SDK for Ruby API Reference</i>
AWS IAM Identity Center. In this guide, see SDK authentication with AWS .	IAM Identity Center credential provider	Aws::SSOCredentials
Trusted entity provider (such as <code>AWS_ROLE_ARN</code>). In this guide, see Creating an AWS STS access token .	Assume role credential provider Using <code>role_arn</code> and <code>role_session_name</code>	Aws::AssumeRoleCredentials
Process credential provider	Process credential provider	Aws::ProcessCredentials
Amazon Elastic Container Service (Amazon ECS) credentials	Container credential provider	Aws::ECSCredentials
Amazon Elastic Compute Cloud (Amazon EC2) instance profile credentials (IMDS credential provider)	IMDS credential provider	Aws::InstanceProfileCredentials

If the AWS SDK for Ruby environment variable `AWS_SDK_CONFIG_OPT_OUT` is set, the shared AWS config file, typically at `~/.aws/config`, will not be parsed for credentials.

If you followed the recommended approach for new users to get started, you set up AWS IAM Identity Center authentication during [SDK authentication with AWS](#) of the Getting started topic. Other authentication methods are useful for different situations. To avoid security risks, we recommend always using short-term credentials. For other authentication method procedures, see [Authentication and access](#) in the *AWS SDKs and Tools Reference Guide*.

Creating an AWS STS access token

Assuming a role involves using a set of temporary security credentials that you can use to access AWS resources that you might not normally have access to. These temporary

credentials consist of an access key ID, a secret access key, and a security token. You can use the [Aws::AssumeRoleCredentials](#) method to create an AWS Security Token Service (AWS STS) access token.

The following example uses an access token to create an Amazon S3 client object, where `linked::account::arn` is the Amazon Resource Name (ARN) of the role to assume and `session-name` is an identifier for the assumed role session.

```
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)
```

For more information about setting `role_arn` or `role_session_name`, or about setting these using the shared AWS config file instead, see [Assume role credential provider](#) in the *AWS SDKs and Tools Reference Guide*.

Setting a Region

You need to set a Region when using most AWS services. The AWS SDK for Ruby searches for a Region in the following order:

1. [Setting the Region in a client or resource object](#)
2. [Setting the Region by using `Aws.config`](#)
3. [Setting the Region by using environment variables](#)
4. [Setting the Region by using the shared config file](#)

For more information on the `region` setting, see [AWS Region](#) in the *AWS SDKs and Tools Reference Guide*. The rest of this section describes how to set a Region, starting with the most common approach.

Setting the Region using the shared config file

Set the region by setting the `region` variable in the shared AWS config file. For more information about the shared config file, see [Shared config and credentials files](#) in the *AWS SDKs and Tools Reference Guide*.

Example of setting this value in the config file:

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

The shared config file is not checked if the environment variable `AWS_SDK_CONFIG_OPT_OUT` is set.

Setting the Region using environment variables

Set the Region by setting the `AWS_REGION` environment variable.

Use the `export` command to set this variable on Unix-based systems, such as Linux or macOS. The following example sets the Region to `us-west-2`.

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

To set this variable on Windows, use the `set` command. The following example sets the Region to `us-west-2`.

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

Setting the Region with `Aws.config`

Set the Region by adding a `region` value to the `Aws.config` hash. The following example updates the `Aws.config` hash to use the `us-west-1` Region.

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

Any clients or resources that you create later are bound to this Region.

Setting the Region in a client or resource object

Set the Region when you create an AWS client or resource. The following example creates an Amazon S3 resource object in the `us-west-1` Region. Choose the correct Region for your AWS resources. A service client object is immutable, so you must create a new client for each service to which you make requests and for making requests to the same service using a different configuration.

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

Setting a nonstandard endpoint

The region is used to construct an SSL endpoint to use for AWS requests. If you need to use a nonstandard endpoint in the Region you've selected, add an `endpoint` entry to `Aws.config`. Alternatively, set the `endpoint:` when creating a service client or resource object. The following example creates an Amazon S3 resource object in the `other_endpoint` endpoint.

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

To use an endpoint of your choosing for API requests and to have that choice persist, see the [Service-specific endpoints](#) configuration option in the *AWS SDKs and Tools Reference Guide*.

Use the AWS SDK for Ruby

This section provides information about developing software with the AWS SDK for Ruby, including how to use some of the SDK's advanced features.

The [AWS SDKs and Tools Reference Guide](#) also contains settings, features, and other foundational concepts common among many of the AWS SDKs.

Topics

- [Use the AWS SDK for Ruby REPL utility](#)
- [Use the SDK with Ruby on Rails](#)
- [Debugging tip: Get wire trace information from a client](#)
- [Stub client responses and errors](#)
- [Pagination](#)
- [Waiters](#)
- [Specify client retry behavior](#)
- [Migrate from version 1 or 2 to version 3 of the AWS SDK for Ruby](#)

Use the AWS SDK for Ruby REPL utility

The `aws-sdk` gem includes a Read-Eval-Print-Loop (REPL) interactive command-line interface where you can test the SDK for Ruby and immediately see the results. SDK for Ruby gems are available at RubyGems.org.

Prerequisites

- [Install the AWS SDK for Ruby](#).
- The `aws-v3.rb` is located in the `aws-sdk-resources` gem. The `aws-sdk-resources` gem is also included by the main `aws-sdk` gem.
- You will need an xml library, such as the `rexml` gem.
- Although the program does work with the Interactive Ruby Shell (`irb`), we recommend that you install the `pry` gem, which provides a more powerful REPL environment.

Bundler setup

If you use [Bundler](#), the following updates to your Gemfile will address the prerequisite gems:

1. Open your Gemfile that you created when you installed the AWS SDK for Ruby. Add the following lines to the file:

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

2. Save the Gemfile.
3. Install the dependencies specified in your Gemfile:

```
$ bundle install
```

Running REPL

You can access the REPL by running `aws-v3.rb` from the command line.

```
aws-v3.rb
```

Alternatively, you can enable HTTP wire logging by setting the verbose flag. HTTP wire logging provides information about the communication between the AWS SDK for Ruby and AWS. Note, the verbose flag also adds overhead that can make your code run slower.

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

The SDK for Ruby includes client classes that provide interfaces to the AWS services. Each client class supports a particular AWS service. In the REPL, every service class has a helper that returns a new client object for interacting with that service. The name of the helper will be the name of the service converted to lower case. For example, the names of the Amazon S3 and Amazon EC2 helper objects are `s3` and `ec2`, respectively. To list the Amazon S3 buckets in your account, you can enter `s3.list_buckets` into the prompt.

You can type `quit` into the REPL prompt to exit.

Use the SDK with Ruby on Rails

[Ruby on Rails](#) provides a web development framework that makes it easy to create websites with Ruby.

AWS provides the `aws-sdk-rails` gem to enable easy integration with Rails. You can use AWS Elastic Beanstalk, AWS OpsWorks, AWS CodeDeploy, or the [AWS Rails Provisioner](#) to deploy and run your Rails applications in the AWS Cloud.

For information on installing and using the `aws-sdk-rails` gem, see the GitHub repository <https://github.com/aws/aws-sdk-rails>.

Debugging tip: Get wire trace information from a client

You can get wire trace information from an AWS client by setting the `http_wire_trace` Boolean. Wire trace information helps differentiate client changes, service issues, and user errors. When `true`, the setting shows what is being sent on the wire. The following example creates an Amazon S3 client with wire tracing enabled at the time of client creation.

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

Given the following code and the argument `bucket_name`, the output displays a message that says whether a bucket with that name exists.

```
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
```

If the bucket exists, the output is similar to the following. (Returns were added to the HEAD line for readability.)

```
opening connection to bucket_name.s3-us-west-1.amazonaws.com:443...
opened
```

```

starting SSL for bucket_name.s3-us-west-1.amazonaws.com:443...
SSL established, protocol: TLSv1.2, cipher: ECDHE-RSA-AES128-GCM-SHA256
-> "HEAD / HTTP/1.1
    Accept-Encoding:
    User-Agent: aws-sdk-ruby3/3.171.0 ruby/3.2.2 x86_64-linux aws-sdk-s3/1.120.0
    Host: bucket_name.s3-us-west-1.amazonaws.com
    X-Amz-Date: 20230427T143146Z
/* omitted */
Accept: */*\r\n\r\n"
-> "HTTP/1.1 200 OK\r\n"
-> "x-amz-id-2: XxB2J+kpHgTjmMUwpkUI1EjaFSPxAjWRgkn/+z7YwWc/
iAX5E30XRBzJ37cfc8T4D7ELC1KFELM=\r\n"
-> "x-amz-request-id: 5MD4APQOS815QVBR\r\n"
-> "Date: Thu, 27 Apr 2023 14:31:47 GMT\r\n"
-> "x-amz-bucket-region: us-east-1\r\n"
-> "x-amz-access-point-alias: false\r\n"
-> "Content-Type: application/xml\r\n"
-> "Server: AmazonS3\r\n"
-> "\r\n"
Conn keep-alive
Bucket bucket_name exists

```

You can also turn on wire tracing after client creation.

```

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

```

For more information on the fields in the wire trace information reported, see [Transfer Family required request headers](#).

Stub client responses and errors

Learn how to stub client responses and client errors in an AWS SDK for Ruby application.

Stubbing client responses

When you stub a response, the AWS SDK for Ruby disables network traffic and the client returns stubbed (or fake) data. If you don't supply stubbed data, the client returns:

- Lists as empty arrays
- Maps as empty hashes

- Numeric values as zero
- Dates as now

The following example returns stubbed names for the list of Amazon S3 buckets.

```
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
```

Running this code displays the following.

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

Note

After you supply any stubbed data, the default values no longer apply for any remaining instance attributes. This means that in the previous example, the remaining instance attribute, `creation_date`, is not now but `nil`.

The AWS SDK for Ruby validates your stubbed data. If you pass in data of the wrong type, it raises an `ArgumentError` exception. For example, if instead of the previous assignment to `bucket_data`, you used the following:

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

The AWS SDK for Ruby raises two `ArgumentError` exceptions.

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

Stubbing client errors

You can also stub errors that the AWS SDK for Ruby raises for specific methods. The following example displays 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
```

Pagination

Some AWS calls provide paged responses to limit the amount of data returned with each response. A page of data represents up to 1,000 items.

Paged responses are enumerable

The simplest way to handle paged response data is to use the built-in enumerator in the response object, as shown in the following example.

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

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

This yields one response object per API call made, and enumerates objects in the named bucket. The SDK retrieves additional pages of data to complete the request.

Handling paged responses manually

To handle paging yourself, use the response's `next_page?` method to verify there are more pages to retrieve, or use the `last_page?` method to verify there are no more pages to retrieve.

If there are more pages, use the `next_page` (notice there is no `?`) method to retrieve the next page of results, as shown in the following example.

```
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

If you call the `next_page` method and there are no more pages to retrieve, the SDK raises an [Aws::PageableResponse::LastPageError](#) exception.

Paged data classes

Paged data in the AWS SDK for Ruby is handled by the [Aws::PageableResponse](#) class, which is included with [Seahorse::Client::Response](#) to provide access to paged data.

Waiters

Waiters are utility methods that poll for a particular state to occur on a client. Waiters can fail after a number of attempts at a polling interval defined for the service client. For an example of how a waiter is used, see the [create_table](#) method of the Amazon DynamoDB Encryption Client in the [AWS Code Examples Repository](#).

Invoking a waiter

To invoke a waiter, call `wait_until` on a service client. In the following example, a waiter waits until the instance `i-12345678` is running before continuing.

```
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
```

The first parameter is the waiter name, which is specific to the service client and indicates which operation is being waited for. The second parameter is a hash of parameters that are passed to the client method called by the waiter, which varies according to the waiter name.

For a list of operations that can be waited for and the client methods called for each operation, see the `waiter_names` and `wait_until` field documentation for the client you are using.

Wait failures

Waiters can fail with any of the following exceptions.

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

A failure state was encountered while waiting.

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

The specified waiter name is not defined for the client being used.

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

The number of attempts exceeded the waiter's `max_attempts` value.

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

An unexpected error occurred while waiting.

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

One of the wait states was exceeded or another failure occurred while waiting.

All of these errors—except `NoSuchWaiterError`—are based on `WaiterFailed`. To catch errors in a waiter, use `WaiterFailed`, as shown in the following example.

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

Configuring a waiter

Each waiter has a default polling interval and a maximum number of attempts it will make before returning control to your program. To set these values, use the `max_attempts` and `delay` parameters in your `wait_until` call. The following example waits for up to 25 seconds, polling every five seconds.

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

To disable wait failures, set the value of either of these parameters to `nil`.

Extending a waiter

To modify the behavior of waiters, you can register callbacks that are triggered before each polling attempt and before waiting.

The following example implements an exponential backoff in a waiter by doubling the amount of time to wait on every attempt.

```
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
```

The following example disables the maximum number of attempts, and instead waits for one hour (3600 seconds) before failing.

```
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
```

Specify client retry behavior

By default, the AWS SDK for Ruby performs up to three retries, with 15 seconds between retries, for a total of up to four attempts. Therefore, an operation could take up to 60 seconds to time out.

The following example creates an Amazon S3 client in the region `us-west-2`, and specifies to wait five seconds between two retries on every client operation. Therefore, Amazon S3 client operations could take up to 15 seconds to time out.

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

This example shows how to change the retry parameters directly within code. However, you can also use environment variables or the shared AWS config file to set these for your application. For more information on these settings, see [Retry behavior](#) in the *AWS SDKs and Tools Reference Guide*. Any explicit setting set in the code or on a service client itself takes precedence over those set in environment variables or the shared config file.

Migrate from version 1 or 2 to version 3 of the AWS SDK for Ruby

The purpose of this topic is to help you migrate from version 1 or 2 of the AWS SDK for Ruby to version 3.

Side-by-side usage

It isn't necessary to replace the version 1 or 2 of the AWS SDK for Ruby with version 3. You can use them together in the same application. See [this blog post](#) for more information.

A quick example follows.

```
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
```

You don't need to rewrite existing working version 1 or 2 code to start using the version 3 SDK. A valid migration strategy is to only write new code against the version 3 SDK.

General differences

Version 3 differs from version 2 in one important way.

- Each service is available as a separate gem.

Version 2 differs from version 1 in several important ways.

- Different root namespace –`Aws` versus `AWS`. This enables side-by-side usage.
- `Aws.config`– Now a vanilla Ruby hash, instead of a method.
- Strict constructor options - When constructing a client or resource object in the version 1 SDK, unknown constructor options are ignored. In version 2, unknown constructor options trigger an `ArgumentError`. For example:

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

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

Client differences

There are no differences between the client classes in version 2 and version 3.

Between version 1 and version 2, the client classes have the fewest external differences. Many service clients will have compatible interfaces after client construction. Some important differences:

- `Aws::S3::Client` - The version 1 Amazon S3 client class was hand-coded. Version 2 is generated from a service model. Method names and inputs are very different in version 2.
- `Aws::EC2::Client` - Version 2 uses plural names for output lists, version 1 uses the suffix `_set`. For example:

```
# 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` - Version 2 uses structured responses, where version 1 uses vanilla Ruby hashes.
- Service class renames - Version 2 uses a different name for multiple services:
 - `AWS::SimpleWorkflow` has become `Aws::SWF`
 - `AWS::ELB` has become `Aws::ElasticLoadBalancing`
 - `AWS::SimpleEmailService` has become `Aws::SES`
- Client configuration options - Some of the version 1 configuration options are renamed in version 2. Others are removed or replaced. Here are the primary changes:
 - `:use_ssl` has been removed. Version 2 uses SSL everywhere. To disable SSL you must configure an `:endpoint` that uses `http://`.
 - `:ssl_ca_file` is now `:ssl_ca_bundle`
 - `:ssl_ca_path` is now `:ssl_ca_directory`
 - Added `:ssl_ca_store`.
 - `:endpoint` must now be a fully qualified HTTP or HTTPS URI instead of a hostname.

- Removed `:_port` options for each service, now replaced by `:endpoint`.
- `:user_agent_prefix` is now `:user_agent_suffix`

Resource differences

There are no differences between the resource interfaces in version 2 and version 3.

There are significant differences between the resource interfaces in version 1 and version 2. Version 1 was entirely hand-coded, where as version 2 resource interfaces are generated from a model. Version 2 resource interfaces are significantly more consistent. Some of the systemic differences include:

- Separate resource class – In version 2, the service name is a module, not a class. In this module, it is the resource interface:

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

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

- Referencing resources – The version 2 SDK separates collections and individual resource getters into two different methods:

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

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

- Batch operations – In version 1, all batch operations were hand-coded utilities. In version 2, many batch operations are autogenerated batching operations over the API. **Version 2 batching interfaces are very different from version 1.**

Work with AWS services in the AWS SDK for Ruby

The following sections contain discussion and examples that show you how to use the AWS SDK for Ruby to work with AWS services.

If you're new to the AWS SDK for Ruby, you might want to read through the [Get started](#) topic first.

- [Code examples with guidance](#) – Provides guided examples for several AWS services.
- [Code examples](#)– Provides a complete list of available service examples (but without additional guidance beyond the code).

Source code for all these examples is available to download in the [AWS Code Examples Repository](#) on GitHub. To propose a new code example for the AWS documentation team to consider producing, create a new request. The team is looking to produce code examples that cover broader scenarios and use cases, versus simple code snippets that cover only individual API calls. For instructions, see the *Proposing new code examples* section in the [Readme on GitHub](#).

Code examples with guidance for the AWS SDK for Ruby

This section provides examples you can use to access AWS services by using the AWS SDK for Ruby.

Find the source code for these examples and others in the [AWS Code Examples Repository](#) on GitHub.

Topics

- [CloudTrail Examples Using the AWS SDK for Ruby](#)
- [Amazon CloudWatch Examples Using the AWS SDK for Ruby](#)
- [CodeBuild Examples Using the AWS SDK for Ruby](#)
- [Amazon EC2 Examples Using the AWS SDK for Ruby](#)
- [AWS Elastic Beanstalk Examples Using the AWS SDK for Ruby](#)
- [AWS Identity and Access Management \(IAM\) Examples Using the AWS SDK for Ruby](#)
- [AWS Key Management Service Examples Using the AWS SDK for Ruby](#)
- [AWS Lambda Examples Using the AWS SDK for Ruby](#)
- [Amazon Polly Examples Using the AWS SDK for Ruby](#)
- [Amazon RDS Examples Using the AWS SDK for Ruby](#)

- [Amazon SES Examples Using the AWS SDK for Ruby](#)
- [Amazon SNS Examples Using the AWS SDK for Ruby](#)
- [Amazon SQS Examples Using the AWS SDK for Ruby](#)
- [Amazon WorkDocs Examples](#)

CloudTrail Examples Using the AWS SDK for Ruby

CloudTrail is an AWS service that you can use to monitor your AWS deployments in the cloud by getting a history of AWS API calls for your account. You can use the following AWS SDK for Ruby code examples to access AWS CloudTrail. For more information about CloudTrail, see the [AWS CloudTrail; documentation](#).

Topics

- [Listing the CloudTrail Trails](#)
- [Creating a CloudTrail Trail](#)
- [Listing CloudTrail Trail Events](#)
- [Deleting a CloudTrail Trail](#)

Listing the CloudTrail Trails

This example uses the [describe_trails](#) method to list the names of the CloudTrail trails and the bucket in which CloudTrail stores information in the us-west-2 region.

Choose Copy to save the code locally.

Create the file *describe_trails.rb* with the following code.

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

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

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

resp = client.describe_trails({})

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

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

See the [complete example](#) on GitHub.

Creating a CloudTrail Trail

This example uses the [create_trail](#) method to create a CloudTrail trail in the us-west-2 region. It requires two inputs, the name of the trail and the name of the bucket in which CloudTrail stores information. If the bucket does not have the proper policy, include the **-p** flag to attach the correct policy to the bucket.

Choose Copy to save the code locally.

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

require 'aws-sdk-cloudtrail' # v2: require 'aws-sdk'
require 'aws-sdk-s3'
```



```
require 'aws-sdk-sts'

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

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

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

  s3_client.put_bucket_policy(
    bucket: bucket,
```

```
    policy: policy
  )

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

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

i = 0

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

    when '-p'
      attach_policy = true

    else
      name = ARGV[i]
    end

    i += 1
  end

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

  if attach_policy
    add_policy(bucket)
  end
end
```

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

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

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

See the [complete example](#) on GitHub.

Listing CloudTrail Trail Events

This example uses the [lookup_events](#) method to list the CloudTrail trail events in the us-west-2 region.

Choose Copy to save the code locally.

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

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

def show_event(event)
  puts 'Event name:   ' + event.event_name
  puts 'Event ID:     ' + event.event_id
  puts "Event time:    #{event.event_time}"
end
```

```
puts 'User name:      ' + event.username

puts 'Resources:'

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

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

resp = client.lookup_events()

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

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

See the [complete example](#) on GitHub.

Deleting a CloudTrail Trail

This example uses the [delete_trail](#) method to delete a CloudTrail trail in the us-west-2 region. It requires one input, the name of the trail.

Choose Copy to save the code locally.

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

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

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

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

name = ARGV[0]

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

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

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

See the [complete example](#) on GitHub.

Amazon CloudWatch Examples Using the AWS SDK for Ruby

Amazon CloudWatch (CloudWatch) is a monitoring service for AWS cloud resources and the applications you run on AWS. You can use the following examples to access CloudWatch by using the AWS SDK for Ruby. For more information about CloudWatch, see the [Amazon CloudWatch documentation](#).

Topics

- [Getting Information about Amazon CloudWatch Alarms](#)
- [Creating an Amazon CloudWatch Alarm](#)
- [Enabling and Disabling Amazon CloudWatch Alarm Actions](#)
- [Getting Information about Custom Metrics for Amazon CloudWatch](#)
- [Sending Events to Amazon CloudWatch Events](#)

Getting Information about Amazon CloudWatch Alarms

The following code example displays information about available metric alarms in Amazon CloudWatch.

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

require 'aws-sdk-cloudwatch'

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

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

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

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

```
    alarm.alarm_actions.each do |a|
      puts ' ' + a
    end
  end

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

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

# Full example call:
def run_me
  region = ''

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

Creating an Amazon CloudWatch Alarm

The following code example creates a new CloudWatch alarm (or updates an existing alarm, if an alarm with the specified name already exists).

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

require 'aws-sdk-cloudwatch'

# Creates or updates an alarm in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param alarm_name [String] The name of the alarm.
# @param alarm_description [String] A description about the alarm.
# @param metric_name [String] The name of the metric associated with the alarm.
# @param alarm_actions [Array] A list of Strings representing the
#   Amazon Resource Names (ARNs) to execute when the alarm transitions to the
#   ALARM state.
# @param namespace [String] The namespace for the metric to alarm on.
# @param statistic [String] The statistic for the metric.
# @param dimensions [Array] A list of dimensions for the metric, specified as
#   Aws::CloudWatch::Types::Dimension.
# @param period [Integer] The number of seconds before re-evaluating the metric.
# @param unit [String] The unit of measure for the statistic.
# @param evaluation_periods [Integer] The number of periods over which data is
#   compared to the specified threshold.
# @param 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

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

Enabling and Disabling Amazon CloudWatch Alarm Actions

The following code example:

1. Creates and enables a new CloudWatch alarm (or updates an existing alarm, if an alarm with the specified name already exists).
2. Disables the new or existing alarm. To enable the alarm again, call `enable_alarm_actions`.

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

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

require 'aws-sdk-cloudwatch'

# Creates or updates an alarm in Amazon CloudWatch.
#
```

```
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param alarm_name [String] The name of the alarm.
# @param alarm_description [String] A description about the alarm.
# @param metric_name [String] The name of the metric associated with the alarm.
# @param alarm_actions [Array] A list of Strings representing the
#   Amazon Resource Names (ARNs) to execute when the alarm transitions to the
#   ALARM state.
# @param namespace [String] The namespace for the metric to alarm on.
# @param statistic [String] The statistic for the metric.
# @param dimensions [Array] A list of dimensions for the metric, specified as
#   Aws::CloudWatch::Types::Dimension.
# @param period [Integer] The number of seconds before re-evaluating the metric.
# @param unit [String] The unit of measure for the statistic.
# @param evaluation_periods [Integer] The number of periods over which data is
#   compared to the specified threshold.
# @param 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

# Disables an alarm in Amazon CloudWatch.
#
# Prerequisites.
#
# - The alarm to disable.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
```

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

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

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

Getting Information about Custom Metrics for Amazon CloudWatch

The following code example:

1. Adds datapoints to a custom metric in CloudWatch.
2. Displays a list of available metrics for a metric namespace in CloudWatch.

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

# The following example shows how to:
# 1. Add a datapoint to a metric in Amazon CloudWatch.
```

```
# 2. List available metrics for a metric namespace in Amazon CloudWatch.

require 'aws-sdk-cloudwatch'

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



```

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

# Lists available metrics for a metric namespace in Amazon CloudWatch.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param metric_namespace [String] The namespace of the metric.
# @example
#   list_metrics_for_namespace(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'SITE/TRAFFIC'
#   )
def list_metrics_for_namespace(cloudwatch_client, metric_namespace)
  response = cloudwatch_client.list_metrics(namespace: metric_namespace)

  if response.metrics.count.positive?
    response.metrics.each do |metric|
      puts " Metric name: #{metric.metric_name}"
      if metric.dimensions.count.positive?
        puts '   Dimensions:'
        metric.dimensions.each do |dimension|
          puts "     Name: #{dimension.name}, Value: #{dimension.value}"
        end
      else
        puts 'No dimensions found.'
      end
    end
  end
end
else

```

```
puts "No metrics found for namespace '#{metric_namespace}'. " \
     'Note that it could take up to 15 minutes for recently-added metrics ' \
     'to become available.'
end
end

# Full example call:
def run_me
  metric_namespace = 'SITE/TRAFFIC'
  region = 'us-east-1'

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

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

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

Sending Events to Amazon CloudWatch Events

The following code example shows how to create and trigger a rule in Amazon CloudWatch Events. This rule sends a notification to the specified topic in Amazon Simple Notification Service (Amazon SNS) whenever an available instance in Amazon Elastic Compute Cloud (Amazon EC2) changes to a running state. Also, related event information is logged to a log group in CloudWatch Events.

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

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

```
# preceding functions in the correct order.

require 'aws-sdk-sns'
require 'aws-sdk-iam'
require 'aws-sdk-cloudwatchevents'
require 'aws-sdk-ec2'
require 'aws-sdk-cloudwatch'
require 'aws-sdk-cloudwatchlogs'
require 'securerandom'

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

# Checks whether the specified topic exists among those available to the
# caller in Amazon Simple Notification Service (Amazon SNS).
#
# @param sns_client [Aws::SNS::Client] An initialized Amazon SNS client.
# @param topic_arn [String] The Amazon Resource Name (ARN) of the
#   topic to find.
# @return [Boolean] true if the topic ARN was found; otherwise, false.
# @example
#   exit 1 unless topic_exists?(
#     Aws::SNS::Client.new(region: 'us-east-1'),
```

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

# Creates a topic in Amazon Simple Notification Service (Amazon SNS)
# and then subscribes an email address to receive notifications to that topic.
#
# @param sns_client [Aws::SNS::Client] An initialized Amazon SNS client.
# @param topic_name [String] The name of the topic to create.
# @param email_address [String] The email address of the recipient to notify.
# @return [String] The Amazon Resource Name (ARN) of the topic that
#   was created.
# @example
#   puts create_topic(
#     Aws::SNS::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-topic',
#     'mary@example.com'
#   )
def create_topic(sns_client, topic_name, email_address)
  puts "Creating the topic named '#{topic_name}'..."
  topic_response = sns_client.create_topic(name: topic_name)
```

```

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

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

# Checks whether the specified role exists among those available to the
# caller in AWS Identity and Access Management (IAM).
#

```

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

# Creates a role in AWS Identity and Access Management (IAM).
# This role is used by a rule in Amazon CloudWatch Events to allow
# that rule to operate within the caller's account.
# This role is designed to be used specifically by this code example.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.
# @param role_name [String] The name of the role to create.
# @return [String] The Amazon Resource Name (ARN) of the role that
#   was created.
# @example
```

```
# puts create_role(
#   Aws::IAM::Client.new(region: 'us-east-1'),
#   'aws-doc-sdk-examples-ec2-state-change'
# )
def create_role(iam_client, role_name)
  puts "Creating the role named '#{role_name}'..."
  response = iam_client.create_role(
    assume_role_policy_document: {
      'Version': '2012-10-17',
      'Statement': [
        {
          'Sid': '',
          'Effect': 'Allow',
          'Principal': {
            'Service': 'events.amazonaws.com'
          },
          'Action': 'sts:AssumeRole'
        }
      ]
    }.to_json,
    path: '/',
    role_name: role_name
  )
  puts "Role created with ARN '#{response.role.arn}'."
  puts 'Adding access policy to role...'
  iam_client.put_role_policy(
    policy_document: {
      'Version': '2012-10-17',
      'Statement': [
        {
          'Sid': 'CloudWatchEventsFullAccess',
          'Effect': 'Allow',
          'Resource': '*',
          'Action': 'events:*'
        },
        {
          'Sid': 'IAMPassRoleForCloudWatchEvents',
          'Effect': 'Allow',
          'Resource': 'arn:aws:iam::*:role/AWS_Events_Invoke_Targets',
          'Action': 'iam:PassRole'
        }
      ]
    }.to_json,
    policy_name: 'CloudWatchEventsPolicy',
```



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

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

# Checks whether the specified rule exists among those available to the
# caller in AWS CloudWatch Events.
#
# @param cloudwatchevents_client [Aws::CloudWatchEvents::Client]
#   An initialized AWS CloudWatch Events client.
# @param rule_name [String] The name of the rule to find.
# @return [Boolean] true if the rule name was found; otherwise, false.
# @example
#   exit 1 unless rule_exists?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1')
#     'aws-doc-sdk-examples-ec2-state-change'
#   )
def rule_exists?(cloudwatchevents_client, rule_name)
```

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

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

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

  put_targets_response = cloudwatchevents_client.put_targets(
```

```

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

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

```

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

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

# Writes an event to a log stream in Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - A log group in Amazon CloudWatch Logs.
# - A log stream within the log group.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group.
```

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

  response = cloudwatchlogs_client.put_log_events(event)
  puts 'Message logged.'
  return response.next_sequence_token
rescue StandardError => e
```

```

    puts "Message not logged: #{e.message}"
  end

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

    puts "Attempting to stop the instance with the ID '#{instance_id}'. " \
      'This might take a few minutes...'
    ec2_client.stop_instances(instance_ids: [instance_id])
  end
end

```

```
ec2_client.wait_until(:instance_stopped, instance_ids: [instance_id])
puts 'Instance stopped.'
sequence_token = log_event(
  cloudwatchlogs_client,
  log_group_name,
  log_stream_name,
  "Instance '#{instance_id}' stopped.",
  sequence_token
)

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

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

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



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

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

```

```

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

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

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

# Full example call:
def run_me
  # Properties for the Amazon SNS topic.

```

```
topic_name = 'aws-doc-sdk-examples-topic'
email_address = 'mary@example.com'
# Properties for the IAM role.
role_name = 'aws-doc-sdk-examples-cloudwatch-events-rule-role'
# Properties for the Amazon CloudWatch Events rule.
rule_name = 'aws-doc-sdk-examples-ec2-state-change'
rule_description = 'Triggers when any available EC2 instance starts.'
instance_state = 'running'
target_id = 'sns-topic'
# Properties for the Amazon EC2 instance.
instance_id = 'i-033c48ef067af3dEX'
# Properties for displaying the event rule's activity.
start_time = Time.now - 600 # Go back over the past 10 minutes
                        # (10 minutes * 60 seconds = 600 seconds).

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

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

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

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

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

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

# Restart the Amazon EC2 instance, which triggers the rule.
```

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

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

CodeBuild Examples Using the AWS SDK for Ruby

CodeBuild is a fully managed build service that compiles source code, runs tests, and produces software packages that are ready to deploy. You can use the following AWS SDK for Ruby code examples to access AWS CodeBuild. For more information about CodeBuild, see the [AWS CodeBuild documentation](#).

Topics

- [Getting Information about All AWS CodeBuild Projects](#)
- [Building an AWS CodeBuild Project](#)

- [Listing AWS CodeBuild Project Builds](#)

Getting Information about All AWS CodeBuild Projects

The following example lists the names of up to 100 of your AWS CodeBuild projects.

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

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

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

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

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

puts
```

Choose Copy to save the code locally. See the [complete example](#) on GitHub.

Building an AWS CodeBuild Project

The following example builds the AWS CodeBuild project specified on the command line. If no command line argument is supplied, it emits an error and quits.

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

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

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

project_name = ''

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

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

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

Choose Copy to save the code locally. See the [complete example](#) on GitHub.

Listing AWS CodeBuild Project Builds

The following example displays information about your AWS CodeBuild project builds. This information includes the name of the project, when the build started, and how long each phase of the build took, in seconds.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
```



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

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

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

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

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

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

Choose Copy to save the code locally. See the [complete example](#) on GitHub.

Amazon EC2 Examples Using the AWS SDK for Ruby

Amazon Elastic Compute Cloud (Amazon EC2) is a web service that provides resizable computing capacity—literally servers in Amazon’s data centers—that you use to build and host your software systems. You can use the following examples to access Amazon EC2 using the AWS SDK for Ruby. For more information about Amazon EC2, see the [Amazon EC2 Documentation](#).

Topics

- [Creating an Amazon EC2 VPC](#)
- [Creating an Internet Gateway and Attaching It to a VPC in Amazon EC2](#)
- [Creating a Public Subnet for Amazon EC2](#)
- [Creating an Amazon EC2 Route Table and Associating It with a Subnet](#)
- [Using Elastic IP Addresses in Amazon EC2](#)
- [Creating an Amazon EC2 Security Group](#)
- [Working with Amazon EC2 Security Groups](#)
- [Working with Key Pairs in Amazon EC2](#)
- [Getting Information about All Amazon EC2 Instances](#)

- [Getting Information about All Amazon EC2 Instances with a Specific Tag Value](#)
- [Getting Information about a Specific Amazon EC2 Instance](#)
- [Creating an Amazon EC2 Instance](#)
- [Stopping an Amazon EC2 Instance](#)
- [Starting an Amazon EC2 Instance](#)
- [Rebooting an Amazon EC2 Instance](#)
- [Managing Amazon EC2 Instances](#)
- [Terminating an Amazon EC2 Instance](#)
- [Getting Information about Regions and Availability Zones for Amazon EC2](#)

Creating an Amazon EC2 VPC

The following code example creates a virtual private cloud (VPC) in Amazon Virtual Private Cloud (Amazon VPC) and then tags the VPC.

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

require 'aws-sdk-ec2'

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

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

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

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

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

# Full example call:
def run_me
  cidr_block = ''
  tag_key = ''
  tag_value = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:  ruby ec2-ruby-example-create-vpc.rb ' \
      'CIDR_BLOCK TAG_KEY TAG_VALUE REGION'
    puts 'Example: ruby ec2-ruby-example-create-vpc.rb ' \
      '10.0.0.0/24 my-key my-value us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    cidr_block = '10.0.0.0/24'
    tag_key = 'my-key'
    tag_value = 'my-value'
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
    cidr_block = ARGV[0]
    tag_key = ARGV[1]
```

```
    tag_value = ARGV[2]
    region = ARGV[3]
end

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

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

run_me if $PROGRAM_NAME == __FILE__
```

Creating an Internet Gateway and Attaching It to a VPC in Amazon EC2

The following code example creates an internet gateway and then attaches it to a virtual private cloud (VPC) in Amazon Virtual Private Cloud (Amazon VPC).

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

require 'aws-sdk-ec2'

# Creates an internet gateway and then attaches it to a virtual private cloud
# (VPC) in Amazon Virtual Private Cloud (Amazon VPC).
#
# Prerequisites:
#
# - A VPC in Amazon VPC.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
#   Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param vpc_id [String] The ID of the VPC to attach the internet gateway.
# @param tag_key [String] The key of the tag to attach to the internet gateway.
# @param tag_value [String] The value of the tag to attach to the
#   internet gateway.
```

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

# Full example call:
def run_me
  vpc_id = ''
  tag_key = ''
  tag_value = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage: ruby ec2-ruby-example-attach-igw-vpc.rb ' \
      'VPC_ID TAG_KEY TAG_VALUE REGION'
    puts 'Example: ruby ec2-ruby-example-attach-igw-vpc.rb ' \
      'vpc-6713dfEX my-key my-value us-east-1'
```

```
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    vpc_id = 'vpc-6713dfEX'
    tag_key = 'my-key'
    tag_value = 'my-value'
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
    vpc_id = ARGV[0]
    tag_key = ARGV[1]
    tag_value = ARGV[2]
    region = ARGV[3]
  end

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

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

run_me if $PROGRAM_NAME == __FILE__
```

Creating a Public Subnet for Amazon EC2

The following code example creates a subnet within a virtual private cloud (VPC) in Amazon Virtual Private Cloud (Amazon VPC) and then tags the subnet.

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

require 'aws-sdk-ec2'

# Creates a subnet within a virtual private cloud (VPC) in
# Amazon Virtual Private Cloud (Amazon VPC) and then tags
```

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

```

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

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

Creating an Amazon EC2 Route Table and Associating It with a Subnet

The following code example creates a route table in Amazon Virtual Private Cloud (Amazon VPC) and then associates the route table with a subnet in Amazon VPC.

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

require 'aws-sdk-ec2'

# Creates a route table in Amazon Virtual Private Cloud (Amazon VPC)
# and then associates the route table with a subnet in Amazon VPC.
#
# Prerequisites:
#
# - A VPC in Amazon VPC.
# - A subnet in that VPC.
# - A gateway attached to that subnet.
#
# @param ec2_resource [Aws::EC2::Resource] An initialized
# Amazon Elastic Compute Cloud (Amazon EC2) resource object.
# @param vpc_id [String] The ID of the VPC for the route table.
# @param subnet_id [String] The ID of the subnet for the route table.
# @param gateway_id [String] The ID of the gateway for the route.
```

```

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

```

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

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

```
    region = ARGV[6]
  end

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

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

run_me if $PROGRAM_NAME == __FILE__
```

Using Elastic IP Addresses in Amazon EC2

The following code example:

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

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

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

require 'aws-sdk-ec2'

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

# Creates an Elastic IP address in Amazon Virtual Private Cloud (Amazon VPC).
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @return [String] The allocation ID corresponding to the Elastic IP address.
# @example
#   puts allocate_elastic_ip_address(Aws::EC2::Client.new(region: 'us-east-1'))
def allocate_elastic_ip_address(ec2_client)
  response = ec2_client.allocate_address(domain: 'vpc')
  return response.allocation_id
end
```

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

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

# Gets information about addresses associated with an
# Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
```

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

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

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

# Full example call:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:  ruby ec2-ruby-example-elastic-ips.rb ' \
      'INSTANCE_ID REGION'
    puts 'Example: ruby ec2-ruby-example-elastic-ips.rb ' \
      'i-033c48ef067af3dEX us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    instance_id = 'i-033c48ef067af3dEX'
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

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

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

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



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

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

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

Creating an Amazon EC2 Security Group

The following code example creates an Amazon EC2 security group and then adds an outbound rule to that security group.

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

require 'aws-sdk-ec2'

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

```

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

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

```

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

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

Working with Amazon EC2 Security Groups

The following example:

1. Creates an Amazon EC2 security group.
2. Adds inbound rules to the security group.
3. Displays information about available security groups.
4. Deletes the security group.

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

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

require 'aws-sdk-ec2'

# Creates an Amazon Elastic Compute Cloud (Amazon EC2) security group.
#
# Prerequisites:
#
# - A VPC in Amazon Virtual Private Cloud (Amazon VPC).
#
# @param ec2_client [Aws::EC2::Client] An initialized
#   Amazon EC2 client.
# @param group_name [String] A name for the security group.
# @param description [String] A description for the security group.
# @param vpc_id [String] The ID of the VPC for the security group.
# @return [String] The ID of security group that was created.
```

```

# @example
#   puts create_security_group(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'my-security-group',
#     'This is my security group.',
#     'vpc-6713dfEX'
#   )
def create_security_group(
  ec2_client,
  group_name,
  description,
  vpc_id
)
  security_group = ec2_client.create_security_group(
    group_name: group_name,
    description: description,
    vpc_id: vpc_id
  )
  puts "Created security group '#{group_name}' with ID " \
    "'#{security_group.group_id}' in VPC with ID '#{vpc_id}'."
  return security_group.group_id
rescue StandardError => e
  puts "Error creating security group: #{e.message}"
  return 'Error'
end

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

```

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

# Displays information about a security group's IP permissions set in
# Amazon Elastic Compute Cloud (Amazon EC2).
#
# Prerequisites:
#
# - A security group with inbound rules, outbound rules, or both.
#
```

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

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

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

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

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

  print "\n"
end

# Displays information about available security groups in
# Amazon Elastic Compute Cloud (Amazon EC2).
#
# @param ec2_client [Aws::EC2::Client] An initialized Amazon EC2 client.
# @example
```



```
# describe_security_groups(Aws::EC2::Client.new(region: 'us-east-1'))
def describe_security_groups(ec2_client)
  response = ec2_client.describe_security_groups

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

      if sg.tags.count.positive?
        puts 'Tags:'
        sg.tags.each do |tag|
          puts "  Key: #{tag.key}, Value: #{tag.value}"
        end
      end
    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_egress.count.positive?
      sg.ip_permissions_egress.each do |p|
        describe_security_group_permissions(p)
      end
    end
  end
else
  puts 'No security groups found.'
end
rescue StandardError => e
  puts "Error getting information about security groups: #{e.message}"
end

# Deletes an Amazon Elastic Compute Cloud (Amazon EC2)
# security group.
#
```

```

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

# Full example call:
def run_me
  group_name = ''
  description = ''
  vpc_id = ''
  ip_protocol_http = ''
  from_port_http = ''
  to_port_http = ''
  cidr_ip_range_http = ''
  ip_protocol_ssh = ''
  from_port_ssh = ''
  to_port_ssh = ''
  cidr_ip_range_ssh = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:  ruby ec2-ruby-example-security-group.rb ' \
      'GROUP_NAME DESCRIPTION VPC_ID IP_PROTOCOL_1 FROM_PORT_1 TO_PORT_1 ' \
      'CIDR_IP_RANGE_1 IP_PROTOCOL_2 FROM_PORT_2 TO_PORT_2 ' \
      'CIDR_IP_RANGE_2 REGION'
    puts 'Example: ruby ec2-ruby-example-security-group.rb ' \
      'my-security-group \'This is my security group.\' vpc-6713dfEX ' \

```

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

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

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

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

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

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

Working with Key Pairs in Amazon EC2

The following code example:

1. Creates a key pair in Amazon EC2.
2. Displays information about available key pairs.
3. Deletes the key pair.

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

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

require 'aws-sdk-ec2'

# Creates a key pair in Amazon Elastic Compute Cloud (Amazon EC2) and
# saves the resulting RSA private key file locally in the calling
# user's home directory.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param key_pair_name [String] The name for the key pair and private
#   key file.
# @return [Boolean] true if the key pair and private key file were
#   created; otherwise, false.
# @example
#   exit 1 unless key_pair_created?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'my-key-pair'
#   )
def key_pair_created?(ec2_client, key_pair_name)
  key_pair = ec2_client.create_key_pair(key_name: key_pair_name)
  puts "Created key pair '#{key_pair.key_name}' with fingerprint " \
    "'#{key_pair.key_fingerprint}' and ID '#{key_pair.key_pair_id}'."
  filename = File.join(Dir.home, key_pair_name + '.pem')
  File.open(filename, 'w') { |file| file.write(key_pair.key_material) }
  puts "Private key file saved locally as '#{filename}'."
end
```

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

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

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

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

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

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

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

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

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

  puts '-' * 10
end
```

```

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__

```

Getting Information about All Amazon EC2 Instances

The following code example lists the IDs and current states of available Amazon EC2 instances.

```

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

require 'aws-sdk-ec2'

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

```



```

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

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

run_me if $PROGRAM_NAME == __FILE__

```

Getting Information about All Amazon EC2 Instances with a Specific Tag Value

The following code example lists the IDs and current states of available Amazon EC2 instances matching the specified tag key and value.

```

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

require 'aws-sdk-ec2'

# Lists the IDs, current states, and tag keys/values of matching

```

```

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

#Full example call:
def run_me
  tag_key = ''
  tag_value = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:  ruby ec2-ruby-example-get-instance-info-by-tag.rb ' \

```

```

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

run_me if $PROGRAM_NAME == __FILE__

```

Getting Information about a Specific Amazon EC2 Instance

The following example lists the state of the specified Amazon EC2 instance.

```

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

require 'aws-sdk-ec2'

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

```

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

# Full example call:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:  ruby ec2-ruby-example-list-state-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION'
    puts 'Example: ruby ec2-ruby-example-list-state-instance-i-123abc.rb ' \
      'i-123abc us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)
  list_instance_state(ec2_client, instance_id)
end

run_me if $PROGRAM_NAME == __FILE__
```

Creating an Amazon EC2 Instance

The following example creates and tags an Amazon EC2 instance.

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

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

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

```
    tag_key,  
    tag_value,  
    instance_type = 't2.micro',  
    user_data_file = ''  
  )  
  encoded_script = ''  
  
  unless user_data_file == ''  
    script = File.read(user_data_file)  
    encoded_script = Base64.encode64(script)  
  end  
  
  instance = ec2_resource.create_instances(  
    image_id: image_id,  
    min_count: 1,  
    max_count: 1,  
    key_name: key_pair_name,  
    instance_type: instance_type,  
    user_data: encoded_script  
  )  
  
  puts 'Creating instance...'  
  
  # Check whether the new instance is in the "running" state.  
  polls = 0  
  loop do  
    polls += 1  
    response = ec2_resource.client.describe_instances(  
      instance_ids: [  
        instance.first.id  
      ]  
    )  
    # Stop polling after 10 minutes (40 polls * 15 seconds per poll) if not running.  
    break if response.reservations[0].instances[0].state.name == 'running' || polls >  
40  
  
    sleep(15)  
  end  
  
  puts "Instance created with ID '#{instance.first.id}'."  
  
  instance.batch_create_tags(  
    tags: [  
      {
```

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

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

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

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

Stopping an Amazon EC2 Instance

The following example attempts to stop the specified Amazon EC2 instance.

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

require 'aws-sdk-ec2'

# Attempts to stop an Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
```



```
# @return [Boolean] true if the instance was stopped; otherwise, false.
# @example
#   exit 1 unless instance_stopped?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'i-123abc'
#   )
def instance_stopped?(ec2_client, instance_id)
  response = ec2_client.describe_instance_status(instance_ids: [instance_id])

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

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

# Full example call:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:  ruby ec2-ruby-example-stop-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION '
    puts 'Example: ruby ec2-ruby-example-start-instance-i-123abc.rb ' \
      'i-123abc us-east-1'
```

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

Starting an Amazon EC2 Instance

The following example attempts to start the specified Amazon EC2 instance.

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

require 'aws-sdk-ec2'

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

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

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

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

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

```

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

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

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

run_me if $PROGRAM_NAME == __FILE__

```

Rebooting an Amazon EC2 Instance

The following example attempts to reboot the specified Amazon EC2 instance.

```

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

require 'aws-sdk-ec2'

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

```

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

# Full example call:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:  ruby ec2-ruby-example-reboot-instance-i-123abc.rb ' \
      'INSTANCE_ID REGION'
    puts 'Example: ruby ec2-ruby-example-reboot-instance-i-123abc.rb ' \
      'i-123abc us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    instance_id = 'i-123abc'
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

  ec2_client = Aws::EC2::Client.new(region: region)
  request_instance_reboot(ec2_client, instance_id)
end

run_me if $PROGRAM_NAME == __FILE__
```

Managing Amazon EC2 Instances

The following code example:

1. Stops an Amazon EC2 instance.
2. Restarts the instance.
3. Reboots the instance.
4. Enables detailed monitoring for the instance.
5. Displays information about available instances.

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

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

require 'aws-sdk-ec2'

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

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

# Attempts to stop an Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was stopped; otherwise, false.
# @example
#   exit 1 unless instance_stopped?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'i-033c48ef067af3dEX'
#   )
def instance_stopped?(ec2_client, instance_id)
  ec2_client.stop_instances(instance_ids: [instance_id])
  wait_for_instance(ec2_client, :instance_stopped, instance_id)
  return true
rescue StandardError => e
  puts "Error stopping instance: #{e.message}"
  return false
end

# Attempts to restart an Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was restarted; otherwise, false.
# @example
#   exit 1 unless instance_restarted?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'i-033c48ef067af3dEX'
#   )
def instance_restarted?(ec2_client, instance_id)
  ec2_client.start_instances(instance_ids: [instance_id])
  wait_for_instance(ec2_client, :instance_running, instance_id)
```

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

# Attempts to reboot an Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if the instance was rebooted; otherwise, false.
# @example
#   exit 1 unless instance_rebooted?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'i-033c48ef067af3dEX'
#   )
def instance_rebooted?(ec2_client, instance_id)
  ec2_client.reboot_instances(instance_ids: [instance_id])
  wait_for_instance(ec2_client, :instance_status_ok, instance_id)
  return true
rescue StandardError => e
  puts "Error rebooting instance: #{e.message}"
  return false
end

# Attempts to enabled detailed monitoring for an
# Amazon Elastic Compute Cloud (Amazon EC2) instance.
#
# Prerequisites:
#
# - The Amazon EC2 instance.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @param instance_id [String] The ID of the instance.
# @return [Boolean] true if detailed monitoring was enabled; otherwise, false.
# @example
#   exit 1 unless instance_detailed_monitoring_enabled?(
#     Aws::EC2::Client.new(region: 'us-east-1'),
#     'i-033c48ef067af3dEX'
#   )
```



```

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

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

```

```
    end
  end
end

# Full example call:
def run_me
  instance_id = ''
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage:  ruby ec2-ruby-example-manage-instances.rb ' \
        'INSTANCE_ID REGION'
    puts 'Example: ruby ec2-ruby-example-manage-instances.rb ' \
        'i-033c48ef067af3dEX us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    instance_id = 'i-033c48ef067af3dEX'
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
    instance_id = ARGV[0]
    region = ARGV[1]
  end

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

  puts 'Attempting to stop the instance. ' \
      'This might take a few minutes...'
  unless instance_stopped?(ec2_client, instance_id)
    puts 'Cannot stop the instance. Skipping this step.'
  end

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

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

```

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

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

run_me if $PROGRAM_NAME == __FILE__

```

Terminating an Amazon EC2 Instance

The following example attempts to terminate the specified Amazon EC2 instance.

```

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

require 'aws-sdk-ec2'

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

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

    puts 'The instance is already terminated.'
  end
end

```

```
    return true
  end

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

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

Getting Information about Regions and Availability Zones for Amazon EC2

The following example:

1. Displays a list of AWS Regions for Amazon EC2 that are available to you.
2. Displays a list of Amazon EC2 Availability Zones available to you depending on the AWS Region of the Amazon EC2 client.

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

require 'aws-sdk-ec2'

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

```
# Displays a list of Amazon Elastic Compute Cloud (Amazon EC2)
# Availability Zones available to you depending on the AWS Region
# of the Amazon EC2 client.
#
# @param ec2_client [Aws::EC2::Client] An initialized EC2 client.
# @example
#   list_availability_zones(Aws::EC2::Client.new(region: 'us-east-1'))
def list_availability_zones(ec2_client)
  result = ec2_client.describe_availability_zones
  # Enable pretty printing.
  max_region_string_length = 16
  max_zone_string_length = 18
  max_state_string_length = 9
  # Print header.
  print 'Region'
  print ' ' * (max_region_string_length - 'Region'.length)
  print ' Zone'
  print ' ' * (max_zone_string_length - 'Zone'.length)
  print " State\n"
  print '-' * max_region_string_length
  print ' '
  print '-' * max_zone_string_length
  print ' '
  print '-' * max_state_string_length
  print "\n"
  # Print Regions, Availability Zones, and their states.
  result.availability_zones.each do |zone|
    print zone.region_name
    print ' ' * (max_region_string_length - zone.region_name.length)
    print ' '
    print zone.zone_name
    print ' ' * (max_zone_string_length - zone.zone_name.length)
    print ' '
    print zone.state
    # Print any messages for this Availability Zone.
    if zone.messages.count.positive?
      print "\n"
      puts ' Messages for this zone:'
      zone.messages.each do |message|
        print "   #{message.message}\n"
      end
    end
  end
  print "\n"
end
```

```
end

# Full example call:
def run_me
  region = ''
  # Print usage information and then stop.
  if ARGV[0] == '--help' || ARGV[0] == '-h'
    puts 'Usage: ruby ec2-ruby-example-regions-availability-zones.rb REGION'
    puts 'Example: ruby ec2-ruby-example-regions-availability-zones.rb us-east-1'
    exit 1
  # If no values are specified at the command prompt, use these default values.
  elsif ARGV.count.zero?
    region = 'us-east-1'
  # Otherwise, use the values as specified at the command prompt.
  else
    region = ARGV[0]
  end

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

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

run_me if $PROGRAM_NAME == __FILE__
```

AWS Elastic Beanstalk Examples Using the AWS SDK for Ruby

AWS Elastic Beanstalk enables you to quickly deploy and manage applications in the AWS Cloud without worrying about the infrastructure that runs those applications. You can use the following examples to access Elastic Beanstalk using the AWS SDK for Ruby. For more information about Elastic Beanstalk, see the [AWS Elastic Beanstalk documentation](#).

Topics

- [Getting Information about All Applications in AWS Elastic Beanstalk](#)
- [Getting Information about a Specific Application in AWS Elastic Beanstalk](#)
- [Updating a Ruby on Rails Application for AWS Elastic Beanstalk](#)

Getting Information about All Applications in AWS Elastic Beanstalk

The following example lists the names, descriptions, and URLs of all of your Elastic Beanstalk applications in the us-west-2 region.

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

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

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

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

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

Getting Information about a Specific Application in AWS Elastic Beanstalk

The following example lists the name, description, and URL of the MyRailsApp application in the us-west-2 region.

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



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

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

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

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

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

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

Updating a Ruby on Rails Application for AWS Elastic Beanstalk

The following example updates the Ruby on Rails application `MyRailsApp` in the `us-west-2` region.

Note

You must be in the root of your Rails app to successfully run the script.

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

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

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

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

app_name = 'MyRailsApp'

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

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

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

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

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

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

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

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

key = app_name + "\\\" + zip_file_name
```

```
s3.put_object({
  body: zip_contents,
  bucket: bucket,
  key: key
})

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

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

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

AWS Identity and Access Management (IAM) Examples Using the AWS SDK for Ruby

AWS Identity and Access Management (IAM) is a web service for securely controlling access to AWS services. You can use the following examples to access IAM using the AWS SDK for Ruby. For more information about IAM, see the [IAM documentation](#).

Topics

- [Getting Information about IAM Users](#)
- [Listing IAM Users who are Administrators](#)
- [Adding a New IAM User](#)
- [Create User Access Keys for an IAM User](#)
- [Adding a Managed Policy to an IAM User](#)
- [Creating an IAM Role](#)

- [Managing IAM Users](#)
- [Working with IAM Policies](#)
- [Managing IAM Access Keys](#)
- [Working with IAM Server Certificates](#)
- [Managing IAM Account Aliases](#)

Getting Information about IAM Users

The following example lists the groups, policies, and access key IDs of the IAM users in the us-west-2 region. If there are more than 100 users, `iam.list_users.IsTruncated` is **true** and `iam.list_users.Marker` contains a value you can use to get information about additional users. See the [Aws::IAM::Client.list_users](#) topic for further information.

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

require 'aws-sdk-iam'

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

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

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

    users_response.users.each do |user|
```

```
name = user.user_name
puts '-' * 30
puts "User name: #{name}"

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

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

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

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

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

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

  access_keys_response.access_key_metadata.each do |access_key|
    puts "  #{access_key.access_key_id}"
  end
else
  puts '  None'
end
end
else
  puts 'No users found.'
end
```

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

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

run_me if $PROGRAM_NAME == __FILE__
```

Listing IAM Users who are Administrators

The following example uses the [get_account_authorization_details](#) method to get the list of users for the current account.

Choose Copy to save the code locally.

Create the file *get_admins.rb*.

Add the required IAM gem and the `os` gem, and use the latter to use the bundled certificate if you are running on Microsoft Windows.

Note

Version 2 of the AWS SDK for Ruby didn't have service-specific gems.

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

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

Create a method to determine whether the user has a policy with administrator privileges.

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

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

    false
  end
end
```

Create a method to determine whether the user has an attached policy with administrator privileges.

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

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

  false
end
```

Create a method to determine whether a group to which the user belongs has a policy with administrator privileges.

Create a method to determine whether a group to which the user belongs has an attached policy with administrator privileges.

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

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

```
false
end
```

Create a method to determine whether a group to which the user belongs has administrator privileges.

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

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

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

  false
end
```

Create a method to determine whether the user has administrator privileges.

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

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

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



```
false
end
```

Create a method to loop through a list of users and return how many of those users have administrator privileges.

```
<code>
```

The main routine starts here. Create an IAM client and variables to store the number of users, number of users who have administrator privileges, and the string that identifies a policy that supplies administrator privileges.

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

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

  num_admins
end
```

Call `get_account_authorization_details` to get the details of the account and get the users for the account from `user_detail_list`. Keep track of how many users we get, call `get_admin_count` to get the number of those users who have administrator privileges, and keep track of the number of those.

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

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

If the first call to `get_account_authorization_details` did not get all of the details, call it again and repeat the process of determining how many have administrator privileges.

```
<code>
```

Finally, display how many users have administrator privileges.

```
more_users = details.is_truncated
```

while more_users

```
  details = client.get_account_authorization_details(
```

```
    filter: ['User'], marker: details.marker
```

```
  )
```

```
  users = details.user_detail_list
```

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

```
  num_admins += more_admins
```

```
  more_users = details.is_truncated
```

```
end
```

See the [complete example](#) on GitHub.

Adding a New IAM User

The following example creates the IAM user `my_groovy_user` in the `us-west-2` region with the password `REPLACE_ME`, and displays the user's account ID. If a user with that name already exists, it displays a message and does not create a new user.

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

require 'aws-sdk-iam'

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

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

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

Create User Access Keys for an IAM User

The following example creates an access key and secret key for the IAM user `my_groovy_user` in the `us-west-2` region.

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

require 'aws-sdk-iam'

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

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

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

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

Adding a Managed Policy to an IAM User

The following example adds the managed policy `AmazonS3FullAccess` to the IAM user `my_groovy_user` in the `us-west-2` region.

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

require 'aws-sdk-iam'

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

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

```

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

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

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

run_me if $PROGRAM_NAME == __FILE__

```

Creating an IAM Role

The following example creates the role `my_groovy_role` so that Amazon EC2 can access Amazon S3 and Amazon DynamoDB in the `us-west-2` region.

```

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

require 'aws-sdk-iam'

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

```

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

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

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

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

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

iam_client = Aws::IAM::Client.new

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

Managing IAM Users

An IAM user represents a person or service that interacts with AWS. For more information about IAM users, see [IAM Users](#).

In this example, you use the AWS SDK for Ruby with IAM to:

1. Get information about available AWS IAM users by using [Aws::IAM::Client#list_users](#).
2. Create a user by using [Aws::IAM::Client#create_user](#).
3. Update the user's name by using [Aws::IAM::Client#update_user](#).
4. Delete the user by using [Aws::IAM::Client#delete_user](#).

Prerequisites

Before running the example code, you need to install and configure the AWS SDK for Ruby, as described in:

- [Installing the AWS SDK for Ruby](#)
- [Configuring the AWS SDK for Ruby](#)

Example

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

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

require 'aws-sdk-iam'

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

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

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

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

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

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

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

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

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

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

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

```
list_user_names(iam_client)

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

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

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

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

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

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

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

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

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

Working with IAM Policies

An IAM policy is a document that specifies one or more permissions. For more information about IAM policies, see [Overview of IAM Policies](#).

In this example, you use the AWS SDK for Ruby with IAM to:

1. Create a policy, using [Aws::IAM::Client#create_policy](#).
2. Get information about the policy, using [Aws::IAM::Client#get_policy](#).
3. Attach the policy to a role, using [Aws::IAM::Client#attach_role_policy](#).
4. List policies attached to the role, using [Aws::IAM::Client#list_attached_role_policies](#).
5. Detach the policy from the role, using [Aws::IAM::Client#detach_role_policy](#).

Prerequisites

Before running the example code, you need to install and configure the AWS SDK for Ruby, as described in:

- [Installing the AWS SDK for Ruby](#)
- [Configuring the AWS SDK for Ruby](#)

You will also need to create the role (*my-role*) specified in the script. You can do this in the IAM console.

Example

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

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

require 'aws-sdk-iam'
```

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

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

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

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

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

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

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

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

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

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

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



```
    puts 'Policy created.'
  end

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

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

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

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

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

end
end

run_me if $PROGRAM_NAME == __FILE__
```

Managing IAM Access Keys

Users need their own access keys to make programmatic calls to AWS from the AWS SDK for Ruby. To fill this need, you can create, modify, view, or rotate access keys (access key IDs and secret access keys) for IAM users. By default, when you create an access key, its status is Active. This means the user can use the access key for API calls. For more information about access keys, see [Managing Access Keys for IAM Users](#).

In this example, you use the AWS SDK for Ruby with IAM to:

1. List AWS IAM user access keys, using [Aws::IAM::Client#list_access_keys](#).

2. Create an access key, using [Aws::IAM::Client#create_access_key](#).
3. Determine when access keys were last used, using [Aws::IAM::Client#get_access_key_last_used](#).
4. Deactivate access keys, using [Aws::IAM::Client#update_access_key](#).
5. Delete the access key, using [Aws::IAM::Client#delete_access_key](#).

Prerequisites

Before running the example code, you need to install and configure the AWS SDK for Ruby, as described in:

- [Installing the AWS SDK for Ruby](#)
- [Configuring the AWS SDK for Ruby](#)

You will also need to create the user (*my-user*) specified in the script. You can create a new IAM user in the IAM console or programmatically, as shown at [Adding a New IAM User](#).

Example

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

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

require 'aws-sdk-iam'

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

```

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

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

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

```

```

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

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

  response.access_key_metadata.each do |key_metadata|
    last_used = iam.get_access_key_last_used(access_key_id: key_metadata.access_key_id)
    if last_used.access_key_last_used.last_used_date.nil?
      puts "  Key '#{key_metadata.access_key_id}' not used or date undetermined."
    else
      puts "  Key '#{key_metadata.access_key_id}' last used on " \
        "#{last_used.access_key_last_used.last_used_date}"
    end
  end
end

rescue StandardError => e
  puts "Error determining when access keys were last used: #{e.message}"
end

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

```

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

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

# Full example call:
```

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

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

  access_key = ''

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

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

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

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

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

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

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

Working with IAM Server Certificates

To enable HTTPS connections to your website or application on AWS, you need an SSL/TLS server certificate. To use a certificate that you obtained from an external provider with your website or application on AWS, you must upload the certificate to IAM or import it into AWS Certificate Manager. For more information about server certificates, see [Working with Server Certificates](#).

In this example, you use the AWS SDK for Ruby with IAM to:

1. Update a server certificate, using [Aws::IAM::Client#update_server_certificate](#).
2. Delete the server certificate, using [Aws::IAM::Client#delete_server_certificate](#).
3. List information about any remaining server certificates, using [Aws::IAM::Client#list_server_certificates](#).

Prerequisites

Before running the example code, you need to install and configure the AWS SDK for Ruby, as described in:

- [Installing the AWS SDK for Ruby](#)
- [Configuring the AWS SDK for Ruby](#)

Note

The server certificate must already exist, or the script will throw an *Aws::IAM::Errors::NoSuchEntity* error.

Example

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

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

require 'aws-sdk-iam'

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

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

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

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



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

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

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

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

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

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

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

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

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

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

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

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

Managing IAM Account Aliases

If you want the URL for your sign-in page to contain your company name or other friendly identifier instead of your AWS account ID, you can create an IAM account alias for your AWS account ID. If you create an IAM account alias, your sign-in page URL changes to incorporate the alias. For more information about IAM account aliases, see [Your AWS Account ID and Its Alias](#).

In this example, you use the AWS SDK for Ruby with IAM to:

1. List AWS account aliases, using [Aws::IAM::Client#list_account_aliases](#).
2. Create an account alias, using [Aws::IAM::Client#create_account_alias](#).
3. Delete the account alias, using [Aws::IAM::Client#delete_account_alias](#).

Prerequisites

Before running the example code, you need to install and configure the AWS SDK for Ruby, as described in:

- [Installing the AWS SDK for Ruby](#)
- [Configuring the AWS SDK for Ruby](#)

In the example code, change the *my-account-alias* string to something that will be unique across all Amazon Web Services products.

Example

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

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

require 'aws-sdk-iam'

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

  if response.account_aliases.count.positive?
    response.account_aliases.each do |account_alias|
      puts " #{account_alias}"
    end
  else
    puts 'No account aliases found.'
  end
end

rescue StandardError => e
  puts "Error listing account aliases: #{e.message}"
end

# Creates an AWS account alias.
#
# @param iam [Aws::IAM::Client] An initialized IAM client.
# @param account_alias [String] The name of the account alias to create.
# @return [Boolean] true if the account alias was created; otherwise, false.
```

```
# @example
#   exit 1 unless alias_created?(Aws::IAM::Client.new, 'my-account-alias')
def alias_created?(iam, account_alias)
  iam.create_account_alias(account_alias: account_alias)
  return true
rescue StandardError => e
  puts "Error creating account alias: #{e.message}"
  return false
end

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

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

  puts 'Account aliases are:'
  list_aliases(iam)

  if create_alias
    puts 'Attempting to create account alias...'
    if alias_created?(iam, account_alias)
      puts 'Account alias created. Account aliases now are:'
      list_aliases(iam)
    else
      puts 'Account alias not created. Stopping program.'
      exit 1
    end
  end
end
```

```
end

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

run_me if $PROGRAM_NAME == __FILE__
```

AWS Key Management Service Examples Using the AWS SDK for Ruby

AWS Key Management Service (AWS KMS) is an encryption and key management service scaled for the cloud. You can use the following examples to access AWS KMS using the AWS SDK for Ruby. For more information about AWS KMS, see the [AWS KMS documentation](#). For reference information about the AWS KMS client, see [Aws::KMS::Client](#).

Topics

- [Creating an AWS KMS key](#)
- [Encrypting Data in AWS KMS](#)
- [Decrypting a Data Blob in AWS KMS](#)
- [Re-encrypting a Data Blob in AWS KMS](#)

Creating an AWS KMS key

The following example uses the AWS SDK for Ruby [create_key](#) method, which implements the [CreateKey](#) operation to create a AWS KMS keys. Because the example only encrypts a small amount of data, a KMS key is fine for our purposes. For larger amounts of data, use the KMS key to encrypt a data encryption key (DEK).

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

```
# Create a AWS KMS key.
# As long we are only encrypting small amounts of data (4 KiB or less) directly,
# a KMS key is fine for our purposes.
# For larger amounts of data,
# use the KMS key to encrypt a data encryption key (DEK).

client = Aws::KMS::Client.new

resp = client.create_key({
  tags: [
    {
      tag_key: "CreatedBy",
      tag_value: "ExampleUser"
    }
  ]
})

puts resp.key_metadata.key_id
```

See the [complete example](#) on GitHub.

Encrypting Data in AWS KMS

The following example uses the AWS SDK for Ruby [encrypt method](#), which implements the [Encrypt operation](#), to encrypt the string "1234567890". The example displays a readable version of the resulting encrypted blob.

```
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*")
```

See the [complete example](#) on GitHub.

Decrypting a Data Blob in AWS KMS

The following example uses the AWS SDK for Ruby [decrypt method](#), which implements the [Decrypt operation](#), to decrypt the provided string and emit the result.

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

# Decrypted blob

blob =
  "01020200785d68faeec386af1057904926253051eb2919d3c16078badf65b808b26dd057c101747cadf3593596e09"
blob_packed = [blob].pack("H*")

client = Aws::KMS::Client.new(region: "us-west-2")

resp = client.decrypt({
  ciphertext_blob: blob_packed
})

puts "Raw text: "
puts resp.plaintext
```

See the [complete example](#) on GitHub.

Re-encrypting a Data Blob in AWS KMS

The following example uses the AWS SDK for Ruby [re_encrypt](#) method, which implements the [ReEncrypt](#) operation, to decrypt encrypted data and then immediately re-encrypt data under a new AWS KMS key. The operations are performed entirely on the server side within AWS KMS, so they never expose your plaintext outside of AWS KMS. The example displays a readable version of the resulting re-encrypted blob.


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

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

blob =
  "01020200785d68faeec386af1057904926253051eb2919d3c16078badf65b808b26dd057c101747cadf3593596e09"
sourceCiphertextBlob = [blob].pack("H*")

# Replace the fictitious key ARN with a valid key ID

destinationKeyId = "arn:aws:kms:us-west-2:111122223333:key/0987dcba-09fe-87dc-65ba-ab0987654321"

client = Aws::KMS::Client.new(region: "us-west-2")

resp = client.re_encrypt({
  ciphertext_blob: sourceCiphertextBlob,
  destination_key_id: destinationKeyId
})

# Display a readable version of the resulting re-encrypted blob.
puts "Blob:"
puts resp.ciphertext_blob.unpack("H*")
```

See the [complete example](#) on GitHub.

AWS Lambda Examples Using the AWS SDK for Ruby

AWS Lambda (Lambda) is a zero-administration compute platform for backend web developers that runs your code for you in the AWS Cloud, and provides you with a fine-grained pricing structure. You can use the following examples to access Lambda using the AWS SDK for Ruby. For more information about Lambda, see the [AWS Lambda documentation](#).

Topics

- [Displaying Information about All Lambda Functions](#)
- [Creating a Lambda Function](#)
- [Running a Lambda Function](#)
- [Configuring a Lambda Function to Receive Notifications](#)

Displaying Information about All Lambda Functions

The following example displays the name, ARN, and role of all of your Lambda functions in the `us-west-2` region.

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

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

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

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

Creating a Lambda Function

The following example creates the Lambda function named `my-notification-function` in the `us-west-2` region using these values:

- Role ARN: `my-resource-arn`. In most cases, you need to attach only the `AWSLambdaExecute` managed policy to the policy for this role.
- Function entry point: `my-package.my-class`
- Runtime: `java8`
- Zip file: `my-zip-file.zip`
- Bucket: `my-notification-bucket`
- Key: `my-zip-file`

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

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

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

args = {}
args[:role] = 'my-resource-arn'
args[:function_name] = 'my-notification-function'
args[:handler] = 'my-package.my-class'

# Also accepts nodejs, nodejs4.3, and python2.7
args[:runtime] = 'java8'

code = {}
code[:zip_file] = 'my-zip-file.zip'
code[:s3_bucket] = 'my-notification-bucket'
code[:s3_key] = 'my-zip-file'

args[:code] = code

client.create_function(args)
```

Running a Lambda Function

The following example runs the Lambda function named `MyGetItemsFunction` in the `us-west-2` region. This function returns a list of items from a database. The input JSON looks like the following.

```
{
  "SortBy": "name|time",
  "SortOrder": "ascending|descending",
```

```
"Number": 50
}
```

where:

- `SortBy` is the criteria for sorting the results. Our examples uses `time`, which means the returned items are sorted in the order in which they were added to the database.
- `SortOrder` is the order of sorting. Our example uses `descending`, which means the most-recent item is last in the list.
- `Number` is the maximum number of items to retrieve (the default is 50). Our example uses `10`, which means get the 10 most-recent items.

The output JSON looks like the following, where:

- `STATUS-CODE` is an HTTP status code, `200` means the call was successful.
- `RESULT` is the result of the call, either `success` or `failure`.
- `ERROR` is an error message if `result` is `failure`, otherwise an empty string
- `DATA` is an array of returned results if `result` is `success`, otherwise `nil`.

```
{
  "statusCode": "STATUS-CODE",
  "body": {
    "result": "RESULT",
    "error": "ERROR",
    "data": "DATA"
  }
}
```

The first step is to load the modules we use:

- `aws-sdk` loads the AWS SDK for Ruby module we use to invoke the Lambda function.
- `json` loads the JSON module we use to marshal and unmarshal the request and response payloads.
- `os` loads the OS module we use to ensure we can run our Ruby application on Microsoft Windows. If you are on a different operating system, you can remove those lines.
- We then create the Lambda client we use to invoke the Lambda function.

- Next we create the hash for the request arguments and call `MyGetItemsFunction`.
- Finally we parse the response, and if are successful, we print out the items.

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

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

# To run on Windows:
require 'os'
if OS.windows?
  Aws.use_bundled_cert!
end

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

# Get the 10 most recent items
req_payload = {:SortBy => 'time', :SortOrder => 'descending', :NumberToGet => 10}
payload = JSON.generate(req_payload)

resp = client.invoke({
  function_name: 'MyGetItemsFunction',
  invocation_type: 'RequestResponse',
  log_type: 'None',
  payload: payload
})

resp_payload = JSON.parse(resp.payload.string) # , symbolize_names: true)

# If the status code is 200, the call succeeded
if resp_payload["statusCode"] == 200
  # If the result is success, we got our items
```

```
if resp_payload["body"]["result"] == "success"
  # Print out items
  resp_payload["body"]["data"].each do |item|
    puts item
  end
end
end
end
```

See the [complete example](#) on GitHub.

Configuring a Lambda Function to Receive Notifications

The following example configures the Lambda function named `my-notification-function` in the `us-west-2` region to accept notifications from the resource with the ARN `my-resource-arn`.

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

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

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

args = {}
args[:function_name] = 'my-notification-function'
args[:statement_id] = 'lambda_s3_notification'
args[:action] = 'lambda:InvokeFunction'
args[:principal] = 's3.amazonaws.com'
args[:source_arn] = 'my-resource-arn'

client.add_permission(args)
```

Amazon Polly Examples Using the AWS SDK for Ruby

Amazon Polly is a cloud service that converts text into lifelike speech. The AWS SDK for Ruby examples can integrate Amazon Polly into your applications. Learn more about Amazon Polly at [Amazon Polly documentation](#). The examples assume you have already set up and configured the SDK (that is, you've imported all required packages and set your credentials and region). For more information, see [Installing the AWS SDK for Ruby](#) and [Configuring the AWS SDK for Ruby](#).

Topics

- [Getting a List of Voices](#)
- [Getting a List of Lexicons](#)
- [Synthesizing Speech](#)

Getting a List of Voices

This example uses the [describe_voices](#) method to get the list of US English voices in the us-west-2 region.

Choose Copy to save the code locally.

Create the file `polly_describe_voices.rb`.

Add the required gem.

Note

Version 2 of the AWS SDK for Ruby didn't have service-specific gems.

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

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

require 'aws-sdk-polly' # In v2: require 'aws-sdk'

begin
  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  # Get US English voices
  resp = polly.describe_voices(language_code: 'en-US')

  resp.voices.each do |v|
    puts v.name
    puts ' ' + v.gender
    puts
  end
rescue StandardError => ex
  puts 'Could not get voices'
  puts 'Error message:'
  puts ex.message
end
```

See the [complete example](#) on GitHub.

Getting a List of Lexicons

This example uses the [list_lexicons](#) method to get the list of lexicons in the us-west-2 region.

Choose Copy to save the code locally.

Create the file *polly_list_lexicons.rb*.

Add the required gem.

Note

Version 2 of the AWS SDK for Ruby didn't have service-specific gems.

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



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

require 'aws-sdk-polly' # In v2: require 'aws-sdk'

begin
  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  resp = polly.list_lexicons

  resp.lexicons.each do |l|
    puts l.name
    puts '  Alphabet:' + l.attributes.alphabet
    puts '  Language:' + l.attributes.language
    puts
  end
rescue StandardError => ex
  puts 'Could not get lexicons'
  puts 'Error message:'
  puts ex.message
end
```

See the [complete example](#) on GitHub.

Synthesizing Speech

This example uses the [synthesize_speech](#) method to get the text from a file and produce an MP3 file containing the synthesized speech.

Choose Copy to save the code locally.

Create the file `polly_synthesize_speech.rb`.

Add the required gem.

Note

Version 2 of the AWS SDK for Ruby didn't have service-specific gems.

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

require 'aws-sdk-polly' # In v2: require 'aws-sdk'

begin
  # Get the filename from the command line
  if ARGV.empty?()
    puts 'You must supply a filename'
    exit 1
  end

  filename = ARGV[0]

  # Open file and get the contents as a string
  if File.exist?(filename)
    contents = IO.read(filename)
  else
    puts 'No such file: ' + filename
    exit 1
  end

  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  resp = polly.synthesize_speech({
```

```
    output_format: "mp3",
    text: contents,
    voice_id: "Joanna",
  })

  # Save output
  # Get just the file name
  # abc/xyz.txt -> xyz.txt
  name = File.basename(filename)

  # Split up name so we get just the xyz part
  parts = name.split('.')
  first_part = parts[0]
  mp3_file = first_part + '.mp3'

  IO.copy_stream(resp.audio_stream, mp3_file)

  puts 'Wrote MP3 content to: ' + mp3_file
rescue StandardError => ex
  puts 'Got error:'
  puts 'Error message:'
  puts ex.message
end
```

Note

The resulting MP3 file is in the MPEG-2 format.

See the [complete example](#) on GitHub.

Amazon RDS Examples Using the AWS SDK for Ruby

Amazon Relational Database Service (Amazon RDS) is a web service that makes it easier to set up, operate, and scale a relational database in the cloud. You can use the following examples to access Amazon RDS using the AWS SDK for Ruby. For more information about Amazon RDS, see the [Amazon Relational Database Service documentation](#).

Note

Some of the following examples use methods that were introduced in the 2.2.18 version of the `Aws::RDS::Resource` class. To run those examples, you must use that version or a later version of the `aws-sdk` gem.

Topics

- [Getting Information about All Amazon RDS Instances](#)
- [Getting Information about All Amazon RDS Snapshots](#)
- [Getting Information about All Amazon RDS Clusters and Their Snapshots](#)
- [Getting Information about All Amazon RDS Security Groups](#)
- [Getting Information about All Amazon RDS Subnet Groups](#)
- [Getting Information about All Amazon RDS Parameter Groups](#)
- [Creating a Snapshot of an Amazon RDS Instance](#)
- [Creating a Snapshot of an Amazon RDS Cluster](#)

Getting Information about All Amazon RDS Instances

The following example lists the name (ID) and status of all of your Amazon RDS instances in the `us-west-2` region.

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

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

rds = Aws::RDS::Resource.new(region: 'us-west-2')
```

```
rds.db_instances.each do |i|
  puts "Name (ID): #{i.id}"
  puts "Status   : #{i.db_instance_status}"
  puts
end
```

Getting Information about All Amazon RDS Snapshots

The following example lists the names (IDs) and status of all of your Amazon RDS (instance) snapshots in the us-west-2 region.

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

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

rds = Aws::RDS::Resource.new(region: 'us-west-2')

rds.db_snapshots.each do |s|
  puts "Name (ID): #{s.snapshot_id}"
  puts "Status:   #{s.status}"
end
```

Getting Information about All Amazon RDS Clusters and Their Snapshots

The following example lists the name (ID) and status of all of your Amazon RDS clusters and the name (ID) and status of their snapshots in the us-west-2 region.

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

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

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

rds = Aws::RDS::Resource.new(region: 'us-west-2')

rds.db_clusters.each do |c|
  puts "Name (ID): #{c.id}"
  puts "Status:    #{c.status}"

  c.snapshots.each do |s|
    puts "  Snapshot: #{s.snapshot_id}"
    puts "  Status:    #{s.status}"
  end
end
```

Getting Information about All Amazon RDS Security Groups

The following example lists the names of all of your Amazon RDS security groups in the us-west-2 region.

Note

Amazon RDS security groups are only applicable when you are using the Amazon EC2-Classical platform. If you are using Amazon EC2-VPC, use VPC security groups. Both are shown in the example.

Warning

We are retiring EC2-Classical on August 15, 2022. We recommend that you migrate from EC2-Classical to a VPC. For more information, see **Migrate from EC2-Classical to a VPC** in the [Amazon EC2 User Guide for Linux Instances](#) or the [Amazon EC2 User Guide for Windows Instances](#). Also see the blog post [EC2-Classical Networking is Retiring – Here's How to Prepare](#).

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

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

rds = Aws::RDS::Resource.new(region: 'us-west-2')

rds.db_instances.each do |i|
  # Show any security group IDs and descriptions
  puts 'Security Groups:'

  i.db_security_groups.each do |sg|
    puts sg.db_security_group_name
    puts '  ' + sg.db_security_group_description
    puts
  end

  # Show any VPC security group IDs and their status
  puts 'VPC Security Groups:'

  i.vpc_security_groups.each do |vsg|
    puts vsg.vpc_security_group_id
    puts '  ' + vsg.status
    puts
  end
end
end
```

Getting Information about All Amazon RDS Subnet Groups

The following example lists the name and status of all of your Amazon RDS subnet groups in the `us-west-2` region.

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

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

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

rds = Aws::RDS::Resource.new(region: 'us-west-2')

rds.db_subnet_groups.each do |s|
  puts s.name
  puts ' ' + s.subnet_group_status
end
```

Getting Information about All Amazon RDS Parameter Groups

The following example lists the names and descriptions of all of your Amazon RDS parameter groups in the us-west-2 region.

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

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

rds = Aws::RDS::Resource.new(region: 'us-west-2')

rds.db_parameter_groups.each do |p|
  puts p.db_parameter_group_name
  puts ' ' + p.description
end
```



```
end
```

Creating a Snapshot of an Amazon RDS Instance

The following example creates a snapshot for the Amazon RDS instance represented by *instance_name* in the *us-west-2* region.

Note

If your instance is a member of a cluster, you can't create a snapshot of the instance. Instead, you must create a snapshot of the cluster (see [Creating a Snapshot of an Amazon RDS Cluster](#)).

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

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

rds = Aws::RDS::Resource.new(region: 'us-west-2')

instance = rds.db_instance(instance_name)

date = Time.new
date_time = date.year.to_s + '-' + date.month.to_s + '-' + date.day.to_s + '-' +
  date.hour.to_s + '-' + date.min.to_s

id = instance_name + '-' + date_time

instance.create_snapshot({db_snapshot_identifier: id})

puts "Created snapshot #{id}"
```

Creating a Snapshot of an Amazon RDS Cluster

The following example creates a snapshot for the Amazon RDS cluster represented by *cluster_name* in the *us-west-2* region.

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

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

rds = Aws::RDS::Resource.new(region: 'us-west-2')

cluster = rds.db_cluster(cluster_name)

date = Time.new
date_time = date.year.to_s + '-' + date.month.to_s + '-' + date.day.to_s + '-' +
  date.hour.to_s + '-' + date.min.to_s

id = cluster_name + '-' + date_time

cluster.create_snapshot({db_cluster_snapshot_identifier: id})

puts "Created cluster snapshot #{id}"
```

Amazon SES Examples Using the AWS SDK for Ruby

Amazon Simple Email Service (Amazon SES) is an email platform that provides an easy, cost-effective way for you to send and receive email using your own email addresses and domains. You can use the following examples to access Amazon SES using the AWS SDK for Ruby. For more information about Amazon SES, see the [Amazon SES documentation](#).

Topics

- [Listing Valid Amazon SES Email Addresses](#)

- [Verifying an Email Address in Amazon SES](#)
- [Sending a Message to an Email Address in Amazon SES](#)
- [Getting Amazon SES Statistics](#)

Listing Valid Amazon SES Email Addresses

The following example demonstrates how to use the AWS SDK for Ruby to list the valid Amazon SES email addresses.

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

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

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

# Get up to 1000 identities
ids = client.list_identities({
  identity_type: "EmailAddress"
})

ids.identities.each do |email|
  attrs = client.get_identity_verification_attributes({
    identities: [email]
  })

  status = attrs.verification_attributes[email].verification_status

  # Display email addresses that have been verified
  if status == "Success"
    puts email
  end
end
```

```
end
```

See the [complete example](#) on GitHub.

Verifying an Email Address in Amazon SES

The following example demonstrates how to use the AWS SDK for Ruby to verify an Amazon SES email address.

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

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

# Replace recipient@example.com with a "To" address.
recipient = "recipient@example.com"

# Create a new SES resource in the us-west-2 region.
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
ses = Aws::SES::Client.new(region: 'us-west-2')

# Try to verify email address.
begin
  ses.verify_email_identity({
    email_address: recipient
  })

  puts 'Email sent to ' + recipient

# If something goes wrong, display an error message.
rescue Aws::SES::Errors::ServiceError => error
  puts "Email not sent. Error message: #{error}"
end
```

See the [complete example](#) on GitHub.

Sending a Message to an Email Address in Amazon SES

The following example demonstrates how to use the AWS SDK for Ruby to send a message to an Amazon SES email address.

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

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

# Replace sender@example.com with your "From" address.
# This address must be verified with Amazon SES.
sender = 'sender@example.com'

# Replace recipient@example.com with a "To" address. If your account
# is still in the sandbox, this address must be verified.
recipient = 'recipient@example.com'

# Specify a configuration set. To use a configuration
# set, uncomment the next line and line 74.
# configsetname = "ConfigSet"

# The subject line for the email.
subject = 'Amazon SES test (AWS SDK for Ruby)'

# The HTML body of the email.
htmlbody =
  '<h1>Amazon SES test (AWS SDK for Ruby)</h1>'\
  '<p>This email was sent with <a href="https://aws.amazon.com/ses/">'\
  'Amazon SES</a> using the <a href="https://aws.amazon.com/sdk-for-ruby/">'\
  'AWS SDK for Ruby</a>.'
```

```
# The email body for recipients with non-HTML email clients.
textbody = 'This email was sent with Amazon SES using the AWS SDK for Ruby.'

# Specify the text encoding scheme.
encoding = 'UTF-8'

# Create a new SES client in the us-west-2 region.
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
ses = Aws::SES::Client.new(region: 'us-west-2')

# Try to send the email.
begin
  # Provide the contents of the email.
  ses.send_email(
    destination: {
      to_addresses: [
        recipient
      ]
    },
    message: {
      body: {
        html: {
          charset: encoding,
          data: htmlbody
        },
        text: {
          charset: encoding,
          data: textbody
        }
      },
      subject: {
        charset: encoding,
        data: subject
      }
    },
    source: sender,
    # Uncomment the following line to use a configuration set.
    # configuration_set_name: configsetname,
  )

  puts 'Email sent to ' + recipient

# If something goes wrong, display an error message.
```

```
rescue Aws::SES::Errors::ServiceError => error
  puts "Email not sent. Error message: #{error}"
end
```

See the [complete example](#) on GitHub.

Getting Amazon SES Statistics

The following example demonstrates how to use the AWS SDK for Ruby to get statistics about Amazon SES. Use this information to avoid damaging your reputation when emails are bounced or rejected.

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

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

# Create a new SES resource in the us-west-2 region.
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
ses = Aws::SES::Client.new(region: 'us-west-2')

begin
  # Get send statistics so we don't ruin our reputation
  resp = ses.get_send_statistics({})

  dps = resp.send_data_points

  puts "Got #{dps.count} data point(s):"
  puts

  dps.each do |dp|
    puts "Timestamp:  #{dp.timestamp}" #=> Time
    puts "Attempts:    #{dp.delivery_attempts}" #=> Integer
    puts "Bounces:      #{dp.bounces}" #=> Integer
```

```
puts "Complaints: #{dp.complaints}" #=> Integer
puts "Rejects:    #{dp.rejects}" #-> Integer
puts
end

# If something goes wrong, display an error message.
rescue Aws::SES::Errors::ServiceError => error
  puts "Error: #{error}"
end
```

See the [complete example](#) on GitHub.

Amazon SNS Examples Using the AWS SDK for Ruby

Amazon Simple Notification Service (Amazon SNS) is a web service that enables applications, end users, and devices to instantly send and receive notifications from the cloud. You can use the following examples to access Amazon SNS using the AWS SDK for Ruby. For more information about Amazon SNS, see the [Amazon SNS documentation](#).

Topics

- [Getting Information about All Amazon SNS Topics](#)
- [Creating an Amazon SNS Topic](#)
- [Getting Information about All Subscriptions in an Amazon SNS Topic](#)
- [Creating a Subscription in an Amazon SNS Topic](#)
- [Sending a Message to All Amazon SNS Topic Subscribers](#)
- [Enabling a Resource to Publish to an Amazon SNS Topic](#)

Getting Information about All Amazon SNS Topics

The following example lists the ARNs of your Amazon SNS topics in the us-west-2 region.

```
# Copyright 2010-2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#
# This file is licensed under the Apache License, Version 2.0 (the "License").
# You may not use this file except in compliance with the License. A copy of the
# License is located at
#
# http://aws.amazon.com/apache2.0/
#
```



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

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

sns = Aws::SNS::Resource.new(region: 'us-west-2')

sns.topics.each do |topic|
  puts topic.arn
end
```

Creating an Amazon SNS Topic

The following example creates the topic `MyGroovyTopic` in the `us-west-2` region and displays the resulting topic ARN.

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

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

sns = Aws::SNS::Resource.new(region: 'us-west-2')

topic = sns.create_topic(name: 'MyGroovyTopic')
puts topic.arn
```

Getting Information about All Subscriptions in an Amazon SNS Topic

The following example lists the email addresses of the Amazon SNS subscriptions for the topic with the ARN `arn:aws:sns:us-west-2:123456789:MyGroovyTopic` in the `us-west-2` region.

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

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

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

sns = Aws::SNS::Resource.new(region: 'us-west-2')

topic = sns.topic('arn:aws:sns:us-west-2:123456789:MyGroovyTopic')

topic.subscriptions.each do |s|
  puts s.attributes['Endpoint']
end
```

Creating a Subscription in an Amazon SNS Topic

The following example creates a subscription for the topic with the ARN `arn:aws:sns:us-west-2:123456789:MyGroovyTopic` for a user who has the email address `MyGroovyUser@MyGroovy.com` in the `us-west-2` region, and displays the resulting ARN. Initially the ARN value is pending confirmation. When the user confirms their email address, this value becomes a true ARN.

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

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

```
sns = Aws::SNS::Resource.new(region: 'us-west-2')

topic = sns.topic('arn:aws:sns:us-west-2:123456789:MyGroovyTopic')

sub = topic.subscribe({
  protocol: 'email',
  endpoint: 'MyGroovyUser@MyGroovy.com'
})

puts sub.arn
```

Sending a Message to All Amazon SNS Topic Subscribers

The following example sends the message "Hello!" to all subscribers to the Amazon SNS topic with the ARN `arn:aws:sns:us-west-2:123456789:MyGroovyTopic`.

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

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

sns = Aws::SNS::Resource.new(region: 'us-west-2')

topic = sns.topic('arn:aws:sns:us-west-2:123456789:MyGroovyTopic')

topic.publish({
  message: 'Hello!'
})
```

Enabling a Resource to Publish to an Amazon SNS Topic

The following example enables the resource with the ARN `my-resource-arn` to publish to the topic with the ARN `my-topic-arn` in the `us-west-2` region.

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

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

policy = '{
  "Version":"2008-10-17",
  "Id":"__default_policy_ID",
  "Statement":[{
    "Sid":"__default_statement_ID",
    "Effect":"Allow",
    "Principal":{"
      "AWS":"*"
    },
    "Action":["SNS:Publish"],
    "Resource":"" + my-topic-arn + "",
    "Condition":{"
      "ArnEquals":{"
        "AWS:SourceArn":"" + my-resource-arn + ""}
      }
    }
  ]
}'

sns = Aws::SNS::Resource.new(region: 'us-west-2')

# Get topic by ARN
topic = sns.topic(my-topic-arn)

# Add policy to topic
topic.set_attributes({
  attribute_name: "Policy",
  attribute_value: policy
})
```

Amazon SQS Examples Using the AWS SDK for Ruby

Amazon Simple Queue Service (Amazon SQS) is a fully managed message queuing service that makes it easy to decouple and scale microservices, distributed systems, and serverless applications. You can use the following examples to access Amazon SQS using the AWS SDK for Ruby. For more information about Amazon SQS, see the [Amazon SQS documentation](#).

Topics

- [Getting Information about All Queues in Amazon SQS](#)
- [Creating a Queue in Amazon SQS](#)
- [Working with Queues in Amazon SQS](#)
- [Sending Messages in Amazon SQS](#)
- [Sending and Receiving Messages in Amazon SQS](#)
- [Receiving Messages in Amazon SQS](#)
- [Receiving Messages Using Long Polling in Amazon SQS](#)
- [Enabling Long Polling in Amazon SQS](#)
- [Receiving Messages Using the QueuePoller Class in Amazon SQS](#)
- [Redirecting Dead Letters in Amazon SQS](#)
- [Deleting a Queue in Amazon SQS](#)
- [Enabling a Resource to Publish to a Queue in Amazon SQS](#)
- [Working with a Dead Letter Queue in Amazon SQS](#)
- [Specifying the Message Visibility Timeout in Amazon SQS](#)

Getting Information about All Queues in Amazon SQS

The following example lists the URLs, ARNs, messages available, and messages in flight of your Amazon SQS queues in the us-west-2 region.

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

require 'aws-sdk-sqs'
require 'aws-sdk-sts'

# Lists the URLs of available queues in Amazon Simple Queue Service (Amazon SQS).
#
```

```
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @example
#   list_queue_urls(Aws::SQS::Client.new(region: 'us-east-1'))
def list_queue_urls(sqs_client)
  queues = sqs_client.list_queues

  queues.queue_urls.each do |url|
    puts url
  end
rescue StandardError => e
  puts "Error listing queue URLs: #{e.message}"
end

# Lists the attributes of a queue in Amazon Simple Queue Service (Amazon SQS).
#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @example
#   list_queue_attributes(
#     Aws::SQS::Client.new(region: 'us-east-1'),
#     'https://sqs.us-east-1.amazonaws.com/111111111111/my-queue'
#   )
def list_queue_attributes(sqs_client, queue_url)
  attributes = sqs_client.get_queue_attributes(
    queue_url: queue_url,
    attribute_names: [ "All" ]
  )

  attributes.attributes.each do |key, value|
    puts "#{key}: #{value}"
  end
end

rescue StandardError => e
  puts "Error getting queue attributes: #{e.message}"
end

# Full example call:
def run_me
  region = 'us-east-1'
  queue_name = 'my-queue'

  sqs_client = Aws::SQS::Client.new(region: region)

  puts 'Listing available queue URLs...'
```

```
list_queue_urls(sqs_client)

sts_client = Aws::STS::Client.new(region: region)

# For example:
# 'https://sqs.us-east-1.amazonaws.com/111111111111/my-queue'
queue_url = 'https://sqs.' + region + '.amazonaws.com/' +
  sts_client.get_caller_identity.account + '/' + queue_name

puts "\nGetting information about queue '#{queue_name}'..."
list_queue_attributes(sqs_client, queue_url)
end

run_me if $PROGRAM_NAME == __FILE__
```

Creating a Queue in Amazon SQS

The following example creates the Amazon SQS queue named `MyGroovyQueue` in the `us-west-2` region and displays its URL.

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

require 'aws-sdk-sqs'

# Creates a queue in Amazon Simple Queue Service (Amazon SQS).
#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_name [String] The name of the queue.
# @return [Boolean] true if the queue was created; otherwise, false.
# @example
#   exit 1 unless queue_created?(
#     Aws::SQS::Client.new(region: 'us-east-1'),
#     'my-queue'
#   )
def queue_created?(sqs_client, queue_name)
  sqs_client.create_queue(queue_name: queue_name)
  true
rescue StandardError => e
  puts "Error creating queue: #{e.message}"
  false
end
```

```
# Full example call:
def run_me
  region = 'us-east-1'
  queue_name = 'my-queue'
  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Creating the queue named '#{queue_name}'..."

  if queue_created?(sqs_client, queue_name)
    puts 'Queue created.'
  else
    puts 'Queue not created.'
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

Working with Queues in Amazon SQS

Amazon SQS provides highly scalable hosted queues for storing messages as they travel between applications or microservices. To learn more about queues, see [How Amazon SQS Queues Work](#).

In this example, you use the AWS SDK for Ruby with Amazon SQS to:

1. Get a list of your queues by using [Aws::SQS::Client#list_queues](#).
2. Create a queue by using [Aws::SQS::Client#create_queue](#).
3. Get the queue's URL by using [Aws::SQS::Client#get_queue_url](#).
4. Delete the queue by using [Aws::SQS::Client#delete_queue](#).

Prerequisites

Before running the example code, you need to install and configure the AWS SDK for Ruby, as described in:

- [Installing the AWS SDK for Ruby](#)
- [Configuring the AWS SDK for Ruby](#)

Example

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



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

# Demonstrates how to:
# 1. Get a list of your queues.
# 2. Create a queue.
# 3. Get the queue's URL.
# 4. Delete the queue.

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

sqs = Aws::SQS::Client.new(region: 'us-east-1')

# Get a list of your queues.
sqs.list_queues.queue_urls.each do |queue_url|
  puts queue_url
end

# Create a queue.
queue_name = "my-queue"

begin
  sqs.create_queue({
    queue_name: queue_name,
    attributes: {
      "DelaySeconds" => "60", # Delay message delivery for 1 minute (60 seconds).
      "MessageRetentionPeriod" => "86400" # Delete message after 1 day (24 hours * 60
minutes * 60 seconds).
    }
  })
rescue Aws::SQS::Errors::QueueDeletedRecently
  puts "A queue with the name '#{queue_name}' was recently deleted. Wait at least 60
seconds and try again."
  exit(false)
end
```

```
# Get the queue's URL.
queue_url = sqs.get_queue_url(queue_name: queue_name).queue_url
puts queue_url

# Delete the queue.
sqs.delete_queue(queue_url: queue_url)
```

Sending Messages in Amazon SQS

The following example sends the message “Hello world” through the Amazon SQS queue with the URL URL in the us-west-2 region.

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

require 'aws-sdk-sqs'
require 'aws-sdk-sts'

# Sends a message to a queue in Amazon Simple Queue Service (Amazon SQS).
#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param message_body [String] The contents of the message to be sent.
# @return [Boolean] true if the message was sent; otherwise, false.
# @example
#   exit 1 unless message_sent?(
#     Aws::SQS::Client.new(region: 'us-east-1'),
#     'https://sqs.us-east-1.amazonaws.com/111111111111/my-queue',
#     'This is my message.'
#   )
def message_sent?(sqs_client, queue_url, message_body)
  sqs_client.send_message(
    queue_url: queue_url,
    message_body: message_body
  )
  true
rescue StandardError => e
  puts "Error sending message: #{e.message}"
  false
end

# Full example call:
def run_me
```

```

region = 'us-east-1'
queue_name = 'my-queue'
message_body = 'This is my message.'

sts_client = Aws::STS::Client.new(region: region)

# For example:
# 'https://sqs.us-east-1.amazonaws.com/111111111111/my-queue'
queue_url = 'https://sqs.' + region + '.amazonaws.com/' +
  sts_client.get_caller_identity.account + '/' + queue_name

sqs_client = Aws::SQS::Client.new(region: region)

puts "Sending a message to the queue named '#{queue_name}'..."

if message_sent?(sqs_client, queue_url, message_body)
  puts 'Message sent.'
else
  puts 'Message not sent.'
end
end

run_me if $PROGRAM_NAME == __FILE__

```

The following example sends the messages “Hello world” and “How is the weather?” through the Amazon SQS queue with the URL URL in the us-west-2 region.

Note

If your queue is a FIFO queue, you must include a `message_group_id` parameter in addition to the `id` and `message_body` parameters.

```

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

require 'aws-sdk-sqs'
require 'aws-sdk-sts'

# Sends multiple messages as a batch to a queue in
# Amazon Simple Queue Service (Amazon SQS).
#

```

```
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param entries [Hash] The contents of the messages to be sent,
#   in the correct format.
# @return [Boolean] true if the messages were sent; otherwise, false.
# @example
#   exit 1 unless messages_sent?(
#     Aws::SQS::Client.new(region: 'us-east-1'),
#     'https://sqs.us-east-1.amazonaws.com/111111111111/my-queue',
#     [
#       {
#         id: 'Message1',
#         message_body: 'This is the first message.'
#       },
#       {
#         id: 'Message2',
#         message_body: 'This is the second message.'
#       }
#     ]
#   )
def messages_sent?(sqs_client, queue_url, entries)
  sqs_client.send_message_batch(
    queue_url: queue_url,
    entries: entries
  )
  true
rescue StandardError => e
  puts "Error sending messages: #{e.message}"
  false
end

# Full example call:
def run_me
  region = 'us-east-1'
  queue_name = 'my-queue'
  entries = [
    {
      id: 'Message1',
      message_body: 'This is the first message.'
    },
    {
      id: 'Message2',
      message_body: 'This is the second message.'
    }
  ]
end
```

```
]

sts_client = Aws::STS::Client.new(region: region)

# For example:
# 'https://sqs.us-east-1.amazonaws.com/111111111111/my-queue'
queue_url = 'https://sqs.' + region + '.amazonaws.com/' +
  sts_client.get_caller_identity.account + '/' + queue_name

sqs_client = Aws::SQS::Client.new(region: region)

puts "Sending messages to the queue named '#{queue_name}'..."

if messages_sent?(sqs_client, queue_url, entries)
  puts 'Messages sent.'
else
  puts 'Messages not sent.'
end
end

run_me if $PROGRAM_NAME == __FILE__
```

Sending and Receiving Messages in Amazon SQS

After you create a queue in Amazon SQS, you can send a message to it and then consume it. To learn more, see [Tutorial: Sending a Message to an Amazon SQS Queue](#) and [Tutorial: Receiving and Deleting a Message from an Amazon SQS Queue](#).

In this example, you use the AWS SDK for Ruby with Amazon SQS to:

1. Send a message to a queue by using [Aws::SQS::Client#send_message](#).

Note

If your queue is a FIFO queue, you must include a `message_group_id` parameter in addition to the `id` and `message_body` parameters.

1. Receive the message in the queue by using [Aws::SQS::Client#receive_message](#).
2. Display information about the message.

3. Delete the message from the queue by using [Aws::SQS::Client#delete_message](#).

Prerequisites

Before running the example code, you need to install and configure the AWS SDK for Ruby, as described in:

- [Installing the AWS SDK for Ruby](#)
- [Configuring the AWS SDK for Ruby](#)

You also need to create the queue *my-queue*, which you can do in the Amazon SQS console.

Example

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

# Demonstrates how to:
# 1. Send a message to a queue.
# 2. Receive the message in the queue.
# 3. Display information about the message.
# 4. Delete the message from the queue.

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

sqs = Aws::SQS::Client.new(region: 'us-east-1')

# Send a message to a queue.
queue_name = "my-queue"

begin
  queue_url = sqs.get_queue_url(queue_name: queue_name).queue_url
```

```
# Create a message with three custom attributes: Title, Author, and WeeksOn.
send_message_result = sqs.send_message({
  queue_url: queue_url,
  message_body: "Information about current NY Times fiction bestseller for week of
2016-12-11.",
  message_attributes: {
    "Title" => {
      string_value: "The Whistler",
      data_type: "String"
    },
    "Author" => {
      string_value: "John Grisham",
      data_type: "String"
    },
    "WeeksOn" => {
      string_value: "6",
      data_type: "Number"
    }
  }
})
rescue Aws::SQS::Errors::NonExistentQueue
  puts "A queue named '#{queue_name}' does not exist."
  exit(false)
end

puts send_message_result.message_id

# Receive the message in the queue.
receive_message_result = sqs.receive_message({
  queue_url: queue_url,
  message_attribute_names: ["All"], # Receive all custom attributes.
  max_number_of_messages: 1, # Receive at most one message.
  wait_time_seconds: 0 # Do not wait to check for the message.
})

# Display information about the message.
# Display the message's body and each custom attribute value.
receive_message_result.messages.each do |message|
  puts message.body
  puts "Title: #{message.message_attributes["Title"]["string_value"]}"
  puts "Author: #{message.message_attributes["Author"]["string_value"]}"
  puts "WeeksOn: #{message.message_attributes["WeeksOn"]["string_value"]}"

  # Delete the message from the queue.
end
```

```
sqs.delete_message({
  queue_url: queue_url,
  receipt_handle: message.receipt_handle
})
end
```

Receiving Messages in Amazon SQS

The following example displays the body of up to 10 messages in the Amazon SQS queue with the URL URL in the us-west-2 region.

Note

`receive_message` does not guarantee to get all messages (see [Properties of Distributed Queues](#)), and by default does not delete the message.

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

require 'aws-sdk-sqs'
require 'aws-sdk-sts'

# Receives messages in a queue in Amazon Simple Queue Service (Amazon SQS).
#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param max_number_of_messages [Integer] The maximum number of messages
#   to receive. This number must be 10 or less. The default is 10.
# @example
#   receive_messages(
#     Aws::SQS::Client.new(region: 'us-east-1'),
#     'https://sqs.us-east-1.amazonaws.com/111111111111/my-queue',
#     10
#   )
def receive_messages(sqs_client, queue_url, max_number_of_messages = 10)

  if max_number_of_messages > 10
    puts 'Maximum number of messages to receive must be 10 or less. ' \
      'Stopping program.'
    return
  end
```



```
response = sqs_client.receive_message(
  queue_url: queue_url,
  max_number_of_messages: max_number_of_messages
)

if response.messages.count.zero?
  puts 'No messages to receive, or all messages have already ' \
    'been previously received.'
  return
end

response.messages.each do |message|
  puts '-' * 20
  puts "Message body: #{message.body}"
  puts "Message ID:  #{message.message_id}"
end

rescue StandardError => e
  puts "Error receiving messages: #{e.message}"
end

# Full example call:
def run_me
  region = 'us-east-1'
  queue_name = 'my-queue'
  max_number_of_messages = 10

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-east-1.amazonaws.com/111111111111/my-queue'
  queue_url = 'https://sqs.' + region + '.amazonaws.com/' +
    sts_client.get_caller_identity.account + '/' + queue_name

  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Receiving messages from queue '#{queue_name}'..."

  receive_messages(sqs_client, queue_url, max_number_of_messages)
end

run_me if $PROGRAM_NAME == __FILE__
```

Receiving Messages Using Long Polling in Amazon SQS

The following example waits up to 10 seconds to display the bodies of up to 10 messages in the Amazon SQS queue with the URL URL in the us-west-2 region.

If you do not specify a wait time, the default value is **0** (Amazon SQS does not wait).

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

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

sqs = Aws::SQS::Client.new(region: 'us-west-2')

resp = sqs.receive_message(queue_url: URL, max_number_of_messages: 10,
  wait_time_seconds: 10)

resp.messages.each do |m|
  puts m.body
end
```

Enabling Long Polling in Amazon SQS

Long polling helps lower your cost of using Amazon SQS by reducing the number of empty responses and eliminating false empty responses. For more information about long polling, see [Amazon SQS Long Polling](#).

In this example, you use the AWS SDK for Ruby with Amazon SQS to:

1. Create a queue and set it for long polling by using [Aws::SQS::Client#create_queue](#).
2. Set long polling for an existing queue by using [Aws::SQS::Client#set_queue_attributes](#).
3. Set long polling when receiving messages for a queue by using [Aws::SQS::Client#receive_message](#).

Prerequisites

Before running the example code, you need to install and configure the AWS SDK for Ruby, as described in:

- [Installing the AWS SDK for Ruby](#)
- [Configuring the AWS SDK for Ruby](#)

You also need to create the queues *existing-queue* and *receive-queue*, which you can do in the Amazon SQS console.

Example

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

# Demonstrates how to:
# 1. Create a queue and set it for long polling.
# 2. Set long polling for an existing queue.
# 3. Set long polling when receiving messages for a queue.

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

sqs = Aws::SQS::Client.new(region: 'us-east-1')

# Create a queue and set it for long polling.
new_queue_name = "new-queue"

create_queue_result = sqs.create_queue({
  queue_name: new_queue_name,
  attributes: {
    "ReceiveMessageWaitTimeSeconds" => "20" # Wait 20 seconds to receive messages.
  },
```

```
})

puts create_queue_result.queue_url

# Set long polling for an existing queue.
begin
  existing_queue_name = "existing-queue"
  existing_queue_url = sqs.get_queue_url(queue_name: existing_queue_name).queue_url

  sqs.set_queue_attributes({
    queue_url: existing_queue_url,
    attributes: {
      "ReceiveMessageWaitTimeSeconds" => "20" # Wait 20 seconds to receive messages.
    },
  })
rescue Aws::SQS::Errors::NonExistentQueue
  puts "Cannot set long polling for a queue named '#{existing_queue_name}', as it does
  not exist."
end

# Set long polling when receiving messages for a queue.

# 1. Using receive_message.
begin
  receive_queue_name = "receive-queue"
  receive_queue_url = sqs.get_queue_url(queue_name: receive_queue_name).queue_url

  puts "Begin receipt of any messages using receive_message..."
  receive_message_result = sqs.receive_message({
    queue_url: receive_queue_url,
    attribute_names: ["All"], # Receive all available built-in message attributes.
    message_attribute_names: ["All"], # Receive any custom message attributes.
    max_number_of_messages: 10 # Receive up to 10 messages, if there are that many.
  })

  puts "Received #{receive_message_result.messages.count} message(s)."
rescue Aws::SQS::Errors::NonExistentQueue
  puts "Cannot receive messages using receive_message for a queue named
  '#{receive_queue_name}', as it does not exist."
end

# 2. Using Aws::SQS::QueuePoller.
begin
  puts "Begin receipt of any messages using Aws::SQS::QueuePoller..."
```

```

puts "(Will keep polling until no more messages available for at least 60 seconds.)"
poller = Aws::SQS::QueuePoller.new(receive_queue_url)

poller_stats = poller.poll({
  max_number_of_messages: 10,
  idle_timeout: 60 # Stop polling after 60 seconds of no more messages available
(polls indefinitely by default).
}) do |messages|
  messages.each do |message|
    puts "Message body: #{message.body}"
  end
end
# Note: If poller.poll is successful, all received messages are automatically deleted
from the queue.

puts "Poller stats:"
puts "  Polling started at: #{poller_stats.polling_started_at}"
puts "  Polling stopped at: #{poller_stats.polling_stopped_at}"
puts "  Last message received at: #{poller_stats.last_message_received_at}"
puts "  Number of polling requests: #{poller_stats.request_count}"
puts "  Number of received messages: #{poller_stats.received_message_count}"
rescue Aws::SQS::Errors::NonExistentQueue
  puts "Cannot receive messages using Aws::SQS::QueuePoller for a queue named
'#{receive_queue_name}', as it does not exist."
end

```

Receiving Messages Using the QueuePoller Class in Amazon SQS

The following example uses the `QueuePoller` utility class to display the body of all messages in the Amazon SQS queue with the URL `URL` in the `us-west-2` region, and deletes the message. After approximately 15 seconds of inactivity, the script times out.

```

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

```

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

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

poller = Aws::SQS::QueuePoller.new(URL)

poller.poll(idle_timeout: 15) do |msg|
  puts msg.body
end
```

The following example loops through the Amazon SQS queue with the URL `URL`, and waits up to *duration* seconds.

You can get the correct URL by executing the Amazon SQS example in [Getting Information about All Queues in Amazon SQS](#).

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

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

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

poller = Aws::SQS::QueuePoller.new(URL)

poller.poll(wait_time_seconds: duration, idle_timeout: duration + 1) do |msg|
  puts msg.body
end
```

The following example loops through the Amazon SQS queue with the URL `URL`, and gives you up to the visibility *timeout* seconds to process the message, represented by the method `do_something`.

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

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

# Process the message
def do_something(msg)
  puts msg.body
end

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

poller = Aws::SQS::QueuePoller.new(URL)

poller.poll(visibility_timeout: timeout, idle_timeout: timeout + 1) do |msg|
  do_something(msg)
end
```

The following example loops through the Amazon SQS queue with the URL `URL`, and changes the `visibility timeout` seconds, for any message that needs additional processing by the method `do_something2`.

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

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

# Process the message
def do_something(_)
  true
end

# Do additional processing
def do_something2(msg)
  puts msg.body
end

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

poller = Aws::SQS::QueuePoller.new(URL)

poller.poll(idle_timeout: timeout + 1) do |msg|
  if do_something(msg)
    # need more time for processing
    poller.change_message_visibility_timeout(msg, timeout)

    do_something2(msg)
  end
end
```

Redirecting Dead Letters in Amazon SQS

The following example redirects any dead letters from the queue with the URL URL to the queue with the ARN ARN.

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



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

sqs = Aws::SQS::Client.new(region: 'us-west-2')

sqs.set_queue_attributes({
  queue_url: URL,
  attributes:
    {
      'RedrivePolicy' => "{\"maxReceiveCount\": \"5\", \"deadLetterTargetArn\": \"#{ARN}\"}"
    }
})
```

Deleting a Queue in Amazon SQS

The following example deletes the Amazon SQS queue with the URL `URL` in the `us-west-2` region.

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

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

sqs = Aws::SQS::Client.new(region: 'us-west-2')

sqs.delete_queue(queue_url: URL)
```

Enabling a Resource to Publish to a Queue in Amazon SQS

The following example enables the resource with the ARN `my-resource-arn` to publish to the queue with the ARN `my-queue-arn` and URL `my-queue-url` in the `us-west-2` region.

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

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

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

sqs = Aws::SQS::Client.new(region: 'us-west-2')

policy = '{
  "Version":"2008-10-17",
  "Id":"' + my-queue-arn + '/SQSDefaultPolicy",
  "Statement":[{
    "Sid":"__default_statement_ID",
    "Effect":"Allow",
    "Principal":{"
      "AWS":"*"
    },
    "Action":["SQS:SendMessage"],
    "Resource":"' + my-queue-arn + '",
    "Condition":{"
      "ArnEquals":{"
        "AWS:SourceArn":"' + my-resource-arn + '"}
      }
    }
  ]
}'

sqs.set_queue_attributes({
  queue_url: my-queue-url,
  attributes: {
    Policy: policy
  }
})
```

Working with a Dead Letter Queue in Amazon SQS

Amazon SQS provides support for dead letter queues. A dead letter queue is a queue that other (source) queues can target for messages that can't be processed successfully. You can set aside and

isolate these messages in the dead letter queue to determine why their processing didn't succeed. For more information about dead letter queues, see [Using Amazon SQS Dead Letter Queues](#).

In this example, you use the AWS SDK for Ruby with Amazon SQS to:

1. Create a queue that represents a dead letter queue by using [Aws::SQS::Client#create_queue](#).
2. Associate the dead letter queue with an existing queue by using [Aws::SQS::Client#set_queue_attributes](#).
3. Send a message to the existing queue by using [Aws::SQS::Client#send_message](#).
4. Poll the queue by using [Aws::SQS::QueuePoller](#).
5. Receive messages in the dead letter queue by using [Aws::SQS::Client#receive_message](#).

Prerequisites

Before running the example code, you need to install and configure the AWS SDK for Ruby, as described in:

- [Installing the AWS SDK for Ruby](#)
- [Configuring the AWS SDK for Ruby](#)

You also need to use the AWS Management Console to create the existing queue, *my-queue*.

Note

For the sake of simplicity, this example code doesn't demonstrate [Aws::SQS::Client#add_permission](#). In a real-world scenario, you should always restrict access to actions such as `SendMessage`, `ReceiveMessage`, `DeleteMessage`, and `DeleteQueue`. Not doing so could cause information disclosure, denial of service, or injection of messages into your queues.

Example

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

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

# Demonstrates how to:
# 1. Create a queue representing a dead letter queue.
# 2. Associate the dead letter queue with an existing queue.

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

# Uncomment for Windows.
# Aws.use_bundled_cert!

sqs = Aws::SQS::Client.new(region: 'us-east-1')

# Create a queue representing a dead letter queue.
dead_letter_queue_name = "dead-letter-queue"

sqs.create_queue({
  queue_name: dead_letter_queue_name
})

# Get the dead letter queue's URL and ARN, so that you can associate it with an
# existing queue.
dead_letter_queue_url = sqs.get_queue_url(queue_name: dead_letter_queue_name).queue_url

dead_letter_queue_arn = sqs.get_queue_attributes({
  queue_url: dead_letter_queue_url,
  attribute_names: ["QueueArn"]
}).attributes["QueueArn"]

# Associate the dead letter queue with an existing queue.
begin
  queue_name = "my-queue"
  queue_url = sqs.get_queue_url(queue_name: queue_name).queue_url

  # Use a redrive policy to specify the dead letter queue and its behavior.
  redrive_policy = {
    "maxReceiveCount" => "5", # After the queue receives the same message 5 times, send
    # that message to the dead letter queue.
    "deadLetterTargetArn" => dead_letter_queue_arn
  }
```

```
}.to_json

sqs.set_queue_attributes({
  queue_url: queue_url,
  attributes: {
    "RedrivePolicy" => redrive_policy
  }
})

rescue Aws::SQS::Errors::NonExistentQueue
  puts "A queue named '#{queue_name}' does not exist."
  exit(false)
end

# Send a message to the queue.
puts "Sending a message..."

sqs.send_message({
  queue_url: queue_url,
  message_body: "I hope I get moved to the dead letter queue."
})

30.downto(0) do |i|
  print "\rWaiting #{i} second(s) for sent message to be receivable..."
  sleep(1)
end

puts "\n"

poller = Aws::SQS::QueuePoller.new(queue_url)
# Receive 5 messages max and stop polling after 20 seconds of no received messages.
poller.poll(max_number_of_messages:5, idle_timeout: 20) do |messages|
  messages.each do |msg|
    puts "Received message ID: #{msg.message_id}"
  end
end

# Check to see if Amazon SQS moved the message to the dead letter queue.
receive_message_result = sqs.receive_message({
  queue_url: dead_letter_queue_url,
  max_number_of_messages: 1
})

if receive_message_result.messages.count > 0
```

```
puts "\n#{receive_message_result.messages[0].body}"
else
  puts "\nNo messages received."
end
```

Specifying the Message Visibility Timeout in Amazon SQS

In Amazon SQS, immediately after a message is received, it remains in the queue. To prevent other consumers from processing the message again, Amazon SQS sets a visibility timeout. This is a period of time during which Amazon SQS prevents other consuming components from receiving and processing the message. To learn more, see [Visibility Timeout](#).

In this example, you use the AWS SDK for Ruby with Amazon SQS to:

1. Get the URL of an existing queue by using [Aws::SQS::Client#get_queue_url](#).
2. Receive up to 10 messages by using [Aws::SQS::Client#receive_message](#).
3. Specify the time interval during which messages are not visible after they are received, by using [Aws::SQS::Client#change_message_visibility](#).

Prerequisites

Before running the example code, you need to install and configure the AWS SDK for Ruby, as described in:

- [Installing the AWS SDK for Ruby](#)
- [Configuring the AWS SDK for Ruby](#)

You also need to create the queue *my-queue*, which you can do in the Amazon SQS console.

Example

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

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

# Demonstrates how to specify the time interval during which messages to a queue are
# not visible after being received.

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

sqs = Aws::SQS::Client.new(region: 'us-east-1')

begin
  queue_name = "my-queue"
  queue_url = sqs.get_queue_url(queue_name: queue_name).queue_url

  receive_message_result_before = sqs.receive_message({
    queue_url: queue_url,
    max_number_of_messages: 10 # Receive up to 10 messages, if there are that many.
  })

  puts "Before attempting to change message visibility timeout: received
#{receive_message_result_before.messages.count} message(s)."
```

Amazon WorkDocs Examples

You can use the following examples to access Amazon WorkDocs (Amazon WorkDocs) using the AWS SDK for Ruby. For more information about Amazon WorkDocs, see the [Amazon WorkDocs documentation](#).

You need your organization ID to use these examples. Get your organization ID from the AWS console using the following steps:

- Select the AWS Directory Service
- Select `Directories`

The organization ID is the `Directory` ID corresponding to your Amazon WorkDocs site.

Examples

Topics

- [Listing Users](#)
- [Listing User Docs](#)

Listing Users

The following example lists the names, email addresses, and root folders of all users in the organization. Choose Copy to save the code locally, or see the link to the complete example at the end of this topic.

1. Require the AWS SDK for Ruby module and create an Amazon WorkDocs client.
 2. Call `describe_users` with your organization ID, and get all of the user names in ascending order.
1. Display the information about the users.

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



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

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

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

# Set to the OrganizationId of your WorkDocs site
orgId = 'd-123456789c'

resp = client.describe_users({
  organization_id: orgId,
  include: "ALL", # accepts ALL, ACTIVE_PENDING
  order: "ASCENDING", # accepts ASCENDING, DESCENDING
  sort: "USER_NAME", # accepts USER_NAME, FULL_NAME, STORAGE_LIMIT, USER_STATUS,
  STORAGE_USED
})

resp.users.each do |user|
  puts "First name:  #{user.given_name}"
  puts "Last name:   #{user.surname}"
  puts "Email:       #{user.email_address}"
  puts "Root folder: #{user.root_folder_id}"
  puts
end
```

See the [complete example](#) on GitHub.

Listing User Docs

The following example lists the documents for a user. Choose Copy to save the code locally, or see the link to the complete example at the end of this topic.

1. Require the AWS SDK for Ruby module.
2. Create a helper method to get the root folder of a user.

3. Create a Amazon WorkDocs client.
4. Get the root folder for that user.
5. Call `describe_folder_contents` to get the contents of the folder in ascending order.
6. Display the name, size (in bytes), last modified date, document ID and version ID for each document in the user's root folder.

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

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

def get_user_folder(client, orgId, user_email)
  root_folder = ''

  resp = client.describe_users({
    organization_id: orgId,
  })

  # resp.users should have only one entry
  resp.users.each do |user|
    if user.email_address == user_email
      root_folder = user.root_folder_id
    end
  end

  return root_folder
end

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

# Set to the email address of a user
```

```
user_email = 'someone@somewhere'

# Set to the OrganizationId of your WorkDocs site.
orgId = 'd-123456789c'

user_folder = get_user_folder(client, orgId, user_email)

if user_folder == ''
  puts 'Could not get root folder for user with email address ' + user_email
  exit(1)
end

resp = client.describe_folder_contents({
  folder_id: user_folder, # required
  sort: "NAME", # accepts DATE, NAME
  order: "ASCENDING", # accepts ASCENDING, DESCENDING
})

resp.documents.each do |doc|
  md = doc.latest_version_metadata

  puts "Name:           #{md.name}"
  puts "Size (bytes):    #{md.size}"
  puts "Last modified:    #{doc.modified_timestamp}"
  puts "Doc ID:           #{doc.id}"
  puts "Version ID:       #{md.id}"
  puts
end
```

See the [complete example](#) on GitHub.

SDK for Ruby code examples

The code examples in this topic show you how to use the AWS SDK for Ruby with AWS.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Cross-service examples are sample applications that work across multiple AWS services.

Examples

- [Actions and scenarios using SDK for Ruby](#)
- [Cross-service examples using SDK for Ruby](#)

Actions and scenarios using SDK for Ruby

The following code examples show how to perform actions and implement common scenarios by using the AWS SDK for Ruby with AWS services.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Services

- [CloudTrail examples using SDK for Ruby](#)
- [CloudWatch examples using SDK for Ruby](#)
- [DynamoDB examples using SDK for Ruby](#)
- [Amazon EC2 examples using SDK for Ruby](#)
- [Elastic Beanstalk examples using SDK for Ruby](#)

- [EventBridge examples using SDK for Ruby](#)
- [AWS Glue examples using SDK for Ruby](#)
- [IAM examples using SDK for Ruby](#)
- [Kinesis examples using SDK for Ruby](#)
- [AWS KMS examples using SDK for Ruby](#)
- [Lambda examples using SDK for Ruby](#)
- [Amazon Polly examples using SDK for Ruby](#)
- [Amazon RDS examples using SDK for Ruby](#)
- [Amazon S3 examples using SDK for Ruby](#)
- [Amazon SES examples using SDK for Ruby](#)
- [Amazon SES API v2 examples using SDK for Ruby](#)
- [Amazon SNS examples using SDK for Ruby](#)
- [Amazon SQS examples using SDK for Ruby](#)
- [AWS STS examples using SDK for Ruby](#)
- [Amazon WorkDocs examples using SDK for Ruby](#)

CloudTrail examples using SDK for Ruby

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Ruby with CloudTrail.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)

Actions

CreateTrail

The following code example shows how to use CreateTrail.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-cloudtrail" # v2: require 'aws-sdk'
require "aws-sdk-s3"
require "aws-sdk-sts"

def create_trail_example(s3_client, sts_client, cloudtrail_client, trail_name,
  bucket_name)

  resp = sts_client.get_caller_identity({})
  account_id = resp.account

  # Attach policy to an Amazon Simple Storage Service (S3) bucket.
  s3_client.create_bucket(bucket: bucket_name)
  begin
    policy = {
      "Version" => "2012-10-17",
      "Statement" => [
        {
          "Sid" => "AWSCloudTrailAclCheck20150319",
          "Effect" => "Allow",
          "Principal" => {
            "Service" => "cloudtrail.amazonaws.com"
          },
          "Action" => "s3:GetBucketAcl",
          "Resource" => "arn:aws:s3:::#{bucket_name}"
        },
        {
          "Sid" => "AWSCloudTrailWrite20150319",
          "Effect" => "Allow",
```

```

    "Principal" => {
      "Service" => "cloudtrail.amazonaws.com"
    },
    "Action" => "s3:PutObject",
    "Resource" => "arn:aws:s3:::#{bucket_name}/AWSLogs/#{account_id}/*",
    "Condition" => {
      "StringEquals" => {
        "s3:x-amz-acl" => "bucket-owner-full-control"
      }
    }
  }
]
}.to_json

s3_client.put_bucket_policy(
  bucket: bucket_name,
  policy: policy
)
puts "Successfully added policy to bucket #{bucket_name}"
end

begin
  cloudtrail_client.create_trail({
                                name: trail_name, # required
                                s3_bucket_name: bucket_name # required
  })

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

```

- For API details, see [CreateTrail](#) in *AWS SDK for Ruby API Reference*.

DeleteTrail

The following code example shows how to use DeleteTrail.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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
```

- For API details, see [DeleteTrail](#) in *AWS SDK for Ruby API Reference*.

ListTrails

The following code example shows how to use `ListTrails`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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
```

- For API details, see [ListTrails](#) in *AWS SDK for Ruby API Reference*.

LookupEvents

The following code example shows how to use LookupEvents.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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
```

- For API details, see [LookupEvents](#) in *AWS SDK for Ruby API Reference*.

CloudWatch examples using SDK for Ruby

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Ruby with CloudWatch.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)

Actions

DescribeAlarms

The following code example shows how to use DescribeAlarms.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-cloudwatch"

# Lists the names of available Amazon CloudWatch alarms.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
```

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

- For API details, see [DescribeAlarms](#) in *AWS SDK for Ruby API Reference*.

DescribeAlarmsForMetric

The following code example shows how to use DescribeAlarmsForMetric.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
#
# @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
    end
  end
end
```

```
puts "Name:           " + alarm.alarm_name
puts "State value:    " + alarm.state_value
puts "State reason:   " + alarm.state_reason
puts "Metric:         " + alarm.metric_name
puts "Namespace:     " + alarm.namespace
puts "Statistic:      " + alarm.statistic
puts "Period:         " + alarm.period.to_s
puts "Unit:           " + alarm.unit.to_s
puts "Eval. periods:  " + alarm.evaluation_periods.to_s
puts "Threshold:      " + alarm.threshold.to_s
puts "Comp. operator: " + alarm.comparison_operator

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

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

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

puts "Dimensions:"
if alarm.key?(:dimensions) && alarm.dimensions.count.positive?
  alarm.dimensions.each do |d|
    puts "  Name: " + d.name + ", Value: " + d.value
  end
else
  puts "  None for this alarm."
end
end
else
```

```
    puts "No alarms found."
  end
rescue StandardError => e
  puts "Error getting information about alarms: #{e.message}"
end

# Example usage:
def run_me
  region = ""

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

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

run_me if $PROGRAM_NAME == __FILE__
```

- For API details, see [DescribeAlarmsForMetric](#) in *AWS SDK for Ruby API Reference*.

DisableAlarmActions

The following code example shows how to use `DisableAlarmActions`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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__
end

```

- For API details, see [DisableAlarmActions](#) in *AWS SDK for Ruby API Reference*.

ListMetrics

The following code example shows how to use ListMetrics.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

# 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
      end
    end
  else

```



```
        puts "No dimensions found."
      end
    end
  end
else
  puts "No metrics found for namespace '#{metric_namespace}'. " \
    "Note that it could take up to 15 minutes for recently-added metrics " \
    "to become available."
end
end
end

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

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

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

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

  puts "Continuing..." unless datapoint_added_to_metric?(
    cloudwatch_client,
    metric_namespace,
    "PageViews",
    "PageURL",
```

```

    "example.html",
    18_057.0,
    "Count"
  )

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

run_me if $PROGRAM_NAME == __FILE__

```

- For API details, see [ListMetrics](#) in *AWS SDK for Ruby API Reference*.

PutMetricAlarm

The following code example shows how to use PutMetricAlarm.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

# 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
```

- For API details, see [PutMetricAlarm](#) in *AWS SDK for Ruby API Reference*.

PutMetricData

The following code example shows how to use PutMetricData.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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
      }
    ]
  )
end
```

```
    }
  ]
)
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
```

- For API details, see [PutMetricData](#) in *AWS SDK for Ruby API Reference*.

DynamoDB examples using SDK for Ruby

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Ruby with DynamoDB.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)
- [Scenarios](#)
- [Serverless examples](#)

Actions

BatchExecuteStatement

The following code example shows how to use BatchExecuteStatement.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Read a batch of items using 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
```

Delete a batch of items using 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

```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for Ruby API Reference*.

BatchWriteItem

The following code example shows how to use BatchWriteItem.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

class DynamoDBBasics
  attr_reader :dynamo_resource
  attr_reader :table

  def initialize(table_name)

```



```

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

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

```

- For API details, see [BatchWriteItem](#) in *AWS SDK for Ruby API Reference*.

CreateTable

The following code example shows how to use CreateTable.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [CreateTable](#) in *AWS SDK for Ruby API Reference*.

DeleteItem

The following code example shows how to use DeleteItem.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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
```

- For API details, see [DeleteItem](#) in *AWS SDK for Ruby API Reference*.

DeleteTable

The following code example shows how to use DeleteTable.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [DeleteTable](#) in *AWS SDK for Ruby API Reference*.

DescribeTable

The following code example shows how to use DescribeTable.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [DescribeTable](#) in *AWS SDK for Ruby API Reference*.

ExecuteStatement

The following code example shows how to use `ExecuteStatement`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Select a single item using 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

```

Update a single item using 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

```

Add a single item using 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
end

```

Delete a single item using 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]
    }
  end
end

```



```

    }
    @dynamodb.client.execute_statement(request)
  end

```

- For API details, see [ExecuteStatement](#) in *AWS SDK for Ruby API Reference*.

GetItem

The following code example shows how to use GetItem.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

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

```

- For API details, see [GetItem](#) in *AWS SDK for Ruby API Reference*.

ListTables

The following code example shows how to use `ListTables`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Determine whether a table exists.

```
# 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

```

- For API details, see [ListTables](#) in *AWS SDK for Ruby API Reference*.

PutItem

The following code example shows how to use PutItem.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

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

```

- For API details, see [PutItem](#) in *AWS SDK for Ruby API Reference*.

Query

The following code example shows how to use Query.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

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
```

- For API details, see [Query](#) in *AWS SDK for Ruby API Reference*.

Scan

The following code example shows how to use Scan.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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

```

- For API details, see [Scan](#) in *AWS SDK for Ruby API Reference*.

UpdateItem

The following code example shows how to use UpdateItem.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

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
```

- For API details, see [UpdateItem](#) in *AWS SDK for Ruby API Reference*.

Scenarios

Get started with tables, items, and queries

The following code example shows how to:

- Create a table that can hold movie data.
- Put, get, and update a single movie in the table.
- Write movie data to the table from a sample JSON file.
- Query for movies that were released in a given year.
- Scan for movies that were released in a range of years.

- Delete a movie from the table, then delete the table.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a class that encapsulates a DynamoDB table.

```
# 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
```

Create a helper function to download and extract the sample JSON file.

```
# 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

```

Run an interactive scenario to create the table and perform actions on it.

```

table_name = "doc-example-table-movies-#{rand(10**4)}"
scaffold = Scaffold.new(table_name)
dynamodb_wrapper = DynamoDBBasics.new(table_name)

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

```

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

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

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

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

new_step(5, "Query for a batch of items by key.")
loop do
  release_year = CLI::UI::Prompt.ask("Enter a year between 1972 and 2018, e.g.
1999:").to_i
  results = dynamodb_wrapper.query_items(release_year)
  if results.any?
    puts("There were #{results.length} movies released in #{release_year}:")
```

```

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

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

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

new_step(8, "Delete the DynamoDB table.")
answer = CLI::UI::Prompt.ask("Delete the table? (y/n)")

```

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

- For API details, see the following topics in *AWS SDK for Ruby API Reference*.
 - [BatchWriteItem](#)
 - [CreateTable](#)
 - [DeleteItem](#)
 - [DeleteTable](#)
 - [DescribeTable](#)
 - [GetItem](#)
 - [PutItem](#)
 - [Query](#)
 - [Scan](#)
 - [UpdateItem](#)

Query a table by using batches of PartiQL statements

The following code example shows how to:

- Get a batch of items by running multiple SELECT statements.
- Add a batch of items by running multiple INSERT statements.
- Update a batch of items by running multiple UPDATE statements.
- Delete a batch of items by running multiple DELETE statements.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Run a scenario that creates a table and runs batch PartiQL queries.

```
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
end
```

- For API details, see [BatchExecuteStatement](#) in *AWS SDK for Ruby API Reference*.

Query a table using PartiQL

The following code example shows how to:

- Get an item by running a SELECT statement.
- Add an item by running an INSERT statement.
- Update an item by running an UPDATE statement.
- Delete an item by running a DELETE statement.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Run a scenario that creates a table and runs PartiQL queries.

```
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
```

- For API details, see [ExecuteStatement](#) in *AWS SDK for Ruby API Reference*.

Serverless examples

Invoke a Lambda function from a DynamoDB trigger

The following code example shows how to implement a Lambda function that receives an event triggered by receiving records from a DynamoDB stream. The function retrieves the DynamoDB payload and logs the record contents.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Serverless examples](#) repository.

Consuming a DynamoDB event with Lambda using Ruby.

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

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
```


Reporting batch item failures for Lambda functions with a DynamoDB trigger

The following code example shows how to implement partial batch response for Lambda functions that receive events from a DynamoDB stream. The function reports the batch item failures in the response, signaling to Lambda to retry those messages later.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Serverless examples](#) repository.

Reporting DynamoDB batch item failures with Lambda using Ruby.

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

Amazon EC2 examples using SDK for Ruby

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Ruby with Amazon EC2.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)

Actions

AllocateAddress

The following code example shows how to use `AllocateAddress`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [AllocateAddress](#) in *AWS SDK for Ruby API Reference*.

AssociateAddress

The following code example shows how to use AssociateAddress.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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

```

- For API details, see [AssociateAddress](#) in *AWS SDK for Ruby API Reference*.

CreateKeyPair

The following code example shows how to use CreateKeyPair.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

# 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
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
end
```

```
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__
```

- For API details, see [CreateKeyPair](#) in *AWS SDK for Ruby API Reference*.

CreateRouteTable

The following code example shows how to use `CreateRouteTable`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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__
```

- For API details, see [CreateRouteTable](#) in *AWS SDK for Ruby API Reference*.

CreateSecurityGroup

The following code example shows how to use CreateSecurityGroup.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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?(:ipv6_ranges) && perm.ipv6_ranges.count.positive?
    print ", CIDR IPv6: #{perm.ipv6_ranges[0].cidr_ipv6}"
  end

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

  print "\n"
end

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

```

```
response = ec2_client.describe_security_groups

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

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

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

    unless sg.ip_permissions_egress.empty?
      puts "Outbound rules:" if sg.ip_permissions_egress.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"
  end
  exit 1
end

```



```
# 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__
```

- For API details, see [CreateSecurityGroup](#) in *AWS SDK for Ruby API Reference*.

CreateSubnet

The following code example shows how to use CreateSubnet.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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_value [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__
```

- For API details, see [CreateSubnet](#) in *AWS SDK for Ruby API Reference*.

CreateVpc

The following code example shows how to use CreateVpc.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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
end
```

```
end

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

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

run_me if $PROGRAM_NAME == __FILE__
```

- For API details, see [CreateVpc](#) in *AWS SDK for Ruby API Reference*.

DescribeInstances

The following code example shows how to use DescribeInstances.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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__
```

- For API details, see [DescribeInstances](#) in *AWS SDK for Ruby API Reference*.

DescribeRegions

The following code example shows how to use DescribeRegions.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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__
```

- For API details, see [DescribeRegions](#) in *AWS SDK for Ruby API Reference*.

ReleaseAddress

The following code example shows how to use ReleaseAddress.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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

```

- For API details, see [ReleaseAddress](#) in *AWS SDK for Ruby API Reference*.

StartInstances

The following code example shows how to use StartInstances.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-ec2"
```

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

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

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

# Example usage:
def run_me
```

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

- For API details, see [StartInstances](#) in *AWS SDK for Ruby API Reference*.

StopInstances

The following code example shows how to use StopInstances.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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__
```

- For API details, see [StopInstances](#) in *AWS SDK for Ruby API Reference*.

TerminateInstances

The following code example shows how to use `TerminateInstances`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-ec2"

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

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

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

  ec2_client.terminate_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_terminated, instance_ids: [instance_id])
  puts "Instance terminated."
```

```

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

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

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

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

  run_me if $PROGRAM_NAME == __FILE__

```

- For API details, see [TerminateInstances](#) in *AWS SDK for Ruby API Reference*.

Elastic Beanstalk examples using SDK for Ruby

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Ruby with Elastic Beanstalk.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)

Actions

DescribeApplications

The following code example shows how to use DescribeApplications.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

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

private

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

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

- For API details, see [DescribeApplications](#) in *AWS SDK for Ruby API Reference*.

ListAvailableSolutionStacks

The following code example shows how to use `ListAvailableSolutionStacks`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

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

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

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

  private

  # Logs summary of listed stacks
  def log_summary(filtered_length, orig_length)
```

```
    if @filter.empty?
      @logger.info("Showed #{orig_length} stack(s)")
    else
      @logger.info("Showed #{filtered_length} stack(s) of #{orig_length}")
    end
  end
end
```

- For API details, see [ListAvailableSolutionStacks](#) in *AWS SDK for Ruby API Reference*.

UpdateApplication

The following code example shows how to use UpdateApplication.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

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

private
```

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

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

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

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

# Creates a new application version and deploys it
def create_and_deploy_new_application_version(zip_file_name)
  version_label = File.basename(zip_file_name, ".zip")
  @eb_client.create_application_version(
    process: false,
    application_name: @app_name,
    version_label: version_label,
    source_bundle: {
```



```

        s3_bucket: fetch_bucket_name,
        s3_key: "#{@app_name}/#{zip_file_name}"
      },
      description: "Updated #{Time.now.strftime('%d/%m/%Y')}}"
    )
    update_environment(version_label)
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to create or deploy application version: #{e.message}")
  end

  # Updates the environment to the new application version
  def update_environment(version_label)
    env_name = fetch_environment_name
    @eb_client.update_environment(
      environment_name: env_name,
      version_label: version_label
    )
  rescue Aws::Errors::ServiceError => e
    @logger.error("Failed to update environment: #{e.message}")
  end

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

```

- For API details, see [UpdateApplication](#) in *AWS SDK for Ruby API Reference*.

EventBridge examples using SDK for Ruby

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Ruby with EventBridge.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Scenarios](#)

Scenarios

Create and trigger a rule

The following code example shows how to create and trigger a rule in Amazon EventBridge.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Call the functions in the correct order.

```
require "aws-sdk-sns"
require "aws-sdk-iam"
require "aws-sdk-cloudwatchevents"
require "aws-sdk-ec2"
require "aws-sdk-cloudwatch"
require "aws-sdk-cloudwatchlogs"
require "securerandom"
```

Checks whether the specified Amazon Simple Notification Service (Amazon SNS) topic exists among those provided to this function.

```
# Checks whether the specified Amazon SNS
# topic exists among those provided to this function.
# This is a helper function that is called by the topic_exists? function.
```

```

#
# @param topics [Array] An array of Aws::SNS::Types::Topic objects.
# @param topic_arn [String] The ARN of the topic to find.
# @return [Boolean] true if the topic ARN was found; otherwise, false.
# @example
#   sns_client = Aws::SNS::Client.new(region: 'us-east-1')
#   response = sns_client.list_topics
#   if topic_found?(
#     response.topics,
#     'arn:aws:sns:us-east-1:111111111111:aws-doc-sdk-examples-topic'
#   )
#     puts 'Topic found.'
#   end

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

```

Checks whether the specified topic exists among those available to the caller in Amazon SNS.

```

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

```

Create a topic in Amazon SNS and then subscribe an email address to receive notifications to that topic.

```

# 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
  )
end

```

```

)
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

```

Check whether the specified AWS Identity and Access Management (IAM) role exists among those provided to this function.

```

# 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

```

Check whether the specified role exists among those available to the caller in IAM.

```

# Checks whether the specified role exists among those available to the
# caller in AWS Identity and Access Management (IAM).

```

```

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

```

Create a role in IAM.

```

# Creates a role in AWS Identity and Access Management (IAM).
# This role is used by a rule in Amazon EventBridge to allow
# that rule to operate within the caller's account.
# This role is designed to be used specifically by this code example.
#
# @param iam_client [Aws::IAM::Client] An initialized IAM client.

```

```
# @param role_name [String] The name of the role to create.
# @return [String] The ARN of the role that was created.
# @example
#   puts create_role(
#     Aws::IAM::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-ec2-state-change'
#   )
def create_role(iam_client, role_name)
  puts "Creating the role named '#{role_name}'..."
  response = iam_client.create_role(
    assume_role_policy_document: {
      'Version': "2012-10-17",
      'Statement': [
        {
          'Sid': "",
          'Effect': "Allow",
          'Principal': {
            'Service': "events.amazonaws.com"
          },
          'Action': "sts:AssumeRole"
        }
      ]
    }.to_json,
    path: "/",
    role_name: role_name
  )
  puts "Role created with ARN '#{response.role.arn}'."
  puts "Adding access policy to role..."
  iam_client.put_role_policy(
    policy_document: {
      'Version': "2012-10-17",
      'Statement': [
        {
          'Sid': "CloudWatchEventsFullAccess",
          'Effect': "Allow",
          'Resource': "*",
          'Action': "events:*"
        },
        {
          'Sid': "IAMPassRoleForCloudWatchEvents",
          'Effect': "Allow",
          'Resource': "arn:aws:iam::*:role/AWS_Events_Invoke_Targets",
          'Action': "iam:PassRole"
        }
      ]
    }
  )
end
```

```

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

```

Checks whether the specified EventBridge rule exists among those provided to this function.

```

# 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

```

Checks whether the specified rule exists among those available to the caller in EventBridge.

```

# Checks whether the specified rule exists among those available to the
# caller in Amazon EventBridge.
#

```



```

# @param cloudwatchevents_client [Aws::CloudWatchEvents::Client]
#   An initialized Amazon EventBridge client.
# @param rule_name [String] The name of the rule to find.
# @return [Boolean] true if the rule name was found; otherwise, false.
# @example
#   exit 1 unless rule_exists?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1')
#     'aws-doc-sdk-examples-ec2-state-change'
#   )
def rule_exists?(cloudwatchevents_client, rule_name)
  puts "Searching for rule with name '#{rule_name}'..."
  response = cloudwatchevents_client.list_rules
  if response.rules.count.positive?
    if rule_found?(response.rules, rule_name)
      puts "Rule found."
      return true
    end
  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

```

Create a rule in EventBridge.

```

# Creates a rule in Amazon EventBridge.
# This rule is triggered whenever an available instance in
# Amazon EC2 changes to the specified state.
# This rule is designed to be used specifically by this code example.
#
# Prerequisites:

```

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

```

    "EC2 Instance State-change Notification"
  ],
  'detail': {
    'state': [
      instance_state
    ]
  }
}.to_json,
state: "ENABLED",
role_arn: role_arn
)
puts "Rule created with ARN '#{put_rule_response.rule_arn}'."

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

```

Check to see whether the specified log group exists among those available to the caller in Amazon CloudWatch Logs.

```

# Checks to see whether the specified log group exists among those available
# to the caller in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group to find.
# @return [Boolean] true if the log group name was found; otherwise, false.
# @example
#   exit 1 unless log_group_exists?(
#     Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#     'aws-doc-sdk-examples-cloudwatch-log'
#   )
def log_group_exists?(cloudwatchlogs_client, log_group_name)
  puts "Searching for log group with name '#{log_group_name}'..."
  response = cloudwatchlogs_client.describe_log_groups(
    log_group_name_prefix: log_group_name
  )
  if response.log_groups.count.positive?
    response.log_groups.each do |log_group|
      if log_group.log_group_name == log_group_name
        puts "Log group found."
        return true
      end
    end
  end
  puts "Log group not found."
  return false
rescue StandardError => e
  puts "Log group not found: #{e.message}"
  return false
end

```

Create a log group in CloudWatch Logs.

```

# Creates a log group in Amazon CloudWatch Logs.
#
# @param cloudwatchlogs_client [Aws::CloudWatchLogs::Client] An initialized
#   Amazon CloudWatch Logs client.
# @param log_group_name [String] The name of the log group to create.
# @return [Boolean] true if the log group name was created; otherwise, false.
# @example
#   exit 1 unless log_group_created?(

```

```

#   Aws::CloudWatchLogs::Client.new(region: 'us-east-1'),
#   'aws-doc-sdk-examples-cloudwatch-log'
#   )
def log_group_created?(cloudwatchlogs_client, log_group_name)
  puts "Attempting to create log group with the name '#{log_group_name}'..."
  cloudwatchlogs_client.create_log_group(log_group_name: log_group_name)
  puts "Log group created."
  return true
rescue StandardError => e
  puts "Error creating log group: #{e.message}"
  return false
end

```

Write an event to a log stream in CloudWatch Logs.

```

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

```

```
# )
def log_event(
  cloudwatchlogs_client,
  log_group_name,
  log_stream_name,
  message,
  sequence_token
)
  puts "Attempting to log '#{message}' to log stream '#{log_stream_name}'..."
  event = {
    log_group_name: log_group_name,
    log_stream_name: log_stream_name,
    log_events: [
      {
        timestamp: (Time.now.utc.to_f.round(3) * 1_000).to_i,
        message: message
      }
    ]
  }
  unless sequence_token.empty?
    event[:sequence_token] = sequence_token
  end

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

Restart an Amazon Elastic Compute Cloud (Amazon EC2) instance and adds information about the related activity to a log stream in CloudWatch Logs.

```
# Restarts an Amazon EC2 instance
# and adds information about the related activity to a log stream
# in Amazon CloudWatch Logs.
#
# Prerequisites:
#
# - The Amazon EC2 instance to restart.
# - The log group in Amazon CloudWatch Logs to add related activity
```

```

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

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

  puts "Attempting to restart the instance. This might take a few minutes..."

```

```

ec2_client.start_instances(instance_ids: [instance_id])
ec2_client.wait_until(:instance_running, instance_ids: [instance_id])
puts "Instance restarted."
sequence_token = log_event(
  cloudwatchlogs_client,
  log_group_name,
  log_stream_name,
  "Instance '#{instance_id}' restarted.",
  sequence_token
)

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

```

Display information about activity for a rule in EventBridge.

```

# Displays information about activity for a rule in Amazon EventBridge.
#
# Prerequisites:
#
# - A rule in Amazon EventBridge.
#
# @param cloudwatch_client [Amazon::CloudWatch::Client] An initialized
#   Amazon CloudWatch client.
# @param rule_name [String] The name of the rule.
# @param start_time [Time] The timestamp that determines the first datapoint
#   to return. Can also be expressed as DateTime, Date, Integer, or String.
# @param end_time [Time] The timestamp that determines the last datapoint
#   to return. Can also be expressed as DateTime, Date, Integer, or String.
# @param period [Integer] The interval, in seconds, to check for activity.
# @example

```



```
# display_rule_activity(
#   Aws::CloudWatch::Client.new(region: 'us-east-1'),
#   'aws-doc-sdk-examples-ec2-state-change',
#   Time.now - 600, # Start checking from 10 minutes ago.
#   Time.now, # Check up until now.
#   60 # Check every minute during those 10 minutes.
# )
def display_rule_activity(
  cloudwatch_client,
  rule_name,
  start_time,
  end_time,
  period
)
  puts "Attempting to display rule activity..."
  response = cloudwatch_client.get_metric_statistics(
    namespace: "AWS/Events",
    metric_name: "Invocations",
    dimensions: [
      {
        name: "RuleName",
        value: rule_name
      }
    ],
    start_time: start_time,
    end_time: end_time,
    period: period,
    statistics: ["Sum"],
    unit: "Count"
  )

  if response.key?(:datapoints) && response.datapoints.count.positive?
    puts "The event rule '#{rule_name}' was triggered:"
    response.datapoints.each do |datapoint|
      puts "  #{datapoint.sum} time(s) at #{datapoint.timestamp}"
    end
  else
    puts "The event rule '#{rule_name}' was not triggered during the " \
      "specified time period."
  end
rescue StandardError => e
  puts "Error getting information about event rule activity: #{e.message}"
end
```

Display log information for all of the log streams in a CloudWatch Logs log group.

```
# 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
```

```

        end
      end
    end
  rescue StandardError => e
    puts "Error getting information about the log streams or their messages: " \
        "#{e.message}"
  end
end

```

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

```

# 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
      )
    end
  end
end
```

```
    )
    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
```

```
    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__
```

- For API details, see the following topics in *AWS SDK for Ruby API Reference*.
 - [PutEvents](#)
 - [PutRule](#)

AWS Glue examples using SDK for Ruby

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Ruby with AWS Glue.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)
- [Scenarios](#)

Actions

CreateCrawler

The following code example shows how to use `CreateCrawler`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
end

```

- For API details, see [CreateCrawler](#) in *AWS SDK for Ruby API Reference*.

CreateJob

The following code example shows how to use CreateJob.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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}")
  end
end
```

```
    raise
  end
```

- For API details, see [CreateJob](#) in *AWS SDK for Ruby API Reference*.

DeleteCrawler

The following code example shows how to use DeleteCrawler.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [DeleteCrawler](#) in *AWS SDK for Ruby API Reference*.

DeleteDatabase

The following code example shows how to use DeleteDatabase.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [DeleteDatabase](#) in *AWS SDK for Ruby API Reference*.

DeleteJob

The following code example shows how to use DeleteJob.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [DeleteJob](#) in *AWS SDK for Ruby API Reference*.

DeleteTable

The following code example shows how to use DeleteTable.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [DeleteTable](#) in *AWS SDK for Ruby API Reference*.

GetCrawler

The following code example shows how to use `GetCrawler`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [GetCrawler](#) in *AWS SDK for Ruby API Reference*.

GetDatabase

The following code example shows how to use GetDatabase.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [GetDatabase](#) in *AWS SDK for Ruby API Reference*.

GetJobRun

The following code example shows how to use GetJobRun.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [GetJobRun](#) in *AWS SDK for Ruby API Reference*.

GetJobRuns

The following code example shows how to use GetJobRuns.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [GetJobRuns](#) in *AWS SDK for Ruby API Reference*.

GetTables

The following code example shows how to use `GetTables`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [GetTables](#) in *AWS SDK for Ruby API Reference*.

ListJobs

The following code example shows how to use ListJobs.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [ListJobs](#) in *AWS SDK for Ruby API Reference*.

StartCrawler

The following code example shows how to use StartCrawler.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [StartCrawler](#) in *AWS SDK for Ruby API Reference*.

StartJobRun

The following code example shows how to use StartJobRun.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [StartJobRun](#) in *AWS SDK for Ruby API Reference*.

Scenarios

Get started with crawlers and jobs

The following code example shows how to:

- Create a crawler that crawls a public Amazon S3 bucket and generates a database of CSV-formatted metadata.
- List information about databases and tables in your AWS Glue Data Catalog.
- Create a job to extract CSV data from the S3 bucket, transform the data, and load JSON-formatted output into another S3 bucket.
- List information about job runs, view transformed data, and clean up resources.

For more information, see [Tutorial: Getting started with AWS Glue Studio](#).

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create a class that wraps AWS Glue functions used in the scenario.

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

```
@logger = logger
end

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

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

```
end

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

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

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

# Retrieves a list of tables in the specified database.
#
# @param db_name [String] The name of the database to retrieve tables from.
# @return [Array<Aws::Glue::Types::Table>]
def get_tables(db_name)
  response = @glue_client.get_tables(database_name: db_name)
  response.table_list
end
```



```
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}/"
    }
  )
end
```

```
)
  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
```

Create a class that runs the scenario.

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

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

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

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

print "\nDone!\n".green

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

```

def main

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

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

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

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

  # Instantiate existing S3 bucket.
  s3 = Aws::S3::Resource.new(region: "us-east-1")

```

```

s3_bucket_name = resource_names["glue_bucket"]
s3_bucket = s3.bucket(s3_bucket_name)

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

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

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

end

```

Create an ETL script that is used by AWS Glue to extract, transform, and load data during job runs.

```

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()
```

- For API details, see the following topics in *AWS SDK for Ruby API Reference*.
 - [CreateCrawler](#)
 - [CreateJob](#)
 - [DeleteCrawler](#)
 - [DeleteDatabase](#)
 - [DeleteJob](#)
 - [DeleteTable](#)
 - [GetCrawler](#)
 - [GetDatabase](#)
 - [GetDatabases](#)
 - [GetJob](#)
 - [GetJobRun](#)
 - [GetJobRuns](#)
 - [GetTables](#)
 - [ListJobs](#)
 - [StartCrawler](#)
 - [StartJobRun](#)

IAM examples using SDK for Ruby

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Ruby with IAM.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)
- [Scenarios](#)

Actions

AttachRolePolicy

The following code example shows how to use `AttachRolePolicy`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

This example module lists, creates, attaches, and detaches role policies.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
```

```
def initialize(iam_client, logger: Logger.new($stdout))
  @iam_client = iam_client
  @logger = logger
  @logger.progname = "PolicyManager"
end

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

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

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

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

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

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

- For API details, see [AttachRolePolicy](#) in *AWS SDK for Ruby API Reference*.

AttachUserPolicy

The following code example shows how to use `AttachUserPolicy`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [AttachUserPolicy](#) in *AWS SDK for Ruby API Reference*.

CreateAccessKey

The following code example shows how to use `CreateAccessKey`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

This example module lists, creates, deactivates, and deletes access keys.

```
# 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
```


- For API details, see [CreateAccessKey](#) in *AWS SDK for Ruby API Reference*.

CreateAccountAlias

The following code example shows how to use `CreateAccountAlias`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

List, create, and delete account aliases.

```
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
```

- For API details, see [CreateAccountAlias](#) in *AWS SDK for Ruby API Reference*.

CreatePolicy

The following code example shows how to use CreatePolicy.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

This example module lists, creates, attaches, and detaches role policies.

```
# 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
```

- For API details, see [CreatePolicy](#) in *AWS SDK for Ruby API Reference*.

CreateRole

The following code example shows how to use CreateRole.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [CreateRole](#) in *AWS SDK for Ruby API Reference*.

CreateServiceLinkedRole

The following code example shows how to use `CreateServiceLinkedRole`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [CreateServiceLinkedRole](#) in *AWS SDK for Ruby API Reference*.

CreateUser

The following code example shows how to use CreateUser.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [CreateUser](#) in *AWS SDK for Ruby API Reference*.

DeleteAccessKey

The following code example shows how to use DeleteAccessKey.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

This example module lists, creates, deactivates, and deletes access keys.

```
# 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
```

- For API details, see [DeleteAccessKey](#) in *AWS SDK for Ruby API Reference*.

DeleteAccountAlias

The following code example shows how to use DeleteAccountAlias.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

List, create, and delete account aliases.

```
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
```

- For API details, see [DeleteAccountAlias](#) in *AWS SDK for Ruby API Reference*.

DeleteRole

The following code example shows how to use DeleteRole.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# Deletes a role and its attached policies.
```

```

#
# @param role_name [String] The name of the role to delete.
def delete_role(role_name)
  begin
    # Detach and delete attached policies
    @iam_client.list_attached_role_policies(role_name: role_name).each do |
response|
      response.attached_policies.each do |policy|
        @iam_client.detach_role_policy({
          role_name: role_name,
          policy_arn: policy.policy_arn
        })

        # Check if the policy is a customer managed policy (not AWS managed)
        unless policy.policy_arn.include?("aws:policy/")
          @iam_client.delete_policy({ policy_arn: policy.policy_arn })
          @logger.info("Deleted customer managed policy #{policy.policy_name}.")
        end
      end
    end

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

```

- For API details, see [DeleteRole](#) in *AWS SDK for Ruby API Reference*.

DeleteServerCertificate

The following code example shows how to use DeleteServerCertificate.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

List, update, and delete server certificates.

```
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
```

- For API details, see [DeleteServerCertificate](#) in *AWS SDK for Ruby API Reference*.

DeleteServiceLinkedRole

The following code example shows how to use DeleteServiceLinkedRole.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

private

# Checks the deletion status of a service-linked role
#
# @param role_name [String] The name of the role being deleted
# @param task_id [String] The task ID for the deletion process
def check_deletion_status(role_name, task_id)
  loop do
    response = @iam_client.get_service_linked_role_deletion_status(
      deletion_task_id: task_id)
    status = response.status
    @logger.info("Deletion of #{role_name} #{status}.")
    break if %w[SUCCEEDED FAILED].include?(status)
    sleep(3)
  end
end

# Handles deletion error
#
# @param e [Aws::Errors::ServiceError] The error encountered during deletion
# @param role_name [String] The name of the role attempted to delete
def handle_deletion_error(e, role_name)
  unless e.code == "NoSuchEntity"
```

```

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

```

- For API details, see [DeleteServiceLinkedRole](#) in *AWS SDK for Ruby API Reference*.

DeleteUser

The following code example shows how to use DeleteUser.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

# 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

```

- For API details, see [DeleteUser](#) in *AWS SDK for Ruby API Reference*.

DeleteUserPolicy

The following code example shows how to use DeleteUserPolicy.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [DeleteUserPolicy](#) in *AWS SDK for Ruby API Reference*.

DetachRolePolicy

The following code example shows how to use DetachRolePolicy.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

This example module lists, creates, attaches, and detaches role policies.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "PolicyManager"
  end

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

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
  end
end
```

```

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

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

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

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

```

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

- For API details, see [DetachRolePolicy](#) in *AWS SDK for Ruby API Reference*.

DetachUserPolicy

The following code example shows how to use `DetachUserPolicy`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
end
```

```

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

```

- For API details, see [DetachUserPolicy](#) in *AWS SDK for Ruby API Reference*.

GetAccountPasswordPolicy

The following code example shows how to use `GetAccountPasswordPolicy`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

# Class to manage IAM account password policies
class PasswordPolicyManager
  attr_accessor :iam_client, :logger

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

  # Retrieves and logs the account password policy
  def print_account_password_policy
    begin
      response = @iam_client.get_account_password_policy
      @logger.info("The account password policy is:
#{response.password_policy.to_h}")
    rescue Aws::IAM::Errors::NoSuchEntity
      @logger.info("The account does not have a password policy.")
    end
  end
end

```

```

    rescue Aws::Errors::ServiceError => e
      @logger.error("Couldn't print the account password policy. Error: #{e.code} -
#{e.message}")
      raise
    end
  end
end
end

```

- For API details, see [GetAccountPasswordPolicy](#) in *AWS SDK for Ruby API Reference*.

GetPolicy

The following code example shows how to use `GetPolicy`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

# 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

```

- For API details, see [GetPolicy](#) in *AWS SDK for Ruby API Reference*.

GetRole

The following code example shows how to use `GetRole`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [GetRole](#) in *AWS SDK for Ruby API Reference*.

GetUser

The following code example shows how to use `GetUser`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [GetUser](#) in *AWS SDK for Ruby API Reference*.

ListAccessKeys

The following code example shows how to use ListAccessKeys.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

This example module lists, creates, deactivates, and deletes access keys.


```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "AccessKeyManager"
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity => e
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

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

- For API details, see [ListAccessKeys](#) in *AWS SDK for Ruby API Reference*.

ListAccountAliases

The following code example shows how to use `ListAccountAliases`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

List, create, and delete account aliases.

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info("Account aliases are:")
      response.account_aliases.each { |account_alias| @logger.info("#{account_alias}") }
    else
      @logger.info("No account aliases found.")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

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

```

    false
  end

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

```

- For API details, see [ListAccountAliases](#) in *AWS SDK for Ruby API Reference*.

ListAttachedRolePolicies

The following code example shows how to use ListAttachedRolePolicies.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

This example module lists, creates, attaches, and detaches role policies.

```

# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "PolicyManager"
  end
end

```

```
end

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

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

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

```

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

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

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

```

- For API details, see [ListAttachedRolePolicies](#) in *AWS SDK for Ruby API Reference*.

ListGroups

The following code example shows how to use ListGroups.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [ListGroups](#) in *AWS SDK for Ruby API Reference*.

ListPolicies

The following code example shows how to use ListPolicies.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

This example module lists, creates, attaches, and detaches role policies.

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = "PolicyManager"
  end

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

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
  end
end
```



```

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

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

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

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

```

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

- For API details, see [ListPolicies](#) in *AWS SDK for Ruby API Reference*.

ListRolePolicies

The following code example shows how to use ListRolePolicies.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [ListRolePolicies](#) in *AWS SDK for Ruby API Reference*.

ListRoles

The following code example shows how to use ListRoles.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [ListRoles](#) in *AWS SDK for Ruby API Reference*.

ListSAMLProviders

The following code example shows how to use ListSAMLProviders.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
class SamlProviderLister
  # Initializes the SamlProviderLister with IAM client and a logger.
  # @param iam_client [Aws::IAM::Client] The IAM client object.
  # @param logger [Logger] The logger object for logging output.
  def initialize(iam_client, logger = Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists up to a specified number of SAML providers for the account.
  # @param count [Integer] The maximum number of providers to list.
  # @return [Aws::IAM::Client::Response]
  def list_saml_providers(count)
    response = @iam_client.list_saml_providers
    response.saml_provider_list.take(count).each do |provider|
      @logger.info("\t#{provider.arn}")
    end
    response
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't list SAML providers. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- For API details, see [ListSAMLProviders](#) in *AWS SDK for Ruby API Reference*.

ListServerCertificates

The following code example shows how to use ListServerCertificates.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

List, update, and delete server certificates.

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

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
      server_certificate_name: name,
      certificate_body: certificate_body,
      private_key: private_key,
    })

    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
  def list_server_certificate_names
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
```

```

    @logger.info("No server certificates found.")
    return
  end

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

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

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

```

- For API details, see [ListServerCertificates](#) in *AWS SDK for Ruby API Reference*.

ListUsers

The following code example shows how to use ListUsers.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [ListUsers](#) in *AWS SDK for Ruby API Reference*.

PutUserPolicy

The following code example shows how to use PutUserPolicy.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [PutUserPolicy](#) in *AWS SDK for Ruby API Reference*.

UpdateServerCertificate

The following code example shows how to use UpdateServerCertificate.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

List, update, and delete server certificates.

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



```
# Creates a new server certificate.
# @param name [String] the name of the server certificate
# @param certificate_body [String] the contents of the certificate
# @param private_key [String] the private key contents
# @return [Boolean] returns true if the certificate was successfully created
def create_server_certificate(name, certificate_body, private_key)
  @iam_client.upload_server_certificate({
    server_certificate_name: name,
    certificate_body: certificate_body,
    private_key: private_key,
  })

  true
rescue Aws::IAM::Errors::ServiceError => e
  puts "Failed to create server certificate: #{e.message}"
  false
end

# Lists available server certificate names.
def list_server_certificate_names
  response = @iam_client.list_server_certificates

  if response.server_certificate_metadata_list.empty?
    @logger.info("No server certificates found.")
    return
  end

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

# Updates the name of a server certificate.
def update_server_certificate_name(current_name, new_name)
  @iam_client.update_server_certificate(
    server_certificate_name: current_name,
    new_server_certificate_name: new_name
  )
  @logger.info("Server certificate name updated from '#{current_name}' to
'#{new_name}'.")
  true
end
```

```

rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error updating server certificate name: #{e.message}")
  false
end

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

```

- For API details, see [UpdateServerCertificate](#) in *AWS SDK for Ruby API Reference*.

UpdateUser

The following code example shows how to use UpdateUser.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

# 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

```

```
end
```

- For API details, see [UpdateUser](#) in *AWS SDK for Ruby API Reference*.

Scenarios

Create a user and assume a role

The following code example shows how to create a user and assume a role.

Warning

To avoid security risks, don't use IAM users for authentication when developing purpose-built software or working with real data. Instead, use federation with an identity provider such as [AWS IAM Identity Center](#).

- Create a user with no permissions.
- Create a role that grants permission to list Amazon S3 buckets for the account.
- Add a policy to let the user assume the role.
- Assume the role and list S3 buckets using temporary credentials, then clean up resources.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Create an IAM user and a role that grants permission to list Amazon S3 buckets. The user has rights only to assume the role. After assuming the role, use temporary credentials to list buckets for the account.

```
# Wraps the scenario actions.
class ScenarioCreateUserAssumeRole
  attr_reader :iam_client
```

```
# @param [Aws::IAM::Client] iam_client: The AWS IAM client.
def initialize(iam_client, logger: Logger.new($stdout))
  @iam_client = iam_client
  @logger = logger
end

# Waits for the specified number of seconds.
#
# @param duration [Integer] The number of seconds to wait.
def wait(duration)
  puts("Give AWS time to propagate resources...")
  sleep(duration)
end

# Creates a user.
#
# @param user_name [String] The name to give the user.
# @return [Aws::IAM::User] The newly created user.
def create_user(user_name)
  user = @iam_client.create_user(user_name: user_name).user
  @logger.info("Created demo user named #{user.user_name}.")
rescue Aws::Errors::ServiceError => e
  @logger.info("Tried and failed to create demo user.")
  @logger.info("\t#{e.code}: #{e.message}")
  @logger.info("\nCan't continue the demo without a user!")
  raise
else
  user
end

# Creates an access key for a user.
#
# @param user [Aws::IAM::User] The user that owns the key.
# @return [Aws::IAM::AccessKeyPair] The newly created access key.
def create_access_key_pair(user)
  user_key = @iam_client.create_access_key(user_name: user.user_name).access_key
  @logger.info("Created accesskey pair for user #{user.user_name}.")
rescue Aws::Errors::ServiceError => e
  @logger.info("Couldn't create access keys for user #{user.user_name}.")
  @logger.info("\t#{e.code}: #{e.message}")
  raise
else
  user_key
end
```

```
end

# Creates a role that can be assumed by a user.
#
# @param role_name [String] The name to give the role.
# @param user [Aws::IAM::User] The user who is granted permission to assume the
role.
# @return [Aws::IAM::Role] The newly created role.
def create_role(role_name, user)
  trust_policy = {
    Version: "2012-10-17",
    Statement: [{
      Effect: "Allow",
      Principal: {'AWS': user.arn},
      Action: "sts:AssumeRole"
    }]
  }.to_json
  role = @iam_client.create_role(
    role_name: role_name,
    assume_role_policy_document: trust_policy
  ).role
  @logger.info("Created role #{role.role_name}.")
rescue Aws::Errors::ServiceError => e
  @logger.info("Couldn't create a role for the demo. Here's why: ")
  @logger.info("\t#{e.code}: #{e.message}")
  raise
else
  role
end

# Creates a policy that grants permission to list S3 buckets in the account, and
# then attaches the policy to a role.
#
# @param policy_name [String] The name to give the policy.
# @param role [Aws::IAM::Role] The role that the policy is attached to.
# @return [Aws::IAM::Policy] The newly created policy.
def create_and_attach_role_policy(policy_name, role)
  policy_document = {
    Version: "2012-10-17",
    Statement: [{
      Effect: "Allow",
      Action: "s3:ListAllMyBuckets",
      Resource: "arn:aws:s3:::*"
    }]
  }
```

```

    }.to_json
    policy = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document
    ).policy
    @iam_client.attach_role_policy(
      role_name: role.role_name,
      policy_arn: policy.arn
    )
    @logger.info("Created policy #{policy.policy_name} and attached it to role
    #{role.role_name}.")
    rescue Aws::Errors::ServiceError => e
      @logger.info("Couldn't create a policy and attach it to role #{role.role_name}.
    Here's why: ")
      @logger.info("\t#{e.code}: #{e.message}")
      raise
    end

    # Creates an inline policy for a user that lets the user assume a role.
    #
    # @param policy_name [String] The name to give the policy.
    # @param user [Aws::IAM::User] The user that owns the policy.
    # @param role [Aws::IAM::Role] The role that can be assumed.
    # @return [Aws::IAM::UserPolicy] The newly created policy.
    def create_user_policy(policy_name, user, role)
      policy_document = {
        Version: "2012-10-17",
        Statement: [{
          Effect: "Allow",
          Action: "sts:AssumeRole",
          Resource: role.arn
        }]
      }.to_json
      @iam_client.put_user_policy(
        user_name: user.user_name,
        policy_name: policy_name,
        policy_document: policy_document
      )
      puts("Created an inline policy for #{user.user_name} that lets the user assume
    role #{role.role_name}.")
      rescue Aws::Errors::ServiceError => e
        @logger.info("Couldn't create an inline policy for user #{user.user_name}.
    Here's why: ")
        @logger.info("\t#{e.code}: #{e.message}")

```

```
    raise
  end

  # Creates an Amazon S3 resource with specified credentials. This is separated into
  # a
  # factory function so that it can be mocked for unit testing.
  #
  # @param credentials [Aws::Credentials] The credentials used by the Amazon S3
  resource.
  def create_s3_resource(credentials)
    Aws::S3::Resource.new(client: Aws::S3::Client.new(credentials: credentials))
  end

  # Lists the S3 buckets for the account, using the specified Amazon S3 resource.
  # Because the resource uses credentials with limited access, it may not be able to
  # list the S3 buckets.
  #
  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def list_buckets(s3_resource)
    count = 10
    s3_resource.buckets.each do |bucket|
      @logger.info "\t#{bucket.name}"
      count -= 1
      break if count.zero?
    end
  rescue Aws::Errors::ServiceError => e
    if e.code == "AccessDenied"
      puts("Attempt to list buckets with no permissions: AccessDenied.")
    else
      @logger.info("Couldn't list buckets for the account. Here's why: ")
      @logger.info("\t#{e.code}: #{e.message}")
      raise
    end
  end
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__
```

- For API details, see the following topics in *AWS SDK for Ruby API Reference*.
 - [AttachRolePolicy](#)
 - [CreateAccessKey](#)
 - [CreatePolicy](#)
 - [CreateRole](#)
 - [CreateUser](#)
 - [DeleteAccessKey](#)
 - [DeletePolicy](#)
 - [DeleteRole](#)
 - [DeleteUser](#)
 - [DeleteUserPolicy](#)
 - [DetachRolePolicy](#)
 - [PutUserPolicy](#)

Kinesis examples using SDK for Ruby

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Ruby with Kinesis.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Serverless examples](#)

Serverless examples

Invoke a Lambda function from a Kinesis trigger

The following code example shows how to implement a Lambda function that receives an event triggered by receiving records from a Kinesis stream. The function retrieves the Kinesis payload, decodes from Base64, and logs the record contents.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Serverless examples](#) repository.

Consuming an Kinesis event with Lambda using 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}"
    end
  end
end
```

```
        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
end
```

Reporting batch item failures for Lambda functions with a Kinesis trigger

The following code example shows how to implement partial batch response for Lambda functions that receive events from a Kinesis stream. The function reports the batch item failures in the response, signaling to Lambda to retry those messages later.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Serverless examples](#) repository.

Reporting Kinesis batch item failures with Lambda using 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'])
    end
  end
end
```

```
    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 examples using SDK for Ruby

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Ruby with AWS KMS.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)

Actions

CreateKey

The following code example shows how to use CreateKey.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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
```

- For API details, see [CreateKey](#) in *AWS SDK for Ruby API Reference*.

Decrypt

The following code example shows how to use Decrypt.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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
```

- For API details, see [Decrypt](#) in *AWS SDK for Ruby API Reference*.

Encrypt

The following code example shows how to use Encrypt.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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*")
```

- For API details, see [Encrypt](#) in *AWS SDK for Ruby API Reference*.

ReEncrypt

The following code example shows how to use ReEncrypt.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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*")
```

- For API details, see [ReEncrypt](#) in *AWS SDK for Ruby API Reference*.

Lambda examples using SDK for Ruby

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Ruby with Lambda.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)

- [Scenarios](#)
- [Serverless examples](#)

Actions

CreateFunction

The following code example shows how to use CreateFunction.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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

```

- For API details, see [CreateFunction](#) in *AWS SDK for Ruby API Reference*.

DeleteFunction

The following code example shows how to use DeleteFunction.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
  end
end

```

```
@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
```

- For API details, see [DeleteFunction](#) in *AWS SDK for Ruby API Reference*.

GetFunction

The following code example shows how to use GetFunction.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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
```

- For API details, see [GetFunction](#) in *AWS SDK for Ruby API Reference*.

Invoke

The following code example shows how to use Invoke.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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
```

- For API details, see [Invoke](#) in *AWS SDK for Ruby API Reference*.

ListFunctions

The following code example shows how to use ListFunctions.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

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

- For API details, see [ListFunctions](#) in *AWS SDK for Ruby API Reference*.

UpdateFunctionCode

The following code example shows how to use UpdateFunctionCode.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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})
  end
end

w = LambdaWrapper.new
w.update_function_code('function_name', 'deployment_package')

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

```

- For API details, see [UpdateFunctionCode](#) in *AWS SDK for Ruby API Reference*.

UpdateFunctionConfiguration

The following code example shows how to use `UpdateFunctionConfiguration`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

class LambdaWrapper
  attr_accessor :lambda_client

  def initialize
    @lambda_client = Aws::Lambda::Client.new
    @logger = Logger.new($stdout)
    @logger.level = Logger::WARN
  end

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

```



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

- For API details, see [UpdateFunctionConfiguration](#) in *AWS SDK for Ruby API Reference*.

Scenarios

Get started with functions

The following code example shows how to:

- Create an IAM role and Lambda function, then upload handler code.
- Invoke the function with a single parameter and get results.
- Update the function code and configure with an environment variable.
- Invoke the function with new parameters and get results. Display the returned execution log.
- List the functions for your account, then clean up resources.

For more information, see [Create a Lambda function with the console](#).

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Set up pre-requisite IAM permissions for a Lambda function capable of writing logs.

```
# Get an AWS Identity and Access Management (IAM) role.
#
# @param iam_role_name: The name of the role to retrieve.
# @param action: Whether to create or destroy the IAM apparatus.
# @return: The IAM role.
def manage_iam(iam_role_name, action)
  role_policy = {
    'Version': "2012-10-17",
    'Statement': [
      {
        'Effect': "Allow",
        'Principal': {
          'Service': "lambda.amazonaws.com"
        },
        'Action': "sts:AssumeRole"
      }
    ]
  }
  case action
  when "create"
    role = $iam_client.create_role(
      role_name: iam_role_name,
      assume_role_policy_document: role_policy.to_json
    )
    $iam_client.attach_role_policy(
      {
        policy_arn: "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole",
        role_name: iam_role_name
      }
    )
    $iam_client.wait_until(:role_exists, { role_name: iam_role_name }) do |w|
      w.max_attempts = 5
      w.delay = 5
    end
    @logger.debug("Successfully created IAM role: #{role['role']['arn']}")
    @logger.debug("Enforcing a 10-second sleep to allow IAM role to activate
fully.")
    sleep(10)
    return role, role_policy.to_json
  when "destroy"
    $iam_client.detach_role_policy(
```

```

    {
      policy_arn: "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole",
      role_name: iam_role_name
    }
  )
  $iam_client.delete_role(
    role_name: iam_role_name
  )
  @logger.debug("Detached policy & deleted IAM role: #{iam_role_name}")
else
  raise "Incorrect action provided. Must provide 'create' or 'destroy'"
end
rescue Aws::Lambda::Errors::ServiceException => e
  @logger.error("There was an error creating role or attaching policy:\n
#{e.message}")
end

```

Define a Lambda handler that increments a number provided as an invocation parameter.

```

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!")
end

```

```

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

```

Zip your Lambda function into a deployment package.

```

# 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

```

Create a new Lambda function.

```

# 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 your Lambda function with optional runtime parameters.

```

# 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

```

Update your Lambda function's configuration to inject a new environment variable.

```
# 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
```

Update your Lambda function's code with a different deployment package containing different code.

```
# 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

```

List all existing Lambda functions using the built-in paginator.

```

# 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

```

Delete a specific Lambda function.

```

# 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
end

```

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

- For API details, see the following topics in *AWS SDK for Ruby API Reference*.
 - [CreateFunction](#)
 - [DeleteFunction](#)
 - [GetFunction](#)
 - [Invoke](#)
 - [ListFunctions](#)
 - [UpdateFunctionCode](#)
 - [UpdateFunctionConfiguration](#)

Serverless examples

Invoke a Lambda function from a Kinesis trigger

The following code example shows how to implement a Lambda function that receives an event triggered by receiving records from a Kinesis stream. The function retrieves the Kinesis payload, decodes from Base64, and logs the record contents.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Serverless examples](#) repository.

Consuming an Kinesis event with Lambda using 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

```

Invoke a Lambda function from a DynamoDB trigger

The following code example shows how to implement a Lambda function that receives an event triggered by receiving records from a DynamoDB stream. The function retrieves the DynamoDB payload and logs the record contents.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Serverless examples](#) repository.

Consuming a DynamoDB event with Lambda using Ruby.

```

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

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
```

Invoke a Lambda function from an Amazon S3 trigger

The following code example shows how to implement a Lambda function that receives an event triggered by uploading an object to an S3 bucket. The function retrieves the S3 bucket name and object key from the event parameter and calls the Amazon S3 API to retrieve and log the content type of the object.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Serverless examples](#) repository.

Consuming an S3 event with Lambda using 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
```

Invoke a Lambda function from an Amazon SNS trigger

The following code example shows how to implement a Lambda function that receives an event triggered by receiving messages from an SNS topic. The function retrieves the messages from the event parameter and logs the content of each message.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Serverless examples](#) repository.

Consuming an SNS event with Lambda using 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
```

Invoke a Lambda function from an Amazon SQS trigger

The following code example shows how to implement a Lambda function that receives an event triggered by receiving messages from an SQS queue. The function retrieves the messages from the event parameter and logs the content of each message.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Serverless examples](#) repository.

Consuming an SQS event with Lambda using 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
```

```
end
end
```

Reporting batch item failures for Lambda functions with a Kinesis trigger

The following code example shows how to implement partial batch response for Lambda functions that receive events from a Kinesis stream. The function reports the batch item failures in the response, signaling to Lambda to retry those messages later.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Serverless examples](#) repository.

Reporting Kinesis batch item failures with Lambda using 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
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
```

Reporting batch item failures for Lambda functions with a DynamoDB trigger

The following code example shows how to implement partial batch response for Lambda functions that receive events from a DynamoDB stream. The function reports the batch item failures in the response, signaling to Lambda to retry those messages later.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Serverless examples](#) repository.

Reporting DynamoDB batch item failures with Lambda using Ruby.

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

```
        return {"batchItemFailures" => [{"itemIdentifier" =>
cur_record_sequence_number}]}
      end
    end

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

Reporting batch item failures for Lambda functions with an Amazon SQS trigger

The following code example shows how to implement partial batch response for Lambda functions that receive events from an SQS queue. The function reports the batch item failures in the response, signaling to Lambda to retry those messages later.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Serverless examples](#) repository.

Reporting SQS batch item failures with Lambda using 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
  end
end
```

```
sqs_batch_response["batchItemFailures"] = batch_item_failures
return sqs_batch_response
end
end
```

Amazon Polly examples using SDK for Ruby

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Ruby with Amazon Polly.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)

Actions

DescribeVoices

The following code example shows how to use DescribeVoices.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).


```
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
```

- For API details, see [DescribeVoices](#) in *AWS SDK for Ruby API Reference*.

ListLexicons

The following code example shows how to use ListLexicons.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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
```

- For API details, see [ListLexicons](#) in *AWS SDK for Ruby API Reference*.

SynthesizeSpeech

The following code example shows how to use SynthesizeSpeech.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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
end
```

```
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
```

- For API details, see [SynthesizeSpeech](#) in *AWS SDK for Ruby API Reference*.

Amazon RDS examples using SDK for Ruby

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Ruby with Amazon RDS.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)

Actions

CreateDBSnapshot

The following code example shows how to use CreateDBSnapshot.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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

```

- For API details, see [CreateDBSnapshot](#) in *AWS SDK for Ruby API Reference*.

DescribeDBInstances

The following code example shows how to use DescribeDBInstances.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

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

```

- For API details, see [DescribeDBInstances](#) in *AWS SDK for Ruby API Reference*.

DescribeDBParameterGroups

The following code example shows how to use `DescribeDBParameterGroups`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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
```

- For API details, see [DescribeDBParameterGroups](#) in *AWS SDK for Ruby API Reference*.

DescribeDBParameters

The following code example shows how to use `DescribeDBParameters`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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
```

- For API details, see [DescribeDBParameters](#) in *AWS SDK for Ruby API Reference*.

DescribeDBSnapshots

The following code example shows how to use DescribeDBSnapshots.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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
```

- For API details, see [DescribeDBSnapshots](#) in *AWS SDK for Ruby API Reference*.

Amazon S3 examples using SDK for Ruby

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Ruby with Amazon S3.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)

- [Scenarios](#)
- [Serverless examples](#)

Actions

CopyObject

The following code example shows how to use CopyObject.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Copy an object.

```
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__

```

Copy an object and add server-side encryption to the destination object.

```

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__

```

- For API details, see [CopyObject](#) in *AWS SDK for Ruby API Reference*.

CreateBucket

The following code example shows how to use CreateBucket.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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__

```

- For API details, see [CreateBucket](#) in *AWS SDK for Ruby API Reference*.

DeleteBucket

The following code example shows how to use DeleteBucket.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

# 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

```

- For API details, see [DeleteBucket](#) in *AWS SDK for Ruby API Reference*.

DeleteBucketCors

The following code example shows how to use DeleteBucketCors.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

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
```

- For API details, see [DeleteBucketCors](#) in *AWS SDK for Ruby API Reference*.

DeleteBucketPolicy

The following code example shows how to use DeleteBucketPolicy.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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

```

- For API details, see [DeleteBucketPolicy](#) in *AWS SDK for Ruby API Reference*.

DeleteObjects

The following code example shows how to use DeleteObjects.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

# 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

```

- For API details, see [DeleteObjects](#) in *AWS SDK for Ruby API Reference*.

GetBucketCors

The following code example shows how to use GetBucketCors.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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
```

- For API details, see [GetBucketCors](#) in *AWS SDK for Ruby API Reference*.

GetBucketPolicy

The following code example shows how to use `GetBucketPolicy`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [GetBucketPolicy](#) in *AWS SDK for Ruby API Reference*.

GetObject

The following code example shows how to use `GetObject`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Get an object.

```
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__

```

Get an object and report its server-side encryption state.

```

require "aws-sdk-s3"

# Wraps Amazon S3 object actions.
class ObjectGetEncryptionWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object into memory.
  #
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
  # successful; otherwise nil.
  def get_object
    @object.get
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

# Example usage:
def run_demo
  bucket_name = "doc-example-bucket"
  object_key = "my-object.txt"

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

  encryption = obj_data.server_side_encryption.nil? ? "no" :
  obj_data.server_side_encryption

```

```

    puts "Object #{object_key} uses #{encryption} encryption."
  end

  run_demo if $PROGRAM_NAME == __FILE__

```

- For API details, see [GetObject](#) in *AWS SDK for Ruby API Reference*.

HeadObject

The following code example shows how to use HeadObject.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

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__
```

- For API details, see [HeadObject](#) in *AWS SDK for Ruby API Reference*.

ListBuckets

The following code example shows how to use ListBuckets.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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__
```

- For API details, see [ListBuckets](#) in *AWS SDK for Ruby API Reference*.

ListObjectsV2

The following code example shows how to use ListObjectsV2.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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__

```

- For API details, see [ListObjectsV2](#) in *AWS SDK for Ruby API Reference*.

PutBucketCors

The following code example shows how to use PutBucketCors.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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
```

- For API details, see [PutBucketCors](#) in *AWS SDK for Ruby API Reference*.

PutBucketPolicy

The following code example shows how to use PutBucketPolicy.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [PutBucketPolicy](#) in *AWS SDK for Ruby API Reference*.

PutBucketWebsite

The following code example shows how to use PutBucketWebsite.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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__
```

- For API details, see [PutBucketWebsite](#) in *AWS SDK for Ruby API Reference*.

PutObject

The following code example shows how to use PutObject.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Upload a file using a managed uploader (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__

```

Upload a file using 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__

```

Upload a file using `Object.put` and add server-side encryption.

```

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__
```

- For API details, see [PutObject](#) in *AWS SDK for Ruby API Reference*.

Scenarios

Create a presigned URL

The following code example shows how to create a presigned URL for Amazon S3 and upload an object.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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__
```

Get started with buckets and objects

The following code example shows how to:

- Create a bucket and upload a file to it.
- Download an object from a bucket.
- Copy an object to a subfolder in a bucket.
- List the objects in a bucket.
- Delete the bucket objects and the bucket.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-s3"

# Wraps the getting started scenario actions.
class ScenarioGettingStarted
  attr_reader :s3_resource

  # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
  def initialize(s3_resource)
    @s3_resource = s3_resource
  end

  # Creates a bucket with a random name in the currently configured account and
```

```
# AWS Region.
#
# @return [Aws::S3::Bucket] The newly created bucket.
def create_bucket
  bucket = @s3_resource.create_bucket(
    bucket: "doc-example-bucket-#{Random.uuid}",
    create_bucket_configuration: {
      location_constraint: "us-east-1" # Note: only certain regions permitted
    }
  )
  puts("Created demo bucket named #{bucket.name}.")
rescue Aws::Errors::ServiceError => e
  puts("Tried and failed to create demo bucket.")
  puts("\t#{e.code}: #{e.message}")
  puts("\nCan't continue the demo without a bucket!")
  raise
else
  bucket
end

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

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

```
end

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

# Copies an Amazon S3 object to a subfolder within the same bucket.
#
# @param source_object [Aws::S3::Object] The source object to copy.
# @return [Aws::S3::Object, nil] The destination object.
def copy_object(source_object)
  dest_object = nil
  puts("\nDo you want to copy #{source_object.key} to a subfolder in your bucket
(y/n)? ")
  answer = gets.chomp.downcase
  if answer == "y"
    dest_object = source_object.bucket.object("demo-folder/#{source_object.key}")
    dest_object.copy_from(source_object)
    puts("Copied #{source_object.key} to #{dest_object.key}.")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't copy #{source_object.key}.")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  dest_object
end

# Lists the objects in an Amazon S3 bucket.
#
```

```
# @param bucket [Aws::S3::Bucket] The bucket to query.
def list_objects(bucket)
  puts("\nYour bucket contains the following objects:")
  bucket.objects.each do |obj|
    puts("\t#{obj.key}")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't list the objects in bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end

# Deletes the objects in an Amazon S3 bucket and deletes the bucket.
#
# @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
def delete_bucket(bucket)
  puts("\nDo you want to delete all of the objects as well as the bucket (y/n)? ")
  answer = gets.chomp.downcase
  if answer == "y"
    bucket.objects.batch_delete!
    bucket.delete
    puts("Emptied and deleted bucket #{bucket.name}.\n")
  end
rescue Aws::Errors::ServiceError => e
  puts("Couldn't empty and delete bucket #{bucket.name}.")
  puts("\t#{e.code}: #{e.message}")
  raise
end

# 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!")
end
```

```
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__
```

- For API details, see the following topics in *AWS SDK for Ruby API Reference*.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Serverless examples

Invoke a Lambda function from an Amazon S3 trigger

The following code example shows how to implement a Lambda function that receives an event triggered by uploading an object to an S3 bucket. The function retrieves the S3 bucket name and object key from the event parameter and calls the Amazon S3 API to retrieve and log the content type of the object.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Serverless examples](#) repository.

Consuming an S3 event with Lambda using 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
```

Amazon SES examples using SDK for Ruby

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Ruby with Amazon SES.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics


- [Actions](#)

Actions

GetIdentityVerificationAttributes

The following code example shows how to use `GetIdentityVerificationAttributes`.

SDK for Ruby

 **Note**

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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
```

- For API details, see [GetIdentityVerificationAttributes](#) in *AWS SDK for Ruby API Reference*.

ListIdentities

The following code example shows how to use `ListIdentities`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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
```

- For API details, see [ListIdentities](#) in *AWS SDK for Ruby API Reference*.

SendEmail

The following code example shows how to use SendEmail.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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
```

- For API details, see [SendEmail](#) in *AWS SDK for Ruby API Reference*.

VerifyEmailIdentity

The following code example shows how to use `VerifyEmailIdentity`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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
```

- For API details, see [VerifyEmailIdentity](#) in *AWS SDK for Ruby API Reference*.

Amazon SES API v2 examples using SDK for Ruby

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Ruby with Amazon SES API v2.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)

Actions

SendEmail

The following code example shows how to use `SendEmail`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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)
```

- For API details, see [SendEmail](#) in *AWS SDK for Ruby API Reference*.

Amazon SNS examples using SDK for Ruby

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Ruby with Amazon SNS.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)
- [Serverless examples](#)

Actions

CreateTopic

The following code example shows how to use CreateTopic.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

```
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_name = "YourTopicName" # Replace with your topic name
  sns_topic_creator = SNSTopicCreator.new

  puts "Creating the topic '#{topic_name}'..."
  unless sns_topic_creator.create_topic(topic_name)
    puts "The topic was not created. Stopping program."
    exit 1
  end
end
```

- For more information, see [AWS SDK for Ruby Developer Guide](#).
- For API details, see [CreateTopic](#) in *AWS SDK for Ruby API Reference*.

ListSubscriptions

The following code example shows how to use ListSubscriptions.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# This class demonstrates how to list subscriptions to an Amazon Simple Notification
Service (SNS) topic
class SnsSubscriptionLister
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Lists subscriptions for a given SNS topic
  # @param topic_arn [String] The ARN of the SNS topic
  # @return [Types::ListSubscriptionsResponse] subscriptions: The response object
```

```
def list_subscriptions(topic_arn)
  @logger.info("Listing subscriptions for topic: #{topic_arn}")
  subscriptions = @sns_client.list_subscriptions_by_topic(topic_arn: topic_arn)
  subscriptions.subscriptions.each do |subscription|
    @logger.info("Subscription endpoint: #{subscription.endpoint}")
  end
  subscriptions
rescue Aws::SNS::Errors::ServiceError => e
  @logger.error("Error listing subscriptions: #{e.message}")
  raise
end

# 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
```

- For more information, see [AWS SDK for Ruby Developer Guide](#).
- For API details, see [ListSubscriptions](#) in *AWS SDK for Ruby API Reference*.

ListTopics

The following code example shows how to use ListTopics.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).


```
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__
```

- For more information, see [AWS SDK for Ruby Developer Guide](#).
- For API details, see [ListTopics](#) in *AWS SDK for Ruby API Reference*.

Publish

The following code example shows how to use Publish.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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.")
end
```

```
unless message_sender.send_message(topic_arn, message)
  @logger.error("Message sending failed. Stopping program.")
  exit 1
end
end
```

- For more information, see [AWS SDK for Ruby Developer Guide](#).
- For API details, see [Publish](#) in *AWS SDK for Ruby API Reference*.

SetTopicAttributes

The following code example shows how to use SetTopicAttributes.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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)
  end
end
```

```

    topic.set_attributes({
      attribute_name: policy_name,
      attribute_value: policy
    })
    @logger.info("Policy #{policy_name} set successfully for topic #{topic_arn}.")
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Failed to set policy: #{e.message}")
  end

private

# Generates a policy string with dynamic resource ARNs
#
# @param topic_arn [String] The ARN of the SNS topic
# @param resource_arn [String] The ARN of the resource
# @return [String] The policy as a JSON string
def generate_policy(topic_arn, resource_arn)
  {
    Version: "2008-10-17",
    Id: "__default_policy_ID",
    Statement: [{
      Sid: "__default_statement_ID",
      Effect: "Allow",
      Principal: { "AWS": "*" },
      Action: ["SNS:Publish"],
      Resource: topic_arn,
      Condition: {
        ArnEquals: {
          "AWS:SourceArn": resource_arn
        }
      }
    }]
  }.to_json
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = "MY_TOPIC_ARN" # Should be replaced with a real topic ARN
  resource_arn = "MY_RESOURCE_ARN" # Should be replaced with a real resource ARN
  policy_name = "POLICY_NAME" # Typically, this is "Policy"

  sns_resource = Aws::SNS::Resource.new
  enabler = SnsResourceEnabler.new(sns_resource)
end

```

```
    enabler.enable_resource(topic_arn, resource_arn, policy_name)
end
```

- For more information, see [AWS SDK for Ruby Developer Guide](#).
- For API details, see [SetTopicAttributes](#) in *AWS SDK for Ruby API Reference*.

Subscribe

The following code example shows how to use Subscribe.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

Subscribe an email address to a topic.

```
require "aws-sdk-sns"
require "logger"

# Represents a service for creating subscriptions in Amazon Simple Notification
# Service (SNS)
class SubscriptionService
  # Initializes the SubscriptionService with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Attempts to create a subscription to a topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param protocol [String] The subscription protocol (e.g., email)
  # @param endpoint [String] The endpoint that receives the notifications (email
  # address)
```

```
# @return [Boolean] true if subscription was successfully created, false otherwise
def create_subscription(topic_arn, protocol, endpoint)
  @sns_client.subscribe(topic_arn: topic_arn, protocol: protocol, endpoint:
endpoint)
  @logger.info("Subscription created successfully.")
  true
rescue Aws::SNS::Errors::ServiceError => e
  @logger.error("Error while creating the subscription: #{e.message}")
  false
end
end

# Main execution if the script is run directly
if $PROGRAM_NAME == __FILE__
  protocol = "email"
  endpoint = "EMAIL_ADDRESS" # Should be replaced with a real email address
  topic_arn = "TOPIC_ARN"    # Should be replaced with a real topic ARN

  sns_client = Aws::SNS::Client.new
  subscription_service = SubscriptionService.new(sns_client)

  @logger.info("Creating the subscription.")
  unless subscription_service.create_subscription(topic_arn, protocol, endpoint)
    @logger.error("Subscription creation failed. Stopping program.")
    exit 1
  end
end
end
```

- For more information, see [AWS SDK for Ruby Developer Guide](#).
- For API details, see [Subscribe](#) in *AWS SDK for Ruby API Reference*.

Serverless examples

Invoke a Lambda function from an Amazon SNS trigger

The following code example shows how to implement a Lambda function that receives an event triggered by receiving messages from an SNS topic. The function retrieves the messages from the event parameter and logs the content of each message.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Serverless examples](#) repository.

Consuming an SNS event with Lambda using 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
```

Amazon SQS examples using SDK for Ruby

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Ruby with Amazon SQS.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)
- [Serverless examples](#)

Actions

ChangeMessageVisibility

The following code example shows how to use ChangeMessageVisibility.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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
end
```



```

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

```

- For API details, see [ChangeMessageVisibility](#) in *AWS SDK for Ruby API Reference*.

CreateQueue

The following code example shows how to use CreateQueue.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```

# 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__
```

- For API details, see [CreateQueue](#) in *AWS SDK for Ruby API Reference*.

DeleteQueue

The following code example shows how to use DeleteQueue.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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)
```

- For API details, see [DeleteQueue](#) in *AWS SDK for Ruby API Reference*.

ListQueues

The following code example shows how to use `ListQueues`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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
```

- For API details, see [ListQueues](#) in *AWS SDK for Ruby API Reference*.

ReceiveMessage

The following code example shows how to use `ReceiveMessage`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
require "aws-sdk-sqs"
require "aws-sdk-sts"

# Receives messages in a queue in Amazon Simple Queue Service (Amazon SQS).
#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param max_number_of_messages [Integer] The maximum number of messages
#   to receive. This number must be 10 or less. The default is 10.
# @example
#   receive_messages(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue',
#     10
#   )
def receive_messages(sqs_client, queue_url, max_number_of_messages = 10)

  if max_number_of_messages > 10
    puts "Maximum number of messages to receive must be 10 or less. " \
      "Stopping program."
    return
  end

  response = sqs_client.receive_message(
    queue_url: queue_url,
    max_number_of_messages: max_number_of_messages
  )
```

```
if response.messages.count.zero?
  puts "No messages to receive, or all messages have already " \
    "been previously received."
  return
end

response.messages.each do |message|
  puts "-" * 20
  puts "Message body: #{message.body}"
  puts "Message ID:  #{message.message_id}"
end

rescue StandardError => e
  puts "Error receiving messages: #{e.message}"
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = "us-west-2"
  queue_name = "my-queue"
  max_number_of_messages = 10

  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
  queue_url = "https://sqs." + region + ".amazonaws.com/" +
    sts_client.get_caller_identity.account + "/" + queue_name

  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Receiving messages from queue '#{queue_name}'..."

  receive_messages(sqs_client, queue_url, max_number_of_messages)
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- For API details, see [ReceiveMessage](#) in *AWS SDK for Ruby API Reference*.

SendMessage

The following code example shows how to use SendMessage.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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__
```

- For API details, see [SendMessage](#) in *AWS SDK for Ruby API Reference*.

SendMessageBatch

The following code example shows how to use `SendMessageBatch`.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
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."
    }
  ]
end
```

```
    }
  ]

  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
```

- For API details, see [SendMessageBatch](#) in *AWS SDK for Ruby API Reference*.

Serverless examples

Invoke a Lambda function from an Amazon SQS trigger

The following code example shows how to implement a Lambda function that receives an event triggered by receiving messages from an SQS queue. The function retrieves the messages from the event parameter and logs the content of each message.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Serverless examples](#) repository.

Consuming an SQS event with Lambda using 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
```

Reporting batch item failures for Lambda functions with an Amazon SQS trigger

The following code example shows how to implement partial batch response for Lambda functions that receive events from an SQS queue. The function reports the batch item failures in the response, signaling to Lambda to retry those messages later.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [Serverless examples](#) repository.

Reporting SQS batch item failures with Lambda using 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 examples using SDK for Ruby

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Ruby with AWS STS.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)

Actions

AssumeRole

The following code example shows how to use AssumeRole.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [AssumeRole](#) in *AWS SDK for Ruby API Reference*.

Amazon WorkDocs examples using SDK for Ruby

The following code examples show you how to perform actions and implement common scenarios by using the AWS SDK for Ruby with Amazon WorkDocs.

Actions are code excerpts from larger programs and must be run in context. While actions show you how to call individual service functions, you can see actions in context in their related scenarios and cross-service examples.

Scenarios are code examples that show you how to accomplish a specific task by calling multiple functions within the same service.

Each example includes a link to GitHub, where you can find instructions on how to set up and run the code in context.

Topics

- [Actions](#)

Actions

DescribeRootFolders

The following code example shows how to use DescribeRootFolders.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
  end
end
```

```

    @logger.error "Could not get root folder for user with email address
#{user_email}"
    exit(1)
  end
end

# Describes the contents of a folder
# @param [String] folder_id - The Id of the folder to describe.
def describe_folder_contents(folder_id)
  resp = @client.describe_folder_contents({
    folder_id: folder_id, # required
    sort: "NAME", # accepts DATE, NAME
    order: "ASCENDING", # accepts
ASCENDING, DESCENDING
  })

  resp.documents.each do |doc|
    md = doc.latest_version_metadata
    @logger.info "Name:          #{md.name}"
    @logger.info "Size (bytes):  #{md.size}"
    @logger.info "Last modified: #{doc.modified_timestamp}"
    @logger.info "Doc ID:        #{doc.id}"
    @logger.info "Version ID:    #{md.id}"
    @logger.info ""
  end
rescue Aws::WorkDocs::Errors::ServiceError => e
  @logger.error "Error listing folder contents: #{e.message}"
  exit(1)
end

```

- For API details, see [DescribeRootFolders](#) in *AWS SDK for Ruby API Reference*.

DescribeUsers

The following code example shows how to use DescribeUsers.

SDK for Ruby

Note

There's more on GitHub. Find the complete example and learn how to set up and run in the [AWS Code Examples Repository](#).

```
# 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
```

- For API details, see [DescribeUsers](#) in *AWS SDK for Ruby API Reference*.

Cross-service examples using SDK for Ruby

The following sample applications use the AWS SDK for Ruby to work across multiple AWS services.

Cross-service examples target an advanced level of experience to help you start building applications.

Examples

- [Create an application that analyzes customer feedback and synthesizes audio](#)

Create an application that analyzes customer feedback and synthesizes audio

SDK for Ruby

This example application analyzes and stores customer feedback cards. Specifically, it fulfills the need of a fictitious hotel in New York City. The hotel receives feedback from guests in various languages in the form of physical comment cards. That feedback is uploaded into the app through a web client. After an image of a comment card is uploaded, the following steps occur:

- Text is extracted from the image using Amazon Textract.
- Amazon Comprehend determines the sentiment of the extracted text and its language.
- The extracted text is translated to English using Amazon Translate.
- Amazon Polly synthesizes an audio file from the extracted text.

The full app can be deployed with the AWS CDK. For source code and deployment instructions, see the project in [GitHub](#).

Services used in this example

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

Security for AWS SDK for Ruby

Cloud security at Amazon Web Services (AWS) is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations. Security is a shared responsibility between AWS and you. The [Shared Responsibility Model](#) describes this as Security of the Cloud and Security in the Cloud.

Security of the Cloud– AWS is responsible for protecting the infrastructure that runs all of the services offered in the AWS Cloud and providing you with services that you can use securely. Our security responsibility is the highest priority at AWS, and the effectiveness of our security is regularly tested and verified by third-party auditors as part of the [AWS Compliance Programs](#).

Security in the Cloud– Your responsibility is determined by the AWS service you are using, and other factors including the sensitivity of your data, your organization’s requirements, and applicable laws and regulations.

Topics

- [Data Protection in AWS SDK for Ruby](#)
- [Identity and Access Management for AWS SDK for Ruby](#)
- [Compliance Validation for AWS SDK for Ruby](#)
- [Resilience for AWS SDK for Ruby](#)
- [Infrastructure Security for AWS SDK for Ruby](#)
- [Enforcing a minimum TLS version in the AWS SDK for Ruby](#)
- [Amazon S3 Encryption Client Migration](#)

Data Protection in AWS SDK for Ruby

The AWS [shared responsibility model](#) applies to data protection in . As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. You are also responsible for the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the [Data Privacy FAQ](#). For information about data protection in Europe, see the [AWS Shared Responsibility Model and GDPR](#) blog post on the *AWS Security Blog*.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual users with AWS IAM Identity Center or AWS Identity and Access Management (IAM). That way, each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use SSL/TLS to communicate with AWS resources. We require TLS 1.2 and recommend TLS 1.3.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing sensitive data that is stored in Amazon S3.
- If you require FIPS 140-2 validated cryptographic modules when accessing AWS through a command line interface or an API, use a FIPS endpoint. For more information about the available FIPS endpoints, see [Federal Information Processing Standard \(FIPS\) 140-2](#).

We strongly recommend that you never put confidential or sensitive information, such as your customers' email addresses, into tags or free-form text fields such as a **Name** field. This includes when you work with or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into tags or free-form text fields used for names may be used for billing or diagnostic logs. If you provide a URL to an external server, we strongly recommend that you do not include credentials information in the URL to validate your request to that server.

Identity and Access Management for AWS SDK for Ruby

AWS Identity and Access Management (IAM) is an Amazon Web Services (AWS) service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use resources AWS services. IAM is an AWS service that you can use with no additional charge.

To use AWS SDK for Ruby to access AWS, you need an AWS account and AWS credentials. To increase the security of your AWS account, we recommend that you use an *IAM user* to provide access credentials instead of using your AWS account credentials.

For details about working with IAM, see [IAM](#).

For an overview of IAM users and why they are important for the security of your account, see [AWS Security Credentials](#) in the [Amazon Web Services General Reference](#).

AWS SDK for Ruby follows the [shared responsibility model](#) through the specific Amazon Web Services (AWS) services it supports. For AWS service security information, see the [AWS service security documentation page](#) and [AWS services that are in scope of AWS compliance efforts by compliance program](#).

Compliance Validation for AWS SDK for Ruby

AWS SDK for Ruby follows the [shared responsibility model](#) through the specific Amazon Web Services (AWS) services it supports. For AWS service security information, see the [AWS service security documentation page](#) and [AWS services that are in scope of AWS compliance efforts by compliance program](#).

The security and compliance of Amazon Web Services (AWS) services is assessed by third-party auditors as part of multiple AWS compliance programs. These include SOC, PCI, FedRAMP, HIPAA, and others. AWS provides a frequently updated list of AWS services in scope of specific compliance programs at [AWS Services in Scope by Compliance Program](#).

Third-party audit reports are available for you to download using AWS Artifact. For more information, see [Downloading Reports in AWS Artifact](#).

For more information about AWS compliance programs, see [AWS Compliance Programs](#).

Your compliance responsibility when using AWS SDK for Ruby to access an AWS service is determined by the sensitivity of your data, your organization's compliance objectives, and applicable laws and regulations. If your use of an AWS service is subject to compliance with standards such as HIPAA, PCI, or FedRAMP, AWS provides resources to help:

- [Security and Compliance Quick Start Guides](#)– Deployment guides that discuss architectural considerations and provide steps for deploying security-focused and compliance-focused baseline environments on AWS.
- [Architecting for HIPAA Security and Compliance Whitepaper](#)– A whitepaper that describes how companies can use AWS to create HIPAA-compliant applications.
- [AWS Compliance Resources](#)– A collection of workbooks and guides that might apply to your industry and location.
- [AWS Config](#)– A service that assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- [AWS Security Hub](#)– A comprehensive view of your security state within AWS that helps you check your compliance with security industry standards and best practices.

Resilience for AWS SDK for Ruby

The Amazon Web Services (AWS) global infrastructure is built around AWS Regions and Availability Zones.

AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking.

With Availability Zones, you can design and operate applications and databases that automatically fail over between Availability Zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see [AWS Global Infrastructure](#).

AWS SDK for Ruby follows the [shared responsibility model](#) through the specific Amazon Web Services (AWS) services it supports. For AWS service security information, see the [AWS service security documentation page](#) and [AWS services that are in scope of AWS compliance efforts by compliance program](#).

Infrastructure Security for AWS SDK for Ruby

AWS SDK for Ruby follows the [shared responsibility model](#) through the specific Amazon Web Services (AWS) services it supports. For AWS service security information, see the [AWS service security documentation page](#) and [AWS services that are in scope of AWS compliance efforts by compliance program](#).

For information about AWS security processes, see the [AWS: Overview of Security Processes](#) whitepaper.

Enforcing a minimum TLS version in the AWS SDK for Ruby

Communication between the AWS SDK for Ruby and AWS is secured using Secure Sockets Layer (SSL) or Transport Layer Security (TLS). All versions of SSL, and versions of TLS earlier than 1.2, have vulnerabilities that can compromise the security of your communication with AWS. For this reason, you should make sure that you're using the AWS SDK for Ruby with a version of Ruby that supports TLS version 1.2 or later.

Ruby uses the OpenSSL library to secure HTTP connections. Supported versions of Ruby (1.9.3 and later) installed through system [package managers](#) (yum, apt, and others), an [official installer](#),

or Ruby [managers](#) (rbenv, RVM, and others) typically incorporate OpenSSL 1.0.1 or later, which supports TLS 1.2.

When used with a supported version of Ruby with OpenSSL 1.0.1 or later, the AWS SDK for Ruby prefers TLS 1.2, and uses the latest version of SSL or TLS supported by both the client and server. This is always at least TLS 1.2 for AWS services. (The SDK uses the Ruby Net `::HTTP` class with `use_ssl=true`.)

Checking the OpenSSL version

To make sure your installation of Ruby is using OpenSSL 1.0.1 or later, enter the following command.

```
ruby -r openssl -e 'puts OpenSSL::OPENSSL_VERSION'
```

An alternative way to get the OpenSSL version is to query the `openssl` executable directly. First, locate the appropriate executable using the following command.

```
ruby -r rbconfig -e 'puts RbConfig::CONFIG["configure_args"]'
```

The output should have `--with-openssl-dir=/path/to/openssl` indicating the location of the OpenSSL installation. Make a note of this path. To check the version of OpenSSL, enter the following commands.

```
cd /path/to/openssl  
bin/openssl version
```

This latter method might not work with all installations of Ruby.

Upgrading TLS support

If the version of OpenSSL used by your Ruby installation is earlier than 1.0.1, upgrade your Ruby or OpenSSL installation using your system package manager, Ruby installer, or Ruby manager, as described in Ruby's [installation guide](#). If you're installing Ruby [from source](#), install the [latest OpenSSL](#) first, and then pass `--with-openssl-dir=/path/to/upgraded/openssl` when running `./configure`.

Amazon S3 Encryption Client Migration

This topic shows how to migrate your applications from Version 1 (V1) of the Amazon Simple Storage Service (Amazon S3) encryption client to Version 2 (V2), and ensure application availability throughout the migration process.

Migration Overview

This migration happens in two phases:

- 1. Update existing clients to read new formats.** First, deploy an updated version of the AWS SDK for Ruby to your application. This will allow existing V1 encryption clients to decrypt objects written by the new V2 clients. If your application uses multiple AWS SDKs, you must upgrade each SDK separately.
- 2. Migrate encryption and decryption clients to V2.** Once all of your V1 encryption clients can read new formats, you can migrate your existing encryption and decryption clients to their respective V2 versions.

Update Existing Clients to Read New Formats

The V2 encryption client uses encryption algorithms that older versions of the client don't support. The first step in the migration is to update your V1 decryption clients to the latest SDK release. After completing this step, your application's V1 clients will be able to decrypt objects encrypted by V2 encryption clients. See details below for each major version of the AWS SDK for Ruby.

Update AWS SDK for Ruby Version 3

Version 3 is the latest version of the AWS SDK For Ruby. To complete this migration, you need to use version 1.76.0 or later of the `aws-sdk-s3` gem.

Installing from the Command Line

For projects that install the `aws-sdk-s3` gem, use the `version` option to verify that the minimum version of 1.76.0 is installed.

```
gem install aws-sdk-s3 -v '>= 1.76.0'
```

Using Gemfiles

For projects that use a Gemfile to manage dependencies, set the minimum version of the `aws-sdk-s3` gem to 1.76.0. For example:

```
gem 'aws-sdk-s3', '>= 1.76.0'
```

1. Modify your Gemfile.
2. Run `bundle update aws-sdk-s3`. To verify your version, run `bundle info aws-sdk-s3`.

Update AWS SDK for Ruby Version 2

Version 2 of the AWS SDK for Ruby will enter [maintenance mode](#) on November 21st, 2021. To complete this migration, you need to use version 2.11.562 or later of the `aws-sdk` gem.

Installing from the Command Line

For projects that install the `aws-sdk` gem, from the command line, use the version option to verify that the minimum version of 2.11.562 is installed.

```
gem install aws-sdk -v '>= 2.11.562'
```

Using Gemfiles

For projects that use a Gemfile to manage dependencies, set the minimum version of the `aws-sdk` gem to 2.11.562. For example:

```
gem 'aws-sdk', '>= 2.11.562'
```

1. Modify your Gemfile. If you have a Gemfile.lock file, delete or update it.
2. Run `bundle update aws-sdk`. To verify your version, run `bundle info aws-sdk`.

Migrate Encryption and Decryption Clients to V2

After updating your clients to read the new encryption formats, you can update your applications to the V2 encryption and decryption clients. The following steps show you how to successfully migrate your code from V1 to V2.

Before updating your code to use the V2 encryption client, ensure that you have followed the preceding steps and are using the `aws-sdk-s3` gem version 2.11.562 or later.

Note

When decrypting with AES-GCM, read the entire object to the end before you start using the decrypted data. This is to verify that the object has not been modified since it was encrypted.

Configuring V2 Encryption Clients

The `EncryptionV2::Client` requires additional configuration. For detailed configuration information, see the [EncryptionV2::Client documentation](#) or the examples provided later in this topic.

1. The key wrap method and content encryption algorithm must be specified on client construction. When creating a new `EncryptionV2::Client`, you need to provide values for `key_wrap_schema` and `content_encryption_schema`.

`key_wrap_schema` - If you are using AWS KMS, this must be set to `:kms_context`. If you are using a symmetric (AES) key, it must be set to `:aes_gcm`. If you are using an asymmetric (RSA) key, it must be set to `:rsa_oaep_sha1`.

`content_encryption_schema` - This must be set to `:aes_gcm_no_padding`.

2. security_profile must be specified on client construction. When creating a new `EncryptionV2::Client`, you need to provide a value for `security_profile`. The `security_profile` parameter determines the support for reading objects written using the older V1 `Encryption::Client`. There are two values: `:v2` and `:v2_and_legacy`. To support migration, set the `security_profile` to `:v2_and_legacy`. Use `:v2` only for new application development.

3. AWS KMS key ID is enforced by default. In V1, `Encryption::Client`, the `kms_key_id` used to create the client was not provided to the AWS KMS `Decrypt` call. AWS KMS can get this information from metadata and add it to the symmetric ciphertext blob. In V2, `EncryptionV2::Client`, the `kms_key_id` is passed to the AWS KMS `Decrypt` call, and the call fails if it does not match the key used to encrypt the object. If your code previously relied on not setting a specific `kms_key_id`, either set `kms_key_id: :kms_allow_decrypt_with_any_cmek` on client creation or set `kms_allow_decrypt_with_any_cmek: true` on `get_object` calls.

Example: Using a Symmetric (AES) Key

Pre-migration

```
client = Aws::S3::Encryption::Client.new(encryption_key: aes_key)
client.put_object(bucket: bucket, key: key, body: secret_data)
resp = client.get_object(bucket: bucket, key: key)
```

Post-migration

```
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
```

Example: Using AWS KMS with kms_key_id

Pre-migration

```
client = Aws::S3::Encryption::Client.new(kms_key_id: kms_key_id)
client.put_object(bucket: bucket, key: key, body: secret_data)
resp = client.get_object(bucket: bucket, key: key)
```

Post-migration

```
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
```

Example: Using AWS KMS without kms_key_id

Pre-migration

```
client = Aws::S3::Encryption::Client.new(kms_key_id: kms_key_id)
client.put_object(bucket: bucket, key: key, body: secret_data)
resp = client.get_object(bucket: bucket, key: key)
```

Post-migration

```
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
```

Post-Migration Alternative

If you only read and decrypt (never write and encrypt) objects using the S2 encryption client, use this code.

```
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
```

Document History

The following table describes important changes in this guide. For notification about updates to this documentation, you can subscribe to an [RSS feed](#).

Change	Description	Date
Table of contents	Updated table of contents to make code examples more accessible.	June 1, 2023
IAM best practices updates	Updated guide to align with the IAM best practices . For more information, see Security best practices in IAM . Updates to Getting started.	May 8, 2023
General updates	Updating welcome page for relevant external resources . Also updated minimum required Ruby version for v2.3. Updated AWS Key Management Service sections to reflect terminology updates. Updated usage information on REPL utility for clarity.	August 8, 2022
Correcting broken links	Fixed broken examples links. Removed redundant Tips and Tricks page; redirecting to Amazon EC2 example content. Included lists of the code examples that are available on GitHub in the Code Examples repository.	August 3, 2022

[Getting Information about All Amazon RDS Security Groups](#)

Added note about retiring EC2-Classic.

July 26, 2022

[SDK Metrics](#)

Removed information about enabling SDK Metrics for Enterprise Support, which has been deprecated.

January 28, 2022