개발자 가이드

AWS SDK for Ruby

AWS SDK for Ruby: 개발자 가이드

Table of Contents

AWS SDK for Ruby란 무엇입니까?

AWS SDK for Ruby 개발자 안내서를 시작합니다. AWS SDK for Ruby는 Amazon Simple Storage Service(Amazon S3), Amazon Elastic Compute Cloud(Amazon EC2), Amazon DynamoDB를 AWS 서비스포함하여 거의 모든에 대한 지원 라이브러리를 제공합니다.

AWS SDK for Ruby 개발자 안내서는 AWS SDK for Ruby를 설치, 설정 및 사용하여를 사용하는 Ruby 애플리케이션을 생성하는 방법에 대한 정보를 제공합니다 AWS 서비스.

AWS SDK for Ruby 시작하기

추가 설명서 및 리소스

AWS SDK for Ruby 개발자를 위한 추가 리소스는 다음을 참조하세요.

- AWS SDKs 및 도구 참조 가이드 - AWS SDKs
- AWS SDK for Ruby API 참조 - 버전 3
- GitHub의 AWS 코드 예제 리포지토리
- RubyGems.org - 최신 버전의 SDK는 서비스별 gem으로 모듈화되어, 여기서 확인할 수 있습니다.
  - 지원되는 서비스 - AWS SDK for Ruby가 지원하는 모든 Gem을 나열합니다.
- AWS GitHub의 SDK for Ruby 소스:
  - 소스 및 README
  - 각 gem의 로그 변경
  - v2에서 v3로 이동
  - 문제
  - 코어 업그레이드 메모
- 개발자 블로그

AWS 클라우드에 배포

AWS Elastic Beanstalk 및와 AWS 서비스 같은 AWS CodeDeploy 를 사용하여 애플리케이션을 AWS 클라우드에 배포할 수 있습니다. Elastic Beanstalk를 사용하여 Ruby 애플리케이션을 배포하려면 AWS Elastic Beanstalk 개발자 안내서의 EB CLI 및 Git을 사용하여 Ruby에서 Elastic Beanstalk 애플리케이션 배포를 참조하세요. AWS 배포 서비스의 개요는 Overview of Deployment Options on AWS를 참조하세요.

SDK 메이저 버전에 대한 유지 관리 및 지원

SDK 메이저 버전 및 기본 종속성의 유지 관리 및 지원에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서에서 다음 내용을 참조하세요.

- AWS SDKs 및 도구 유지 관리 정책
- AWS SDKs 및 도구 버전 지원 매트릭스

AWS SDK for Ruby 시작하기

SDK를 설치, 설정 및 사용하여 프로그래밍 방식으로 AWS 리소스에 액세스하는 Ruby 애플리케이션을 생성하는 방법을 알아봅니다.

주제

- AWS SDK for Ruby를 AWS 사용하여 로 인증
- AWS SDK for Ruby 설치
- AWS SDK for Ruby를 사용하여 간단한 애플리케이션 생성

AWS SDK for Ruby를 AWS 사용하여 로 인증

를 사용하여 개발할 AWS 때 코드가 로 인증되는 방법을 설정해야 합니다 AWS 서비스. 환경 및 사용 가능한 액세스에 따라 다양한 방식으로 AWS 리소스에 대한 프로그래밍 방식 AWS 액세스를 구성할 수 있습니다.

인증 방법을 선택하고 SDK에 맞게 구성하려면AWS SDK 및 도구 참조 안내서의 Authentication and access를 참조하세요.

콘솔 자격 증명 사용

로컬 개발의 경우 새 사용자는 기존 AWS Management Console 로그인 자격 증명을 사용하여 AWS 서비스에 프로그래밍 방식으로 액세스하는 것이 좋습니다. 브라우저 기반 인증 후는 AWS 명령줄 인터페이스(AWS CLI) 및 AWS SDK for Ruby와 같은 로컬 개발 도구에서 작동하는 임시 자격 증명을 AWS 생성합니다.

이 방법을 선택하는 경우 AWS CLI를 사용하여 콘솔 자격 증명을 사용하여 AWS 로컬 개발에 로그인 지침을 따릅니다.

AWS SDK for Ruby는 콘솔 자격 증명으로 로그인을 사용하기 위해 애플리케이션에 추가 젬(예: `aws-sdk-signin`)을 추가할 필요가 없습니다.

IAM Identity Center 인증 사용

이 방법을 선택하는 경우 AWS SDK 및 도구 참조 가이드의 IAM Identity Center 인증 절차를 완료합니다. 그런 다음 환경에 다음 요소가 포함되어야 합니다.

- 애플리케이션을 실행하기 전에 AWS 액세스 포털 세션을 시작하는 데 AWS CLI사용하는 입니다.
- SDK에서 참조할 수 있는 구성 값 세트가 포함된 `[default]` 프로필이 있는 shared AWSconfig file입니다. 이 파일의 위치를 찾으려면 AWS SDK 및 도구 참조 가이드에서 공유 파일의 위치를 참조하세요.
- 공유 `config` 파일은 `region` 설정을 지정합니다. 이렇게 하면 SDK AWS 리전 가 AWS 요청에 사용하는 기본값이 설정됩니다. 이 리전은 사용할 리전이 지정되지 않은 SDK 서비스 요청에 사용됩니다.
- SDK는 AWS에 요청을 보내기 전에 프로필의 SSO token provider configuration을 사용하여 보안 인증을 얻습니다. IAM Identity Center 권한 세트에 연결된 IAM 역할인 `sso_role_name` 값은 애플리케이션에 AWS 서비스 사용되는에 대한 액세스를 허용합니다.

  다음 샘플 `config` 파일은 SSO 토큰 공급자 구성으로 설정된 기본 프로필을 보여줍니다. 프로필의 `sso_session` 설정은 이름이 지정된 `sso-session` section을 참조합니다. `sso-session` 섹션에는 AWS 액세스 포털 세션을 시작하는 설정이 포함되어 있습니다.

```
[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
```

AWS SDK for Ruby는 IAM Identity Center 인증을 사용하기 위해 애플리케이션에 추가 젬(예: `aws-sdk-sso` 및 `aws-sdk-ssooidc`)을 추가할 필요가 없습니다.

AWS 액세스 포털 세션 시작

에 액세스하는 애플리케이션을 실행하기 전에 SDK가 IAM Identity Center 인증을 사용하여 자격 증명을 확인하려면 활성 AWS 액세스 포털 세션이 AWS 서비스필요합니다. 구성된 세션 길이에 따라 결국 액세스가 만료되고 SDK에 인증 오류가 발생합니다. AWS 액세스 포털에 로그인하려면에서 다음 명령을 실행합니다 AWS CLI.

```
aws sso login
```

지침에 따라 기본 프로필을 설정했다면 `--profile` 옵션으로 명령을 직접적으로 호출할 필요가 없습니다. SSO 토큰 공급자 구성에서 명명된 프로필을 사용하는 경우 `aws sso login --profile named-profile` 명령을 사용합니다.

필요할 경우 이미 활성 세션이 있는지 테스트하려면 다음 AWS CLI 명령을 실행합니다.

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

세션이 활성 상태인 경우 이 명령에 대한 응답은 공유 `config` 파일에 구성된 IAM Identity Center 계정 및 권한 집합을 보고합니다.

> **Note**
>
> 이미 활성 AWS 액세스 포털 세션이 있고를 실행하는 경우 `aws sso login`자격 증명을 제공할 필요가 없습니다.
> 로그인 프로세스에서 데이터에 대한 AWS CLI 액세스를 허용하라는 메시지가 표시될 수 있습니다. AWS CLI 는 SDK for Python을 기반으로 구축되므로 권한 메시지에 `botocore` 이름의 변형이 포함될 수 있습니다.

세부 인증 정보

인간 사용자(인간 ID라고도 함)는 애플리케이션의 사용자, 관리자, 개발자, 운영자 및 소비자입니다. AWS 환경 및 애플리케이션에 액세스하려면 자격 증명이 있어야 합니다. 조직의 구성원인 인간 사용자, 즉 개발자는 작업 인력 ID라고도 합니다.

에 액세스할 때 임시 자격 증명을 사용합니다 AWS. 인간 사용자의 자격 증명 공급자를 사용하여 임시 자격 증명을 제공하는 역할을 수임하여 AWS 계정에 대한 페더레이션 액세스를 제공할 수 있습니다. 중앙 액세스 관리를 위해 AWS IAM Identity Center (IAM Identity Center)을 사용하여 계정에 대한 액세스 권한과 해당 계정 내 권한을 관리하는 것이 좋습니다. 더 많은 대안을 보려면 다음을 참조하세요.

- 모범 사례에 대해 자세히 알아보려면 IAM 사용 설명서에서 IAM의 보안 모범 사례를 참조하세요.
- 단기 AWS 자격 증명을 생성하려면 IAM 사용 설명서의 임시 보안 자격 증명을 참조하세요.
- AWS SDK for Ruby 자격 증명 공급자 체인과 SDK가 시퀀스에서 다양한 인증 방법을 자동으로 시도하는 방법에 대해 알아보려면 섹션을 참조하세요보안 인증 공급자 체인.
- AWS SDK 자격 증명 공급자 구성 설정은 SDK 및 도구 참조 안내서의 표준화된 자격 증명 공급자를 참조하세요. AWS SDKs

AWS SDK for Ruby 설치

이 섹션에는 AWS SDK for Ruby의 사전 조건 및 설치 지침이 포함되어 있습니다.

사전 조건

AWS SDK for Ruby를 사용하기 전에 먼저를 인증해야 합니다 AWS. 인증 설정에 대한 자세한 내용은 AWS SDK for Ruby를 AWS 사용하여 로 인증를 참조하세요.

SDK 설치

Ruby Gem과 마찬가지로 AWS SDK for Ruby를 설치할 수 있습니다. gem은 RubyGems에서 확인할 수 있습니다. AWS SDK for Ruby는 모듈식으로 설계되었으며 로 구분됩니다 AWS 서비스. 전체 `aws-sdk` gem을 설치하는 데는 많은 시간이 소요되며 한 시간 이상 걸릴 수 있습니다.

사용하는에 대해서만 젬을 설치하는 AWS 서비스 것이 좋습니다. 이러한 이름은와 같`aws-sdk-`*`service_abbreviation`*으며 전체 목록은 AWS SDK for Ruby README 파일의 지원되는 서비스 테이블에서 확인할 수 있습니다. 예를 들어 Amazon S3 서비스와 인터페이스하기 위한 gem은 `aws-sdk-s3`에서 직접 사용할 수 있습니다.

Ruby 버전 관리자

시스템 Ruby를 사용하는 대신 다음과 같은 Ruby 버전 관리자를 사용하는 것이 좋습니다.

- RVM
- chruby
- rbenv

예를 들어 Amazon Linux 2 운영 체제를 사용하는 경우 다음 명령을 사용하여 RVM을 업데이트하고 사용 가능한 Ruby 버전을 나열한 다음 AWS SDK for Ruby를 사용하여 개발하는 데 사용할 버전을 선택할 수 있습니다. 필요한 최소 Ruby 버전은 2.5입니다.

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

Bundler

Bundler를 사용하는 경우 다음 명령은 AWS SDK for Ruby Gem for Amazon S3를 설치합니다.

1. Bundler를 설치하고 `Gemfile`을 생성합니다.

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

2. 생성된를 열고 코드가 사용할 각 AWS 서비스 젬에 대한 `gem` 줄을 `Gemfile` 추가합니다. Amazon S3 예제를 사용하여 따라하려면 파일 하단에 다음과 같은 줄을 추가합니다.

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

3. Gemfile을 저장합니다.
4. `Gemfile`에 지정된 종속성을 설치합니다.

```
$ bundle install
```

AWS SDK for Ruby를 사용하여 간단한 애플리케이션 생성

AWS SDK for Ruby를 사용하여 Amazon S3에 인사합니다. 다음 예제에서는 Amazon S3 버킷의 목록을 표시합니다.

코드 작성

다음 코드를 복사하여 새로운 소스 파일에 붙여 넣습니다. 파일 이름을 `hello-s3.rb`으로 지정합니다.

```
require 'aws-sdk-s3'

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

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

```
  # Lists buckets for the current account.
  #
  # @param count [Integer] The maximum number of buckets to list.
  def list_buckets(count)
    puts 'Found these buckets:'
    @s3_resource.buckets.each do |bucket|
      puts "\t#{bucket.name}"
      count -= 1
      break if count.zero?
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list buckets. Here's why: #{e.message}"
    false
  end
end

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

run_demo if $PROGRAM_NAME == __FILE__
```

AWS SDK for Ruby는 모듈식으로 설계되었으며 로 구분됩니다 AWS 서비스. gem이 설치된 후 Ruby 소스 파일 상단에 있는 `require` 명령문을 통해 Amazon S3 서비스용 AWS SDK 클래스 및 메서드를 가져옵니다. 사용 가능한 AWS 서비스 Gem의 전체 목록은 AWS SDK for Ruby README 파일의 지원되는 서비스 테이블을 참조하세요.

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

프로그램 실행

명령 프롬프트를 열어 Ruby 프로그램을 실행합니다. Ruby 프로그램을 실행하는 일반적인 명령 구문은 다음과 같습니다.

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

이 샘플 코드는 인수를 사용하지 않습니다. 이 코드를 실행하려면 명령 프롬프트에 다음을 입력합니다.

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

Windows 사용자를 위한 참고 사항

Windows에서 SSL 인증서를 사용하고 Ruby 코드를 실행하면 다음과 비슷한 오류가 표시될 수 있습니다.

```
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'
...
```

이 문제를 해결하려면 첫 번째 AWS 호출 전에 Ruby 소스 파일에 다음 줄을 추가합니다.

```
Aws.use_bundled_cert!
```

Ruby 프로그램에 `aws-sdk-s3` gem만 사용하는 경우, 번들 인증서를 사용하려면 `aws-sdk-core` gem도 추가해야 합니다.

다음 단계

다른 많은 Amazon S3 작업을 테스트하려면 GitHub의 AWS 코드 예제 리포지토리를 확인하세요.

AWS SDK for Ruby에서 서비스 클라이언트 구성

프로그래밍 방식으로 AWS 서비스에 액세스하기 위해 AWS SDK for Ruby는 각 AWS 서비스에 클라이언트 클래스를 사용합니다. 예를 들어, 애플리케이션이 Amazon EC2에 액세스해야 하는 경우, 애플리케이션은 Amazon EC2 클라이언트 객체를 생성하여 해당 서비스와 인터페이스 합니다. 그런 다음 서비스 클라이언트를 사용하여 요청을 AWS 서비스에 보내면 됩니다.

AWS 서비스에 요청하려면 먼저 서비스 클라이언트를 생성해야 합니다. 코드가 사용하는 각 AWS 서비스에는 고유한 Gem과 상호 작용을 위한 전용 유형이 있습니다. 클라이언트는 서비스에서 노출되는 각 API 작업에 대해 하나의 메서드를 노출합니다.

SDK 동작을 구성하는 방법은 다양하지만, 궁극적으로 모든 것은 서비스 클라이언트의 동작과 관련이 있습니다. 구성에서 생성된 서비스 클라이언트가 사용될 때까지 해당 구성은 적용되지 않습니다.

AWS 서비스로 개발할 때는 코드가 AWS에서 인증되는 방법을 설정해야 합니다. 사용하려는 AWS 리전도 설정해야 합니다.

AWS SDK 및 도구 참조 안내서에는 많은 AWS SDK에 공통적인 설정, 기능 및 기타 기본 개념도 포함되어 있습니다.

주제

- 설정의 우선 순위
- 외부에서 AWS SDK for Ruby 서비스 클라이언트 구성
- 코드에서 AWS SDK for Ruby 서비스 클라이언트 구성
- AWS SDK for Ruby에 AWS 리전 대한 설정
- AWS SDK for Ruby 자격 증명 공급자 사용
- AWS SDK for Ruby에서 재시도 구성
- AWS SDK for Ruby에서 관찰성 기능 구성
- AWS SDK for Ruby 내에서 HTTP 수준 설정 구성

공유 `config` 및 `credentials` 파일을 구성 설정에 사용할 수 있습니다. 모든 AWS SDK 설정은 AWS SDK 및 도구 참조 가이드의 설정 참조를 참조하세요.

서로 다른 프로파일을 사용하여 서로 다른 구성을 저장할 수 있습니다. SDK가 로드할 활성 프로파일을 지정하려면 `AWS_PROFILE` 환경 변수 또는 `Aws.config`의 `profile` 옵션을 사용합니다.

설정의 우선 순위

글로벌 설정은 대부분의 SDK가 지원하는 기능, 보안 인증 공급자 및 기타 기능을 구성하며 AWS 서비스 전반에 광범위하게 영향을 미칩니다. 모든 AWS SDK에는 글로벌 설정 값을 찾기 위해 확인하는 일련의 위치(또는 소스)가 있습니다. 모든 소스에서 모든 설정을 사용할 수 있는 것은 아닙니다. 조회 우선 순위 설정은 다음과 같습니다.

1. 코드나 서비스 클라이언트 자체에 설정된 모든 명시적 설정은 다른 모든 설정보다 우선합니다.
   a. 클라이언트 생성자에 직접 전달되는 모든 파라미터가 가장 우선합니다.
   b. `Aws.config`에서 글로벌 또는 서비스별 설정이 확인됩니다.
2. 환경 변수를 확인합니다.
3. 공유 AWS `credentials` 파일이 확인됩니다.
4. 공유 AWS `config` 파일이 확인됩니다.
5. AWS SDK for Ruby 소스 코드 자체에서 제공하는 모든 기본값이 마지막에 사용됩니다.

외부에서 AWS SDK for Ruby 서비스 클라이언트 구성

코드 외부에서 많은 구성 설정을 처리할 수 있습니다. 구성이 외부에서 처리되면 모든 애플리케이션에 구성이 적용됩니다. 대부분의 구성 설정은 환경 변수로 설정하거나 별도의 공유 AWS `config` 파일로 설정할 수 있습니다. 공유 `config` 파일은 프로파일이라는 별도의 설정 세트를 유지하여 다양한 환경 또는 테스트에서 다양한 구성을 제공할 수 있습니다.

환경 변수와 공유 `config` 파일 설정은 표준화되어 있으며 다양한 프로그래밍 언어와 애플리케이션에서 일관된 기능을 지원하기 위해 AWS SDK 및 도구 전반에 걸쳐 공유됩니다.

이러한 메서드를 통해 애플리케이션을 구성하는 방법과 교차 SDK 설정에 대한 자세한 정보를 알아보려면 AWS SDK 및 도구 참조 가이드를 참조하세요. 환경 변수 또는 구성 파일에서 SDK가 확인할 수 있는 모든 설정을 보려면 AWS SDK 및 도구 참조 가이드에 나와 있는 설정 참조를 참조하세요.

AWS 서비스에 요청하려면 먼저 해당 서비스의 클라이언트를 인스턴스화합니다. 제한 시간, HTTP 클라이언트 및 재시도 구성을 비롯하여 서비스 클라이언트에 대한 공통 설정을 구성할 수 있습니다.

각 서비스 클라이언트에는 AWS 리전 및 자격 증명 공급자가 필요합니다. SDK는 이러한 값을 사용하여 리소스의 올바른 리전으로 요청을 보내고 올바른 자격 증명으로 요청에 서명합니다. 이러한 값은 코드에서 프로그래밍 방식으로 지정하거나 환경에서 자동으로 로드할 수 있습니다.

SDK에는 구성 설정 값을 찾기 위해 확인하는 일련의 위치(또는 소스)가 있습니다.

1. 코드나 서비스 클라이언트 자체에 설정된 모든 명시적 설정은 다른 모든 설정보다 우선합니다.
2. 환경 변수
   - 환경 변수를 설정하는 방법에 대한 자세한 내용은 AWS SDK 및 도구 참조 가이드에 나와 있는 환경 변수 섹션을 참조하세요.
   - 쉘 환경 변수는 시스템 전체, 사용자 전체, 특정 터미널 세션 등 다양한 수준에서 구성할 수 있습니다.
3. 공유 `config` 및 `credentials` 파일
   - 이 파일 설정에 관한 자세한 정보를 알아보려면 AWS SDK 및 도구 참조 가이드에 나와 있는 공유 `config` 및 `credentials` 파일 섹션을 참조하세요.
4. SDK 소스 코드 자체에서 제공하는 모든 기본값이 마지막에 사용됩니다.
   - 리전과 같은 일부 속성에는 기본값이 없습니다. 코드, 환경 설정 또는 공유 `config` 파일에서 이를 명시적으로 지정해야 합니다. SDK가 필요한 구성을 확인할 수 없는 경우 API 요청이 런타임에 실패할 수 있습니다.

AWS SDK for Ruby 환경 변수

대부분의 AWS SDK에서 지원되는 교차 sdk 환경 변수 외에도 AWS SDK for Ruby는 몇 가지 고유한 변수를 지원합니다.

`AWS_SDK_CONFIG_OPT_OUT`

AWS SDK for Ruby 환경 변수 `AWS_SDK_CONFIG_OPT_OUT`이 설정된 경우 일반적으로 `~/.aws/config`에 있는 공유 AWS `config` 파일은 구성 값에 사용되지 않습니다.

`AMAZON_REGION`

AWS 리전을 설정하기 위한 `AWS_REGION`의 대체 환경 변수입니다. 이 값은 `AWS_REGION`을 사용하지 않는 경우에만 확인됩니다.

코드에서 AWS SDK for Ruby 서비스 클라이언트 구성

구성을 코드 내에서 직접 처리할 경우 구성 범위는 해당 코드를 사용하는 애플리케이션으로 제한됩니다. 해당 애플리케이션 내에는 모든 서비스 클라이언트의 글로벌 구성, 특정 AWS 서비스 유형의 모든 클라이언트에 대한 구성 또는 특정 서비스 클라이언트 인스턴스에 대한 구성 옵션이 있습니다.

Aws.config

코드 내에서 모든 AWS 클래스에 대한 전역 구성을 제공하려면 aws-sdk-core Gem에서 사용할 수 Aws.config 있는를 사용합니다.

Aws.config는 다양한 용도를 위해 두 가지 구문을 지원합니다. 글로벌 설정은 모든 AWS 서비스 또는 특정 서비스에 적용할 수 있습니다. 지원되는 설정의 전체 목록은 AWS SDK for Ruby API 참조의 Client Options 섹션을 참조하세요.

Aws.config를 통한 글로벌 설정

Aws.config를 통해 서비스에 구애받지 않는 설정을 지정하려면 다음 구문을 사용합니다.

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

이러한 설정은 생성된 모든 서비스 클라이언트에 병합됩니다.

글로벌 설정의 예:

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

전역적으로 지원되지 않는 설정 이름을 사용하려고 하면 해당 이름을 지원하지 않는 서비스 유형의 인스턴스를 생성하려고 할 때 오류가 발생합니다. 이 경우 서비스별 구문을 대신 사용합니다.

Aws.config를 통한 서비스별 설정

Aws.config를 통해 서비스별 설정을 지정하려면 다음 구문을 사용합니다.

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

이러한 설정은 해당 서비스 유형의 생성된 모든 서비스 클라이언트에 병합됩니다.

Amazon S3에만 적용되는 설정의 예:

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

*<service identifier>*는 해당 AWS SDK for Ruby Gem 이름의 이름을 보고 "aws-sdk-" 뒤에 오는 접미사를 사용하여 식별할 수 있습니다. 예제:

- aws-sdk-s3의 경우 서비스 식별자 문자열은 "s3"입니다.
- aws-sdk-ecs의 경우 서비스 식별자 문자열은 "ecs"입니다.

AWS SDK for Ruby에 AWS 리전 대한 설정

를 사용하여 특정 지리적 영역에서 AWS 서비스 작동하는에 액세스할 수 있습니다 AWS 리전. 이는 중복성은 물론, 데이터와 애플리케이션을 고객과 고객의 사용자가 액세스할 위치에 가까운 곳에서 실행 상태로 유지하는 데도 유용할 수 있습니다.

> **Important**
>
> 대부분의 리소스는 특정에 상주 AWS 리전 하며 SDK를 사용할 때 리소스에 대한 올바른 리전을 제공해야 합니다.

AWS 요청에 AWS 리전 사용할 SDK for Ruby의 기본값을 설정해야 합니다. 이 기본값은 리전이 지정되지 않은 모든 SDK 서비스 메서드 직접 호출에 사용됩니다.

`region` 설정에 대한 자세한 내용은AWS SDK 및 도구 참조 안내서의 AWS 리전을 참조하세요. 여기에는 공유 AWS `config` 파일 또는 환경 변수를 통해 기본 리전을 설정하는 방법에 대한 예제도 포함됩니다.

확인을 위한 리전 검색 순서

대부분의 AWS 서비스를 사용할 때에는 리전을 설정해야 합니다. AWS SDK for Ruby는 다음 순서로 리전을 검색합니다.

1. 클라이언트 또는 리소스 객체에서 리전 설정
2. 를 사용하여 리전 설정 `Aws.config`
3. 환경 변수를 사용하여 리전 설정
4. 공유 `config` 파일을 사용하여 리전 설정

리전 설정 방법

이 섹션에서는 가장 일반적인 접근 방식부터 시작해 리전을 설정하는 여러 가지 방법을 설명합니다.

공유 **`config`** 파일을 사용하여 리전 설정

공유 AWS `config` 파일에서 `region` 변수를 설정하여 리전을 설정합니다. 공유 `config` 파일에 대한 자세한 내용은AWS SDK 및 도구 참조 안내서의 Shared config and credentials files를 참조하세요.

config 파일에서 이 값을 설정하는 예:

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

환경 변수 AWS_SDK_CONFIG_OPT_OUT이 설정되어 있으면 공유 config 파일을 확인하지 않습니다.

환경 변수를 사용하여 리전 설정

AWS_REGION 환경 변수를 설정해 리전을 설정합니다.

Linux 또는 macOS와 같은 Unix 기반 시스템에서 export 명령을 사용하여 이 변수를 설정합니다. 다음 예제는 리전을 us-west-2로 설정합니다.

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

Windows에서 이러한 변수를 설정하려면 set 명령을 사용합니다. 다음 예제는 리전을 us-west-2로 설정합니다.

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

Aws.config로 리전 설정

Aws.config 해시에 region 값을 추가해 리전을 설정합니다. 다음 예제에서는 us-west-1 리전을 사용하도록 Aws.config 해시를 업데이트합니다.

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

이후에 생성하는 클라이언트나 리소스는 이 리전에 구속됩니다.

클라이언트 또는 리소스 객체에서 리전 설정

AWS 클라이언트 또는 리소스를 생성할 때 리전을 설정합니다. 다음 예제는 us-west-1 리전에서 Amazon S3 리소스 객체를 생성합니다. AWS 리소스에 적합한 리전을 선택합니다. 서비스 클라이언트 객체는 변경할 수 없으므로 요청하는 각 서비스에 대한 새 클라이언트 및 다른 구성을 사용하여 동일한 서비스에 요청을 보낼 새 클라이언트를 만들어야 합니다.

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

AWS SDK for Ruby 자격 증명 공급자 사용

에 대한 모든 요청은에서 발급한 자격 증명을 사용하여 암호화 방식으로 서명해야 AWS 합니다 AWS. 런타임에 SDK는 여러 위치를 확인하여 자격 증명의 구성 값을 검색합니다.

를 사용한 인증은 코드베이스 외부에서 처리할 AWS 수 있습니다. SDK는 자격 증명 공급자 체인을 사용하여 많은 인증 방법을 자동으로 감지하고 사용하며 새로 고칠 수 있습니다.

프로젝트의 AWS 인증을 시작하기 위한 안내 옵션은 AWS SDKs 및 도구 참조 안내서의 인증 및 액세스를 참조하세요.

보안 인증 공급자 체인

모든 SDK에는 AWS 서비스에 요청하는 데 사용할 유효한 보안 인증을 얻기 위해 확인하는 일련의 장소(또는 소스)가 있습니다. 유효한 보안 인증 정보를 찾은 후에는 검색이 중지됩니다. 이러한 체계적인 검색을 기본 보안 인증 공급자 체인이라고 합니다.

> **Note**
>
> 새 사용자가 시작할 수 있도록 권장 접근 방식을 따른 경우에서 콘솔 자격 증명으로 로그인을 사용하여 인증한 것입니다AWS SDK for Ruby를 AWS 사용하여 로 인증. 상황에 따라 다른 인증 방법이 유용할 수 있습니다. 보안 위험을 방지하려면 항상 단기 보안 인증을 사용하는 것이 좋습니다. 다른 인증 방법 절차에 대해서는AWS SDK 및 도구 참조 안내서의 Authentication and access를 참조하세요.

체인의 각 단계마다 값을 설정하는 다양한 방법이 있습니다. 코드에서 직접 값을 설정하는 것이 항상 우선하며, 환경 변수로를 설정한 다음 공유 AWS `config` 파일에서를 설정합니다.

AWS SDKs 및 도구 참조 가이드에는 모든 AWS SDKs 및에서 사용하는 SDK 구성 설정에 대한 정보가 있습니다 AWS CLI. 공유 AWS `config` 파일을 통해 SDK를 구성하는 방법에 대한 자세한 내용은 공유 구성 및 자격 증명 파일을 참조하세요. 환경 변수 설정을 통해 SDK를 구성하는 방법에 관해 자세히 알아보려면 Environment variables support 단원을 참조하세요.

를 인증하기 위해 AWS SDK for Ruby AWS는 다음 표에 나열된 순서대로 자격 증명 공급자를 확인합니다.

우선 순위에 따른 보안 인증 공급자	AWS SDKs 및 도구 참조 가이드	AWS SDK for Ruby API Reference
AWS 액세스 키(임시 및 장기 자격 증명)	AWS 액세스 키	`Aws::Credentials` `Aws::SharedCredentials`
의 웹 자격 증명 토큰 AWS Security Token Service (AWS STS)	역할 보안 인증 공급자 위임 `role_arn`, `role_session_name` , 및 `web_identity_token_file` 사용	`Aws::AssumeRoleWebIdentityCredentials`
AWS IAM Identity Center. 이 안내서의 AWS SDK for Ruby를 AWS 사용하여 로 인증를 참조하세요.	IAM Identity Center 보안 인증 공급자	`Aws::SSOCredentials`
신뢰할 수 있는 엔터티 공급자 (예: `AWS_ROLE_ARN`). 이 안내서의 AWS STS 액세스 토큰 생성을 참조하세요.	역할 보안 인증 공급자 위임 `role_arn` 및 `role_session_name` 사용	`Aws::AssumeRoleCredentials`
로그인 자격 증명 공급자	로그인 자격 증명 공급자	`Aws::LoginCredentials`
프로세스 보안 인증 공급자	프로세스 보안 인증 공급자	`Aws::ProcessCredentials`
Amazon Elastic Container Service(Amazon ECS) 보안 인증	컨테이너 보안 인증 공급자	`Aws::ECSCredentials`
Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스 프로파일 보안 인증(IMDS 보안 인증 공급자)	IMDS 보안 인증 공급자	`Aws::InstanceProfileCredentials`

AWS SDK for Ruby 환경 변수AWS_SDK_CONFIG_OPT_OUT가 설정된 경우 일반적으로에 있는 공유 AWS config 파일은 자격 증명에 대해 구문 분석 ~/.aws/config되지 않습니다.

AWS STS 액세스 토큰 생성

역할을 수임하려면 일반적으로 액세스할 수 없는 AWS 리소스에 액세스하는 데 사용할 수 있는 임시 보안 자격 증명 세트를 사용해야 합니다. 이러한 임시 보안 인증은 액세스 키 ID, 보안 액세스 키 및 보안 토큰으로 구성됩니다. Aws::AssumeRoleCredentials 메서드를 사용하여 AWS Security Token Service (AWS STS) 액세스 토큰을 생성할 수 있습니다.

다음 예제에서는 액세스 토큰을 사용해 Amazon S3 클라이언트 객체를 생성합니다. 여기서 linked::account::arn은 위임할 역할의 Amazon 리소스 이름(ARN)이고 session-name은 위임된 역할 세션의 식별자입니다.

```
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)
```

role_arn 또는 설정에 AWS config 대한 자세한 내용은 SDK 및 도구 role_session_name참조 안내서의 역할 자격 증명 공급자 수임을 참조하세요. AWS SDKs

AWS SDK for Ruby에서 재시도 구성

AWS SDK for Ruby는 서비스 요청 및 사용자 지정 가능한 구성 옵션에 대한 기본 재시도 동작을 제공합니다. AWS 서비스를 직접적으로 호출하면 예기치 않은 예외가 가끔 반환됩니다. 직접 호출을 재시도하면 스로틀링 또는 일시적 오류와 같은 특정 유형의 오류가 성공할 수 있습니다.

공유 AWS config 파일의 환경 변수 또는 설정을 사용하여 전역적으로 재시도 동작을 구성할 수 있습니다. 이 접근법에 대한 정보는 AWS SDK 및 도구 참조 설명서의 재시도 동작을 참조하세요. 또한 재시도 전략 구현에 대한 자세한 정보와 적절한 구현을 선택하는 방법도 포함되어 있습니다.

또는 다음 섹션과 같이 코드에서 이러한 옵션을 구성할 수도 있습니다.

코드에서 클라이언트 재시도 동작 지정

기본적으로 AWS SDK for Ruby는 최대 3번의 재시도를 수행하며, 총 4번의 시도 동안 시도 간에 15초의 시간을 둡니다. 그래서 작업이 최대 60초 소요되어 시간 초과에 걸릴 수 있습니다.

다음 예제는 리전 `us-west-2`에서 Amazon S3 클라이언트를 생성하고 모든 클라이언트 작업에서 두 번의 시도 사이에 5초간 대기하도록 지정합니다. 그래서 Amazon S3 클라이언트 작업이 최대 15초 소요되어 시간 초과에 걸릴 수 있습니다.

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

코드나 서비스 클라이언트 자체에 설정된 모든 명시적 설정은 환경 변수 또는 공유 `config` 파일에 설정된 설정보다 우선합니다.

AWS SDK for Ruby에서 관찰성 기능 구성

관찰성은 시스템의 현재 상태를 내보내는 데이터에서 추론할 수 있는 범위입니다. 방출되는 데이터를 일반적으로 원격 측정이라고 합니다. AWS SDK for Ruby는 추적을 원격 측정 신호로 제공할 수 있습니다. `TelemetryProvider`를 연결해 원격 측정 데이터를 수집하고 관찰성 백엔드로 전송할 수 있습니다. SDK는 현재 원격 측정 공급자로 OpenTelemetry(OTel)를 지원하며 OpenTelemetry에는 AWS X-Ray 또는 Amazon CloudWatch를 사용하는 등 원격 측정 데이터를 내보내는 다양한 방법이 있습니다. Ruby용 OpenTelemetry 내보내기에 대한 자세한 내용은 OpenTelemetry 웹 사이트의 Exporters를 참조하세요.

기본적으로 SDK는 원격 측정 데이터를 기록하거나 내보내지 않습니다. 이 주제에서는 원격 측정 출력을 구성하고 내보내는 방법을 설명합니다.

특정 서비스에 대해 또는 전역적으로 원격 측정을 구성할 수 있습니다. SDK for Ruby는 OpenTelemetry 공급자를 제공합니다. 원하는 사용자 지정 원격 측정 공급자를 정의할 수도 있습니다.

서비스 클라이언트에 대한 `OTelProvider` 구성

SDK for Ruby는 `OTelProvider`라는 OpenTelemetry 공급자를 제공합니다. 다음 예제에서는 Amazon Simple Storage Service 서비스 클라이언트에 대해 OpenTelemetry를 사용하여 원격 측정 내보내기를 구성합니다. 이 간단한 예제에서는 코드를 실행할 때 OpenTelemetry의 `OTEL_TRACES_EXPORTER` 환경 변수를 사용하여 트레이스를 콘솔 출력으로 내보냅니다.

OTEL_TRACES_EXPORTER에 대한 자세한 내용은 OpenTelemetry 설명서의 Exporter Selection을 참조하세요.

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

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

OpenTelemetry::SDK.configure

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

이전 코드 예제에서는 서비스 클라이언트에 대해 트레이스 출력을 구성하는 단계를 보여줍니다.

1. OpenTelemetry 종속 항목이 필요합니다.
   a. `Aws::Telemetry::OTelProvider`를 사용하는 데 필요한 `opentelemetry-sdk`.
   b. 원격 측정 데이터를 내보내는 데 필요한 `opentelemetry-exporter-otlp`.
2. `OpenTelemetry::SDK.configure`를 직접적으로 호출하여 OpenTelemetry SDK를 구성 기본값으로 설정합니다.
3. SDK for Ruby의 OpenTelemetry 공급자를 사용해 `OTelProvider`의 인스턴스를 생성하여 추적하려는 서비스 클라이언트에 구성 옵션으로 전달합니다.

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

이 단계를 사용하면 해당 서비스 클라이언트에서 직접적으로 호출되는 모든 메서드가 트레이스 데이터를 내보내게 됩니다.

Amazon S3의 `list_buckets` 메서드에 대한 직접 호출에서 생성된 트레이스 출력의 예는 다음과 같습니다.

OpenTelemetry 트레이스 출력 예

```
#<struct OpenTelemetry::SDK::Trace::SpanData
 name="Handler.NetHttp",
 kind=:internal,
```

```
 status=#<OpenTelemetry::Trace::Status:0x000000011da17bd8 @code=1, @description="">,
 parent_span_id="\xBFb\xC9\xFD\xA6F!\xE1",
 total_recorded_attributes=7,
 total_recorded_events=0,
 total_recorded_links=0,
 start_timestamp=1736190567061767000,
 end_timestamp=1736190567317160000,
 attributes=
  {"http.method"=>"GET",
   "net.protocol.name"=>"http",
   "net.protocol.version"=>"1.1",
   "net.peer.name"=>"s3.amazonaws.com",
   "net.peer.port"=>"443",
   "http.status_code"=>"200",
   "aws.request_id"=>"22HSH7NQTYMB5NHQ"},
 links=nil,
 events=nil,
 resource=
  #<OpenTelemetry::SDK::Resources::Resource:0x000000011e0bf990
   @attributes=
    {"service.name"=>"unknown_service",
     "process.pid"=>37013,
     "process.command"=>"example.rb",
     "process.runtime.name"=>"ruby",
     "process.runtime.version"=>"3.3.0",
     "process.runtime.description"=>"ruby 3.3.0 (2023-12-25 revision 5124f9ac75)
 [arm64-darwin23]",
     "telemetry.sdk.name"=>"opentelemetry",
     "telemetry.sdk.language"=>"ruby",
     "telemetry.sdk.version"=>"1.6.0"}>,
 instrumentation_scope=#<struct OpenTelemetry::SDK::InstrumentationScope
 name="aws.s3.client", version="">,
 span_id="\xEF%\x9C\xB5\x8C\x04\xDB\x7F",
 trace_id=" \xE7\xF1\xF8\x9D\e\x16/\xAC\xE6\x1A\xAC%j\x81\xD8",
 trace_flags=#<OpenTelemetry::Trace::TraceFlags:0x000000011d994328 @flags=1>,
 tracestate=#<OpenTelemetry::Trace::Tracestate:0x000000011d990638 @hash={}>>
#<struct OpenTelemetry::SDK::Trace::SpanData
 name="S3.ListBuckets",
 kind=:client,
 status=#<OpenTelemetry::Trace::Status:0x000000011da17bd8 @code=1, @description="">,
 parent_span_id="\x00\x00\x00\x00\x00\x00\x00\x00",
 total_recorded_attributes=5,
 total_recorded_events=0,
 total_recorded_links=0,
```

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

이전 트레이스 출력에는 두 가지 범위의 데이터가 있습니다. 각 트레이스 항목은 하나 이상의 속성에서 이벤트에 대한 추가 메타데이터를 제공합니다.

모든 서비스 클라이언트에 대한 **OTelProvider** 구성

이전 섹션에서 설명한 대로 특정 서비스 클라이언트에 대해 원격 측정을 켜는 대신 전역적으로 원격 측정을 켤 수 있습니다.

모든 AWS 서비스 클라이언트에 대한 원격 측정 데이터를 내보내려면 서비스 클라이언트를 생성하기 `Aws.config` 전에에서 원격 측정 공급자를 설정할 수 있습니다.

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

이 구성을 사용하면 이후에 생성된 모든 서비스 클라이언트가 자동으로 원격 측정을 내보냅니다. `Aws.config`를 사용하여 글로벌 설정을 지정하는 방법에 대한 자세한 내용은 `Aws.config` 섹션을 참조하세요.

사용자 지정 원격 측정 공급자 구성

OpenTelemetry를 원격 측정 공급자로 사용하지 않으려면 AWS SDK for Ruby에서 사용자 지정 공급자 구현을 지원합니다. AWS SDK for Ruby GitHub 리포지토리에서 사용할 수 있는 `OTelProvider` 구현을 예로 들면 도움이 될 수 있습니다. 추가 컨텍스트는 AWS SDK for Ruby API 참조에 있는 `Module: Aws::Telemetry`의 참고 사항을 확인하세요.

스팬 속성

트레이스는 원격 측정의 출력입니다. 트레이스는 하나 이상의 스팬으로 구성됩니다. 스팬에는 메서드 직접 호출에 적합한 경우 자동으로 포함되는 추가 메타데이터가 포함된 속성이 있습니다. 다음은 SDK for Ruby에서 지원하는 속성 목록입니다. 여기서:

- 속성 이름 - 트레이스에 나타나는 데이터를 레이블링하는 데 사용되는 이름입니다.
- 유형 - 값의 데이터 유형입니다.
- 설명 - 값이 나타내는 내용에 대한 설명입니다.

Attribute Name	Type	Description
`##`	Boolean	True if the unit of work was unsuccessful. Otherwise, false.
`exception.message`	String	The exception or error message.
`exception.stacktrace`	String	A stacktrace as provided by the language runtime if available.
`exception.type`	String	The type (fully qualified name) of the exception or error.

`rpc.system`	String	The remote system identifier set to '`aws-api`'.
`rpc.method`	String	The name of the operation being invoked.
`rpc.service`	String	The name of the remote service.
`aws.request_id`	String	The AWS request ID returned in the response headers, per HTTP attempt. The latest request ID is used when possible.
`code.function`	String	The method or function name.
`code.namespace`	String	The namespace within which `code.function` is defined.
`http.status_code`	Long	The HTTP response status code.
`http.request_content_length`	Long	The size of the request payload body in bytes.
`http.response_content_length`	Long	The size of the response payload body in bytes.
`http.method`	String	The HTTP request method.
`net.protocol.name`	String	The name of the application layer protocol.
`net.protocol.version`	String	The version of the application layer protocol (e.g. 1.0, 1.1, 2.0).
`net.peer.name`	String	The logical remote hostname.

`net.peer.port`	String	The logical remote port number.

> **Tip**
>
> OpenTelemetry-Ruby에는 SDK for Ruby의 기존 원격 측정 지원과 통합된 추가 구현이 있습니다. 자세한 내용은 `open-telemetry` GitHub 리포지토리의 OpenTelemetry AWS-SDK Instrumentation을 참조하세요.

AWS SDK for Ruby 내에서 HTTP 수준 설정 구성

비표준 엔드포인트 설정

리전은 AWS 요청에 사용할 SSL 엔드포인트를 구성하는 데 사용됩니다. 선택한 리전에서 비표준 엔드포인트를 사용해야 하는 경우 `Aws.config`에 `endpoint` 입력을 추가하세요. 또는 서비스 클라이언트나 리소스 객체를 생성할 때 `endpoint:`를 설정하세요. 다음 예제는 `other_endpoint` 엔드포인트에서 Amazon S3 리소스 객체를 생성합니다.

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

API 요청에 대해 선택한 엔드포인트를 사용하고 해당 선택 사항을 계속 유지하려면AWS SDK 및 도구 참조 안내서의 Service-specific endpoints를 참조하세요.

AWS SDK for Ruby 사용

이 섹션에서는 AWS SDK의 일부 고급 기능을 사용하는 방법을 포함하여 SDK for Ruby를 사용하여 소프트웨어를 개발하는 방법에 대한 정보를 제공합니다.

AWS SDKs 및 도구 참조 가이드에는 많은 AWS SDKs.

주제

- AWS SDK for Ruby를 사용하여 AWS 서비스 요청
- AWS SDK for Ruby REPL 유틸리티 사용
- Ruby on Rails에서 Ruby용 AWS SDK 사용
- AWS SDK for Ruby 클라이언트의 유선 추적 정보를 사용하여 디버깅
- AWS SDK for Ruby 애플리케이션에 스텁을 사용한 테스트 추가
- AWS SDK for Ruby에서 페이지가 매겨진 결과 사용
- AWS SDK for Ruby에서 웨이터 사용

AWS SDK for Ruby를 사용하여 AWS 서비스 요청

프로그래밍 방식으로 AWS 서비스에 액세스하기 위해 SDK는 각각의 AWS 서비스에 대해 클라이언트 클래스를 사용합니다. 예를 들어, 애플리케이션이 Amazon EC2에 액세스해야 하는 경우, 애플리케이션은 Amazon EC2 클라이언트 객체를 생성하여 해당 서비스와 인터페이스 합니다. 그런 다음 서비스 클라이언트를 사용하여 요청을 AWS 서비스에 보내면 됩니다.

AWS 서비스에 요청하려면 먼저 서비스 클라이언트를 생성하고 구성해야 합니다. 코드가 사용하는 각 AWS 서비스에는 고유한 Gem과 상호 작용을 위한 전용 유형이 있습니다. 클라이언트는 서비스에서 노출되는 각 API 작업에 대해 하나의 메서드를 노출합니다.

각 서비스 클라이언트에는 AWS 리전 및 자격 증명 공급자가 필요합니다. SDK는 이러한 값을 사용하여 리소스의 올바른 리전으로 요청을 보내고 올바른 자격 증명으로 요청에 서명합니다. 이러한 값은 코드에서 프로그래밍 방식으로 지정하거나 환경에서 자동으로 로드할 수 있습니다.

- 클라이언트 클래스를 인스턴스화할 때 AWS 자격 증명을 제공해야 합니다. SDK가 인증 공급자를 확인하는 순서는 보안 인증 공급자 체인 섹션을 참조하세요.
- SDK에는 구성 설정 값을 찾기 위해 확인하는 일련의 위치(또는 소스)가 있습니다. 자세한 내용은 설정의 우선 순위을 참조하세요.

SDK for Ruby에는 AWS 서비스에 인터페이스를 제공하는 클라이언트 클래스가 포함되어 있습니다. 각 클라이언트 클래스는 특정 AWS 서비스를 지원하며 `Aws::<service identifier>::Client` 규칙을 따릅니다. 예를 들어 `Aws::S3::Client`는 Amazon Simple Storage Service 서비스에 대한 인터페이스를 제공하고 `Aws::SQS::Client`는 Amazon Simple Queue Service 서비스에 대한 인터페이스를 제공합니다.

모든 AWS 서비스의 모든 클라이언트 클래스는 스레드 세이프입니다.

구성 옵션을 클라이언트와 리소스 생성자에 직접 전달할 수 있습니다. 이러한 옵션은 환경 및 `Aws.config` 기본값보다 우선합니다.

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

AWS SDK for Ruby REPL 유틸리티 사용

`aws-sdk` gem에는 SDK for Ruby를 테스트하고 결과를 즉시 확인할 수 있는 Read-Eval-Print-Loop(REPL) 대화형 명령줄 인터페이스가 포함되어 있습니다. SDK for Ruby gem은 RubyGems.org에서 사용할 수 있습니다.

사전 조건

- AWS SDK for Ruby 설치.
- `aws-v3.rb`는 `aws-sdk-resources` gem에 있습니다. 메인 `aws-sdk` gem에도 `aws-sdk-resources` gem이 포함되어 있습니다.
- `rexml` gem과 같은 xml 라이브러리가 필요합니다.
- 프로그램이 Interactive Ruby Shell(`irb`)과 연동되기는 하지만 더 강력한 REPL 환경을 제공하는 `pry` gem을 설치하는 것이 좋습니다.

Bundler 설정

Bundler를 사용하는 경우 `Gemfile`에 다음 업데이트를 하면 전제 조건 gem을 해결할 수 있습니다.

1. AWS SDK for Ruby를 설치할 때 만든 `Gemfile`을 엽니다. 파일에 다음 줄을 추가합니다.

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

```
gem "pry"
```

2. Gemfile을 저장합니다.
3. `Gemfile`에 지정된 종속성을 설치합니다.

```
$ bundle install
```

REPL 실행

사용자는 명령줄에서 `aws-v3.rb`를 실행하여 REPL에 액세스할 수 있습니다.

```
aws-v3.rb
```

또는 verbose 플래그를 설정하여 HTTP 와이어 로깅을 사용할 수 있습니다. HTTP 와이어 로깅은 AWS SDK for Ruby와 AWS 간의 통신에 대한 정보를 제공합니다. verbose 플래그는 오버헤드를 가중시켜 코드 실행 속도를 늦출 수도 있습니다.

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

SDK for Ruby에는 AWS 서비스에 인터페이스를 제공하는 클라이언트 클래스가 포함되어 있습니다. 각 클라이언트 클래스는 특정 AWS 서비스를 지원합니다. REPL의 모든 서비스 클래스에는 해당 서비스와 상호작용하기 위한 새 클라이언트 객체를 반환하는 도우미가 있습니다. 도우미 이름은 소문자로 변환된 서비스 이름이 됩니다. 예를 들어 Amazon S3 및 Amazon EC2 도우미 객체의 이름은 각각 `s3` 및 `ec2`입니다. 계정에 있는 Amazon S3 버킷을 나열하려면 프롬프트에 `s3.list_buckets`를 입력하면 됩니다.

REPL 프롬프트에 `quit`를 입력하여 종료할 수 있습니다.

Ruby on Rails에서 Ruby용 AWS SDK 사용

Ruby on Rails는 Ruby로 손쉽게 웹 사이트를 만들 수 있게 해 주는 웹 개발 프레임워크를 제공합니다.

AWS 는 Rails와 쉽게 통합할 수 있도록 `aws-sdk-rails` Gem을 제공합니다. AWS Elastic Beanstalk AWS OpsWorks AWS CodeDeploy또는 AWS Rails Provisioner를 사용하여 AWS 클라우드에서 Rails 애플리케이션을 배포하고 실행할 수 있습니다.

`aws-sdk-rails` gem 설치 및 사용에 대한 자세한 내용은 GitHub 리포지토리(https://github.com/aws/aws-sdk-rails)를 참조하십시오.

AWS SDK for Ruby 클라이언트의 유선 추적 정보를 사용하여 디버깅

`http_wire_trace` 부울을 설정하여 AWS 클라이언트에서 유선 추적 정보를 가져올 수 있습니다. 와이어 트레이스 정보는 클라이언트 변경 사항, 서비스 문제, 사용자 오류를 구분하는 데 도움이 됩니다. `true`인 경우 설정에 와이어로 전송되는 내용이 표시됩니다. 다음 예제에서는 클라이언트 생성 시에 와이어 트레이싱을 사용하는 Amazon S3 클라이언트를 생성합니다.

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

다음 코드와 인수 `bucket_name`에 따라 출력에 해당 이름의 버킷이 있는지 여부를 알려 주는 메시지가 표시됩니다.

```
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
```

버킷이 있는 경우 출력은 다음과 비슷합니다. (가독성을 높이기 위해 `HEAD` 줄에 반환이 추가되었습니다.)

```
opening connection to bucket_name.s3-us-west-1.amazonaws.com:443...
opened
starting SSL for bucket_name.s3-us-west-1.amazonaws.com:443...
SSL established, protocol: TLSv1.2, cipher: ECDHE-RSA-AES128-GCM-SHA256
-> "HEAD / HTTP/1.1
    Accept-Encoding:
    User-Agent: aws-sdk-ruby3/3.171.0 ruby/3.2.2 x86_64-linux aws-sdk-s3/1.120.0
    Host: bucket_name.s3-us-west-1.amazonaws.com
    X-Amz-Date: 20230427T143146Z
/* omitted */
Accept: */*\r\n\r\n"
-> "HTTP/1.1 200 OK\r\n"
-> "x-amz-id-2: XxB2J+kpHgTjmMUwpkUI1EjaFSPxAjWRgkn/+z7YwWc/
iAX5E3OXRBzJ37cfc8T4D7ELC1KFELM=\r\n"
```

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

클라이언트 생성 후 와이어 트레이싱을 켤 수도 있습니다.

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

보고된 와이어 트레이스 정보의 필드에 대한 자세한 내용은 Transfer Family required request headers를 참조하세요.

AWS SDK for Ruby 애플리케이션에 스텁을 사용한 테스트 추가

AWS SDK for Ruby 애플리케이션에서 클라이언트 응답 및 클라이언트 오류를 스텁하는 방법을 알아봅니다.

클라이언트 응답 및 오류 스텁

응답을 스텁하면 AWS SDK for Ruby가 네트워크 트래픽을 비활성화하고 클라이언트가 스텁된(또는 거짓) 데이터를 반환합니다. 스텁된 데이터를 제공하지 않으면 클라이언트가 다음을 반환합니다.

- 빈 어레이인 목록
- 빈 해시인 맵
- 0인 숫자 값
- now인 데이터

다음 예제에서는 Amazon S3 버킷 목록에 대해 스텁된 이름을 반환합니다.

```
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
```

이 코드를 실행하면 다음이 표시됩니다.

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

> **Note**
>
> 스텁된 데이터를 제공하면 남아 있는 인스턴스 속성에 대해 기본값이 더 이상 적용되지 않습니다. 즉, 이전 예제에서 남은 인스턴스 속성 `creation_date`는 `now`가 아니라 `nil`입니다.

AWS SDK for Ruby는 스텁된 데이터를 검증합니다. 잘못된 유형의 데이터를 전달하면 `ArgumentError` 예외가 발생합니다. 예를 들어 `bucket_data`에 대한 이전 배정 대신 다음을 사용했습니다.

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

AWS SDK for Ruby는 두 가지 `ArgumentError` 예외를 생성합니다.

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

클라이언트 오류 스텁

AWS SDK for Ruby가 특정 메서드에 대해 발생시키는 오류를 스텁할 수도 있습니다. 다음 예제에서는 `Caught Timeout::Error error calling head_bucket on aws-sdk`를 표시합니다.

```
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
```

AWS SDK for Ruby에서 페이지가 매겨진 결과 사용

페이로드가 너무 커서 단일 응답으로 반환할 수 없는 경우 많은 AWS 작업이 잘린 결과를 반환합니다. 대신 서비스는 데이터의 일부와 토큰을 반환하여 다음 항목 세트를 검색합니다. 이 패턴을 페이지 매김이라고 합니다.

페이징된 응답은 열거 가능함

페이징된 응답 데이터를 처리하는 가장 간단한 방법은 다음 예제에서와 같이 응답 객체에 기본 제공되는 열거자를 사용하는 것입니다.

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

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

이렇게 하면 API 호출당 하나의 응답 개체가 생기며, 명명된 버킷에 객체가 열거됩니다. SDK는 데이터의 추가 페이지를 검색해 요청을 완료합니다.

페이징된 응답 수동 처리

페이징을 직접 처리하려면 응답의 `next_page?` 메서드를 사용하여 검색할 페이지가 더 있는지 확인하거나 `last_page?` 메서드를 사용하여 검색할 페이지가 더 이상 없는지 확인하십시오.

페이지가 더 있으면 `next_page`(?가 없음) 메서드를 사용하여 다음 예제에서와 같이 다음 결과 페이지를 검색하십시오.

```
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**
>
> `next_page` 메서드를 호출했는데 검색할 페이지가 더 이상 없으면 SDK가 Aws::PageableResponse::LastPageError 예외를 발생시킵니다.

페이징된 데이터 클래스

AWS SDK for Ruby의 페이지가 지정된 데이터는 Aws::PageableResponse 클래스에서 처리합니다.이 클래스는 페이지가 지정된 데이터에 대한 액세스를 제공하기 위해 Seahorse::Client::Response에 포함되어 있습니다.

AWS SDK for Ruby에서 웨이터 사용

Waiter는 클라이언트에서 특정 상태에 대해 폴링되는 유틸리티 메서드입니다. Waiter는 서비스 클라이언트에 대해 정의된 폴링 간격에서 몇 번의 시도 후에 실패할 수 있습니다. 웨이터 사용 방법에 대한 예는 AWS 코드 예제 리포지토리에서 Amazon DynamoDB Encryption Client의 create_table 메서드를 참조하세요.

Waiter 호출

Waiter를 호출하려면 서비스 클라이언트에서 `wait_until`을 호출하십시오. 다음 예제에서는 계속하기 전에 waiter가 인스턴스 `i-12345678`이 실행될 때까지 기다립니다.

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

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

첫 번째 파라미터는 서비스 클라이언트에 고유하고 어떤 작업을 기다리고 있는지 나타내는 waiter 이름입니다. 두 번째 파라미터는 waiter가 호출한 클라이언트 메서드에 전달되는 파라미터의 해시로, waiter 이름에 따라 다릅니다.

대기 가능한 작업 목록과 각 작업에 대해 호출된 클라이언트 메서드의 목록은 사용 중인 클라이언트의 `waiter_names` 및 `wait_until` 필드 설명서를 참조하십시오.

Wait 오류

다음 예외 사례의 경우 Waiter가 실패할 수 있습니다.

Aws::Waiters::Errors::FailureStateError

대기하는 중에 오류 상태가 발생했습니다.

Aws::Waiters::Errors::NoSuchWaiterError

지정된 waiter 이름이 사용 중인 클라이언트에 대해 정의되지 않았습니다.

Aws::Waiters::Errors::TooManyAttemptsError

시도 횟수가 waiter의 `max_attempts` 값을 초과했습니다.

Aws::Waiters::Errors::UnexpectedError

대기하는 중에 예상치 못한 오류가 발생했습니다.

Aws::Waiters::Errors::WaiterFailed

대기 상태 중 하나가 초과했거나 대기 중에 또 다른 오류가 발생했습니다.

이 모든 오류(`NoSuchWaiterError` 제외)는 `WaiterFailed`를 기반으로 합니다. Waiter에서 오류를 잡아내려면 다음 예제에 표시된 대로 `WaiterFailed`를 사용하십시오.

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

Waiter 구성

각 waiter에는 기본 폴링 간격과 프로그램에 제어권을 반환하기 전 최대 시도 횟수가 적용됩니다. 이 값을 설정하려면 `max_attempts` 호출에서 `delay:` 및 `wait_until`를 사용하십시오. 다음 예에서는 최대 25초 동안 대기하면서 5초마다 폴링합니다.

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

대기 오류를 비활성화하려면 이 파라미터 중 하나의 값을 `nil`로 설정하십시오.

Waiter 연장

Waiter의 동작을 수정하기 위해 각 폴링 시도 및 대기 전에 트리거되는 콜백을 등록할 수 있습니다.

다음 예제는 시도마다 대기하는 시간을 두 배로 늘려 waiter에 지수 백오프를 구현합니다.

```
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
```

다음 예제는 최대 시도 횟수를 비활성화하고 그 대신 실패하기 전에 1시간(3600초) 동안 대기합니다.

```
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
```

SDK for Ruby 코드 예제

이 주제의 코드 예제에서는를 AWS SDK for Ruby 와 함께 사용하는 방법을 보여줍니다 AWS.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

일부 서비스에는 서비스와 관련된 라이브러리 또는 함수를 활용하는 방법을 보여주는 추가 예제 범주가 포함되어 있습니다.

서비스

- SDK for Ruby를 사용한 Aurora 예제
- SDK for Ruby를 사용한 Auto Scaling 예제
- SDK for Ruby를 사용한 CloudTrail 예제
- SDK for Ruby를 사용한 CloudWatch 예제
- SDK for Ruby를 사용한 Amazon Cognito ID 공급자 예제
- SDK for Ruby를 사용한 Amazon Comprehend 예제
- SDK for Ruby를 사용한 Amazon DocumentDB 예제
- SDK for Ruby를 사용한 DynamoDB 예제
- SDK for Ruby를 사용한 Amazon EC2 예제
- SDK for Ruby를 사용한 Elastic Beanstalk 예제
- SDK for Ruby를 사용한 EventBridge 예제
- AWS Glue SDK for Ruby를 사용한 예제
- SDK for Ruby를 사용한 IAM 예제
- SDK for Ruby를 사용한 Kinesis 예제
- AWS KMS SDK for Ruby를 사용한 예제
- SDK for Ruby를 사용한 Lambda 예제
- SDK for Ruby를 사용한 Amazon MSK 예제
- SDK for Ruby를 사용한 Amazon Polly 예제

- SDK for Ruby를 사용한 Amazon RDS 예제
- SDK for Ruby를 사용한 Amazon S3 예제
- SDK for Ruby를 사용한 Amazon SES 예제
- SDK for Ruby를 사용한 Amazon SES API v2 예제
- SDK for Ruby를 사용한 Amazon SNS 예제
- SDK for Ruby를 사용한 Amazon SQS 예제
- AWS STS SDK for Ruby를 사용한 예제
- SDK for Ruby를 사용한 Amazon Textract 예제
- SDK for Ruby를 사용한 Amazon Translate 예제

SDK for Ruby를 사용한 Aurora 예제

다음 코드 예제에서는 Aurora와 AWS SDK for Ruby 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시작하기

시작하기

Hello Aurora

다음 코드 예제에서는 Aurora를 사용하여 시작하는 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-rds'
```

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

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

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

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 DescribeDBClusters를 참조하세요.

SDK for Ruby를 사용한 Auto Scaling 예제

다음 코드 예제에서는 Auto Scaling과 AWS SDK for Ruby 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시작하기

시작하기

Auto Scaling 시작

다음 코드 예제에서는 Auto Scaling 사용을 시작하는 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-autoscaling'
require 'logger'

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

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

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

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

- API에 대한 세부 정보는 AWS SDK for Ruby API 참조의 DescribeAutoScalingGroups를 참조하세요.

SDK for Ruby를 사용한 CloudTrail 예제

다음 코드 예제에서는 CloudTrail과 AWS SDK for Ruby 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`CreateTrail`

다음 코드 예시는 `CreateTrail`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 CreateTrail을 참조하세요.

`DeleteTrail`

다음 코드 예시는 `DeleteTrail`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

```
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteTrail을 참조하세요.

`ListTrails`

다음 코드 예시는 `ListTrails`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListTrails을 참조하세요.

`LookupEvents`

다음 코드 예시는 `LookupEvents`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 LookupEvents를 참조하세요.

SDK for Ruby를 사용한 CloudWatch 예제

다음 코드 예제에서는 CloudWatch와 AWS SDK for Ruby 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`DescribeAlarms`

다음 코드 예시는 `DescribeAlarms`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-cloudwatch'

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 DescribeAlarms를 참조하세요.

`DescribeAlarmsForMetric`

다음 코드 예시는 `DescribeAlarmsForMetric`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

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

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

```
        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__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DescribeAlarmsForMetric을 참조하세요.

`DisableAlarmActions`

다음 코드 예시는 `DisableAlarmActions`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# Disables an alarm in Amazon CloudWatch.
#
# Prerequisites.
#
# - The alarm to disable.
#
# @param cloudwatch_client [Aws::CloudWatch::Client]
#   An initialized CloudWatch client.
# @param alarm_name [String] The name of the alarm to disable.
# @return [Boolean] true if the alarm was disabled; otherwise, false.
```

```
# @example
#   exit 1 unless alarm_actions_disabled?(
#     Aws::CloudWatch::Client.new(region: 'us-east-1'),
#     'ObjectsInBucket'
#   )
def alarm_actions_disabled?(cloudwatch_client, alarm_name)
  cloudwatch_client.disable_alarm_actions(alarm_names: [alarm_name])
  true
rescue StandardError => e
  puts "Error disabling alarm actions: #{e.message}"
  false
end

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

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

- API 세부 정보에 대한 내용은 AWS SDK for Ruby API 참조의 DisableAlarmActions를 참조하세요.

`ListMetrics`

다음 코드 예시는 `ListMetrics`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

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

# Example usage:
def run_me
```

```
  metric_namespace = 'SITE/TRAFFIC'
  # Replace us-west-2 with the AWS Region you're using for Amazon CloudWatch.
  region = 'us-east-1'

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

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

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListMetrics를 참조하세요.

`PutMetricAlarm`

다음 코드 예시는 `PutMetricAlarm`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

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

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 PutMetricAlarm을 참조하세요.

`PutMetricData`

다음 코드 예시는 `PutMetricData`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-cloudwatch'

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

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 PutMetricData를 참조하세요.

SDK for Ruby를 사용한 Amazon Cognito ID 공급자 예제

다음 코드 예제에서는 Amazon Cognito 자격 증명 공급자와 AWS SDK for Ruby 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시작하기

시작하기

Hello Amazon Cognito

다음 코드 예시에서는 Amazon Cognito 사용을 시작하는 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-cognitoidentityprovider'
require 'logger'

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

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

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

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

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListUserPools를 참조하세요.

SDK for Ruby를 사용한 Amazon Comprehend 예제

다음 코드 예제에서는 AWS SDK for Ruby Amazon Comprehend에서를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시나리오

고객 피드백 분석을 위한 애플리케이션 생성

다음 코드 예제에서는 고객 의견 카드를 분석하고, 원어에서 번역하고, 감정을 파악하고, 번역된 텍스트에서 오디오 파일을 생성하는 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Ruby

이 예제 애플리케이션은 고객 피드백 카드를 분석하고 저장합니다. 특히 뉴욕시에 있는 가상 호텔의 필요를 충족합니다. 호텔은 다양한 언어의 고객들로부터 물리적인 의견 카드의 형태로 피드백을 받습니다. 피드백은 웹 클라이언트를 통해 앱에 업로드됩니다. 의견 카드의 이미지가 업로드된 후 다음 단계가 수행됩니다.

- Amazon Textract를 사용하여 이미지에서 텍스트가 추출됩니다.
- Amazon Comprehend가 추출된 텍스트와 해당 언어의 감정을 파악합니다.
- 추출된 텍스트는 Amazon Translate를 사용하여 영어로 번역됩니다.
- Amazon Polly가 추출된 텍스트에서 오디오 파일을 합성합니다.

전체 앱은 AWS CDK를 사용하여 배포할 수 있습니다. 소스 코드와 배포 지침은 GitHub의 프로젝트를 참조하세요.

이 예제에서 사용되는 서비스

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

SDK for Ruby를 사용한 Amazon DocumentDB 예제

다음 코드 예제에서는 Amazon DocumentDB와 AWS SDK for Ruby 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

서버리스 예제

Amazon DocumentDB 트리거에서 간접적으로 Lambda 함수 호출

다음 코드 예제에서는 DocumentDB 변경 스트림에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 DocumentDB 페이로드를 검색하고 레코드 콘텐츠를 로깅합니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Ruby를 사용하여 Lambda로 Amazon DocumentDB 이벤트 소비

```
require 'json'

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

def log_document_db_event(record)
  event_data = record['event'] || {}
```

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

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

SDK for Ruby를 사용한 DynamoDB 예제

다음 코드 예제에서는 DynamoDB와 AWS SDK for Ruby 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시작하기

Hello DynamoDB

다음 코드 예제는 DynamoDB를 사용하여 시작하는 방법을 보여 줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

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

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

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

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

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

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListTables를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 영화 데이터를 저장할 수 있는 테이블을 생성합니다.
- 테이블에 하나의 영화를 추가하고 가져오고 업데이트합니다.
- 샘플 JSON 파일에서 테이블에 영화 데이터를 씁니다.
- 특정 연도에 개봉된 영화를 쿼리합니다.
- 특정 연도 범위 동안 개봉된 영화를 스캔합니다.
- 테이블에서 영화를 삭제한 다음, 테이블을 삭제합니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

DynamoDB 테이블을 캡슐화하는 클래스를 생성합니다.

```
# Creates an Amazon DynamoDB table that can be used to store movie data.
# The table uses the release year of the movie as the partition key and the
# title as the sort key.
#
# @param table_name [String] The name of the table to create.
```

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

헬퍼 함수를 생성하여 샘플 JSON 파일을 다운로드하고 추출합니다.

```
  # 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
```

대화식 시나리오를 실행하여 테이블을 생성하고 테이블에 대한 작업을 수행합니다.

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

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

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

  new_step(3, 'Update a record in the DynamoDB table.')
  my_movie[:rating] = CLI::UI::Prompt.ask("Let's update the movie you added with a
new rating, e.g. 3:").to_i
  response = dynamodb_wrapper.update_item(my_movie)
```

```
  puts("Updated '#{my_movie[:title]}' with new attributes:")
  puts JSON.pretty_generate(response).green
  print "Done!\n".green

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

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

  new_step(5, 'Query for a batch of items by key.')
  loop do
    release_year = CLI::UI::Prompt.ask('Enter a year between 1972 and 2018, e.g.
1999:').to_i
    results = dynamodb_wrapper.query_items(release_year)
    if results.any?
      puts("There were #{results.length} movies released in #{release_year}:")
      results.each do |movie|
        print "\t #{movie['title']}".green
      end
      break
    else
      continue = CLI::UI::Prompt.ask("Found no movies released in #{release_year}!
Try another year? (y/n)")
      break unless continue.eql?('y')
    end
  end
  print "\nDone!\n".green

  new_step(6, 'Scan for a batch of items using a filter expression.')
  years = {}
  years[:start] = CLI::UI::Prompt.ask('Enter a starting year between 1972 and
2018:')
  years[:end] = CLI::UI::Prompt.ask('Enter an ending year between 1972 and 2018:')
  releases = dynamodb_wrapper.scan_items(years)
```

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

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

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 다음 주제를 참조하세요.
  - BatchWriteItem
  - CreateTable

- DeleteItem
- DeleteTable
- DescribeTable
- GetItem
- PutItem
- Query
- Scan
- UpdateItem

작업

`BatchExecuteStatement`

다음 코드 예시는 `BatchExecuteStatement`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

PartiQL을 사용하여 항목 배치를 읽습니다.

```
class DynamoDBPartiQLBatch
  attr_reader :dynamo_resource, :table

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

  # Selects a batch of items from a table using PartiQL
  #
  # @param batch_titles [Array] Collection of movie titles
  # @return [Aws::DynamoDB::Types::BatchExecuteStatementOutput]
```

```
  def batch_execute_select(batch_titles)
    request_items = batch_titles.map do |title, year|
      {
        statement: "SELECT * FROM \"#{@table.name}\" WHERE title=? and year=?",
        parameters: [title, year]
      }
    end
    @dynamodb.client.batch_execute_statement({ statements: request_items })
  end
```

PartiQL을 사용하여 항목 배치를 삭제합니다.

```
class DynamoDBPartiQLBatch
  attr_reader :dynamo_resource, :table

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

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 BatchExecuteStatement를 참조하세요.

`BatchWriteItem`

다음 코드 예시는 `BatchWriteItem`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

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

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

```
    raise
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 BatchWriteItem을 참조하세요.

`CreateTable`

다음 코드 예시는 `CreateTable`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

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

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

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 CreateTable을 참조하세요.

`DeleteItem`

다음 코드 예시는 `DeleteItem`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

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

  # Deletes a movie from the table.
  #
  # @param title [String] The title of the movie to delete.
```

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteItem을 참조하세요.

`DeleteTable`

다음 코드 예시는 `DeleteTable`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

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

  # Deletes the table.
  def delete_table
    @table.delete
    @table = nil
  rescue Aws::DynamoDB::Errors::ServiceError => e
```

```
    puts("Couldn't delete table. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteTable을 참조하세요.

`DescribeTable`

다음 코드 예시는 `DescribeTable`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

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

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

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 DescribeTable을 참조하세요.

`ExecuteStatement`

다음 코드 예시는 `ExecuteStatement`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

PartiQL을 사용하여 항목을 한 개 선택합니다.

```
class DynamoDBPartiQLSingle
  attr_reader :dynamo_resource, :table

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

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

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

PartiQL을 사용하여 항목을 한 개 업데이트합니다.

```
class DynamoDBPartiQLSingle
  attr_reader :dynamo_resource, :table

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

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

PartiQL을 사용하여 항목을 한 개 추가합니다.

```
class DynamoDBPartiQLSingle
  attr_reader :dynamo_resource, :table

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

```
  end

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

PartiQL을 사용하여 항목을 한 개 삭제합니다.

```
class DynamoDBPartiQLSingle
  attr_reader :dynamo_resource, :table

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

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 ExecuteStatement를 참조하세요.

`GetItem`

다음 코드 예시는 `GetItem`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

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

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 GetItem을 참조하세요.

`ListTables`

다음 코드 예시는 `ListTables`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

테이블이 존재하는지 확인합니다.

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

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

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListTables를 참조하세요.

`PutItem`

다음 코드 예시는 `PutItem`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

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

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 PutItem을 참조하세요.

`Query`

다음 코드 예시는 `Query`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

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

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 Query를 참조하세요.

Scan

다음 코드 예시는 Scan의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

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

  # Scans for movies that were released in a range of years.
  # Uses a projection expression to return a subset of data for each movie.
  #
  # @param year_range [Hash] The range of years to retrieve.
  # @return [Array] The list of movies released in the specified years.
  def scan_items(year_range)
    movies = []
    scan_hash = {
      filter_expression: '#yr between :start_yr and :end_yr',
      projection_expression: '#yr, title, info.rating',
      expression_attribute_names: { '#yr' => 'year' },
      expression_attribute_values: {
        ':start_yr' => year_range[:start], ':end_yr' => year_range[:end]
      }
    }
    done = false
    start_key = nil
```

```
    until done
      scan_hash[:exclusive_start_key] = start_key unless start_key.nil?
      response = @table.scan(scan_hash)
      movies.concat(response.items) unless response.items.empty?
      start_key = response.last_evaluated_key
      done = start_key.nil?
    end
  rescue Aws::DynamoDB::Errors::ServiceError => e
    puts("Couldn't scan for movies. Here's why:")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    movies
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 Scan을 참조하세요.

`UpdateItem`

다음 코드 예시는 `UpdateItem`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class DynamoDBBasics
  attr_reader :dynamo_resource, :table

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

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

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 UpdateItem을 참조하세요.

시나리오

PartiQL 문 배치를 사용하여 테이블 쿼리

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 여러 SELECT 문을 실행하여 항목 배치를 가져옵니다.
- 여러 INSERT 문을 실행하여 항목 배치를 추가합니다.
- 여러 UPDATE 문을 실행하여 항목 배치를 업데이트합니다.
- 여러 DELETE 문을 실행하여 항목 배치를 삭제합니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

테이블을 생성하고 배치 PartiQL 쿼리를 실행하는 시나리오를 실행합니다.

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

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

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

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

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

  new_step(5, 'Delete the table.')
  return unless scaffold.exists?(table_name)

  scaffold.delete_table
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 BatchExecuteStatement를 참조하세요.

PartiQL을 사용하여 테이블 쿼리

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- SELECT 문을 실행하여 항목을 가져옵니다.
- INSERT 문을 실행하여 항목을 추가합니다.
- UPDATE 문을 실행하여 항목을 업데이트합니다.
- DELETE 문을 실행하여 항목을 삭제합니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

테이블을 생성하고 PartiQL 쿼리를 실행하는 시나리오를 실행합니다.

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

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

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

new_step(3, 'Select a single item from the movies table.')
```

```
  response = sdk.select_item_by_title('Star Wars')
  puts("Items selected for title 'Star Wars': #{response.items.length}\n")
  print response.items.first.to_s.yellow
  print "\n\nDone!\n".green

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

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

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

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

  scaffold.delete_table
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ExecuteStatement를 참조하세요.

서버리스 예제

DynamoDB 트리거에서 간접적으로 Lambda 함수 간접 호출

다음 코드 예제에서는 DynamoDB 스트림에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 DynamoDB 페이로드를 검색하고 레코드 콘텐츠를 로깅합니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Ruby를 사용하여 Lambda로 DynamoDB 이벤트 사용.

```
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
```

DynamoDB 트리거로 Lambda 함수에 대한 배치 항목 실패 보고

다음 코드 예제에서는 DynamoDB 스트림에서 이벤트를 수신하는 Lambda 함수에 대한 부분 배치 응답을 구현하는 방법을 보여줍니다. 이 함수는 응답으로 배치 항목 실패를 보고하고 나중에 해당 메시지를 다시 시도하도록 Lambda에 신호를 보냅니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Ruby를 사용하여 Lambda로 DynamoDB 배치 항목 실패 보고.

```
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
```

SDK for Ruby를 사용한 Amazon EC2 예제

다음 코드 예제에서는 Amazon EC2와 AWS SDK for Ruby 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시작하기
- 작업

시작하기

Hello Amazon EC2

다음 코드 예제에서는 Amazon EC2 사용을 시작하는 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

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

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

    instances = fetch_instances

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

  private

  # Fetches all EC2 instances using pagination.
```

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

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

    instances
  end

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

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 DescribeSecurityGroups를 참조하세요.

작업

`AllocateAddress`

다음 코드 예시는 `AllocateAddress`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 AllocateAddress를 참조하세요.

`AssociateAddress`

다음 코드 예시는 `AssociateAddress`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 AssociateAddress를 참조하세요.

`CreateKeyPair`

다음 코드 예시는 `CreateKeyPair`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

require 'aws-sdk-ec2'

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

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

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

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

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

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

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

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

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

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

  puts '-' * 10
```

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 CreateKeyPair를 참조하세요.

`CreateRouteTable`

다음 코드 예시는 `CreateRouteTable`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

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

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

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

```
    destination_cidr_block = '0.0.0.0/0'
    tag_key = 'my-key'
    tag_value = 'my-value'
    # Replace us-west-2 with the AWS Region you're using for Amazon EC2.
    region = 'us-west-2'
  # Otherwise, use the values as specified at the command prompt.
  else
    vpc_id = ARGV[0]
    subnet_id = ARGV[1]
    gateway_id = ARGV[2]
    destination_cidr_block = ARGV[3]
    tag_key = ARGV[4]
    tag_value = ARGV[5]
    region = ARGV[6]
  end

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

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

run_me if $PROGRAM_NAME == __FILE__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 CreateRouteTable을 참조하세요.

`CreateSecurityGroup`

다음 코드 예시는 `CreateSecurityGroup`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

require 'aws-sdk-ec2'

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

  return if sg.ip_permissions_egress.empty?

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

```
  end
end

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

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

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

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

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

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

```
    ARGV
  end
end

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

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

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

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

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

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 CreateSecurityGroup을 참조하세요.

`CreateSubnet`

다음 코드 예시는 `CreateSubnet`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-ec2'

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

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

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

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

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

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

```
  end
end

run_me if $PROGRAM_NAME == __FILE__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 CreateSubnet을 참조하세요.

`CreateVpc`

다음 코드 예시는 `CreateVpc`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-ec2'

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

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

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

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

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

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

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 CreateVpc를 참조하세요.

`DescribeInstances`

다음 코드 예시는 `DescribeInstances`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-ec2'

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

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

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

run_me if $PROGRAM_NAME == __FILE__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DescribeInstances 참조하세요.

`DescribeRegions`

다음 코드 예시는 `DescribeRegions`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DescribeRegions를 참조하세요.

`ReleaseAddress`

다음 코드 예시는 `ReleaseAddress`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 ReleaseAddress를 참조하세요.

`StartInstances`

다음 코드 예시는 `StartInstances`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-ec2'

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

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

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

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

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

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

  puts 'Could not start instance.'
end

run_me if $PROGRAM_NAME == __FILE__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 StartInstances를 참조하세요.

`StopInstances`

다음 코드 예시는 `StopInstances`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-ec2'

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

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

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

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

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

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

  puts 'Could not stop instance.'
end

run_me if $PROGRAM_NAME == __FILE__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 StopInstances를 참조하세요.

`TerminateInstances`

다음 코드 예시는 `TerminateInstances`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-ec2'

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

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

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

  ec2_client.terminate_instances(instance_ids: [instance_id])
  ec2_client.wait_until(:instance_terminated, instance_ids: [instance_id])
  puts 'Instance terminated.'
```

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

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

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

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

  puts 'Could not terminate instance.'
end

run_me if $PROGRAM_NAME == __FILE__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 TerminateInstances를 참조하세요.

SDK for Ruby를 사용한 Elastic Beanstalk 예제

다음 코드 예제에서는 Elastic Beanstalk와 AWS SDK for Ruby 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`DescribeApplications`

다음 코드 예시는 `DescribeApplications`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

  # Lists applications and their environments
  def list_applications
    @eb_client.describe_applications.applications.each do |application|
      log_application_details(application)
```

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

  private

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

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 DescribeApplications을 참조하세요.

`ListAvailableSolutionStacks`

다음 코드 예시는 `ListAvailableSolutionStacks`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListAvailableSolutionStacks을 참조하세요.

`UpdateApplication`

다음 코드 예시는 `UpdateApplication`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 UpdateApplication을 참조하세요.

SDK for Ruby를 사용한 EventBridge 예제

다음 코드 예제에서는 EventBridge와 AWS SDK for Ruby 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시나리오

시나리오

규칙 생성 및 트리거

다음 코드 예제는 Amazon EventBridge에서 규칙을 생성하고 트리거하는 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

함수를 올바른 순서로 직접적으로 호출합니다.

```
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'
```

지정된 Amazon Simple Notification Service(SNS) 주제가 이 함수에 제공된 주제 중에 있는지 확인합니다.

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

```
#   )
#     puts 'Topic found.'
#   end
def topic_found?(topics, topic_arn)
  topics.each do |topic|
    return true if topic.topic_arn == topic_arn
  end
  false
end
```

Amazon SNS에서 호출자가 사용할 수 있는 주제 중에 지정된 주제가 있는지 확인합니다.

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

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

```
  false
rescue StandardError => e
  puts "Topic not found: #{e.message}"
  false
end
```

Amazon SNS에서 주제를 생성한 다음, 이메일 주소를 구독하여 해당 주제에 대한 알림을 받습니다.

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

이 함수에 제공된 역할 중에서 지정된 AWS Identity and Access Management (IAM) 역할이 존재하는지 확인합니다.

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

IAM에서 호출자가 사용할 수 있는 역할 중에 지정된 역할이 있는지 확인합니다.

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

```
  if response.roles.count.positive?
    if role_found?(response.roles, role_arn)
      puts 'Role found.'
      return true
    end
    while response.next_page?
      response = response.next_page
      next unless response.roles.count.positive?

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

IAM에서 역할을 생성합니다.

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

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

```
end
```

이 함수에 제공된 역할 중에 지정된 EventBridge 역할이 있는지 확인합니다.

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

EventBridge에서 호출자가 사용할 수 있는 규칙 중에 지정된 규칙이 있는지 확인합니다.

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

```
  if response.rules.count.positive?
    if rule_found?(response.rules, rule_name)
      puts 'Rule found.'
      return true
    end
    while response.next_page?
      response = response.next_page
      next unless response.rules.count.positive?

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

EventBridge에서 규칙을 생성합니다.

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

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

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

Amazon CloudWatch Logs에서 호출자가 사용할 수 있는 로그 그룹 중에 지정된 로그 그룹이 있는지 확인합니다.

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

```
def log_group_exists?(cloudwatchlogs_client, log_group_name)
  puts "Searching for log group with name '#{log_group_name}'..."
  response = cloudwatchlogs_client.describe_log_groups(
    log_group_name_prefix: log_group_name
  )
  if response.log_groups.count.positive?
    response.log_groups.each do |log_group|
      if log_group.log_group_name == log_group_name
        puts 'Log group found.'
        return true
      end
    end
  end
  puts 'Log group not found.'
  false
rescue StandardError => e
  puts "Log group not found: #{e.message}"
  false
end
```

CloudWatch Logs의 로그 그룹을 생성합니다.

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

CloudWatch Logs의 로그 스트림에 이벤트를 기록합니다.

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

```
    log_events: [
      {
        timestamp: (Time.now.utc.to_f.round(3) * 1_000).to_i,
        message: message
      }
    ]
  }
  event[:sequence_token] = sequence_token unless sequence_token.empty?

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

Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스를 다시 시작하고 관련 활동에 대한 정보를 CloudWatch Logs의 로그 스트림에 추가합니다.

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

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

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

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

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

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

EventBridge의 규칙 활동에 대한 정보를 표시합니다.

```
# 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
```

CloudWatch Logs 로그 그룹의 모든 로그 스트림에 대한 로그 정보를 표시합니다.

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

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

호출자에게 더 이상 필요하지 않은 연결된 AWS 리소스를 수동으로 정리하라는 알림을 표시합니다.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 다음 주제를 참조하세요.
  - PutEvents
  - PutRule

AWS Glue SDK for Ruby를 사용한 예제

다음 코드 예제에서는를와 AWS SDK for Ruby 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS Glue.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시작하기

안녕하세요 AWS Glue

다음 코드 예제에서는 AWS Glue를 사용하여 시작하는 방법을 보여 줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-glue'
require 'logger'

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

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

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

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

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

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

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListJobs를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 퍼블릭 Amazon S3 버킷을 크롤링하고 CSV 형식의 메타데이터 데이터베이스를 생성하는 크롤러를 생성합니다.
- 의 데이터베이스 및 테이블에 대한 정보를 나열합니다 AWS Glue Data Catalog.
- 작업을 생성하여 S3 버킷에서 CSV 데이터를 추출하고, 데이터를 변환하며, JSON 형식의 출력을 다른 S3 버킷으로 로드합니다.
- 작업 실행에 대한 정보를 나열하고 변환된 데이터를 확인하며 리소스를 정리합니다.

자세한 내용은 자습서: AWS Glue Studio 시작하기를 참조하세요.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

시나리오에 사용되는 AWS Glue 함수를 래핑하는 클래스를 생성합니다.

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

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

  # Creates a new crawler with the specified configuration.
  #
  # @param name [String] The name of the crawler.
```

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

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

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

```
    raise
  end

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

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

  # Creates a new job with the specified configuration.
  #
  # @param name [String] The name of the job.
  # @param description [String] The description of the job.
  # @param role_arn [String] The ARN of the IAM role to be used by the job.
  # @param script_location [String] The location of the ETL script for the job.
  # @return [void]
  def create_job(name, description, role_arn, script_location)
    @glue_client.create_job(
      name: name,
      description: description,
      role: role_arn,
      command: {
        name: 'glueetl',
        script_location: script_location,
        python_version: '3'
```

```
    },
    glue_version: '3.0'
  )
rescue Aws::Glue::Errors::GlueException => e
  @logger.error("Glue could not create job #{name}: \n#{e.message}")
  raise
end

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

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

# Retrieves a list of job runs for the specified job.
#
# @param job_name [String] The name of the job to retrieve job runs for.
# @return [Array<Aws::Glue::Types::JobRun>]
```

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

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

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

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

  # Removes a specified database from a Data Catalog.
  #
  # @param database_name [String] The name of the database to delete.
  # @return [void]
  def delete_database(database_name)
```

```
    @glue_client.delete_database(name: database_name)
  rescue Aws::Glue::Errors::ServiceError => e
    @logger.error("Glue could not delete database: \n#{e.message}")
  end

  # Uploads a job script file to an S3 bucket.
  #
  # @param file_path [String] The local path of the job script file.
  # @param bucket_resource [Aws::S3::Bucket] The S3 bucket resource to upload the
 file to.
  # @return [void]
  def upload_job_script(file_path, bucket_resource)
    File.open(file_path) do |file|
      bucket_resource.client.put_object({
                                          body: file,
                                          bucket: bucket_resource.name,
                                          key: file_path
                                        })
    end
  rescue Aws::S3::Errors::S3UploadFailedError => e
    @logger.error("S3 could not upload job script: \n#{e.message}")
    raise
  end
end
```

시나리오를 실행하는 클래스를 생성합니다.

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

  def run(crawler_name, db_name, db_prefix, data_source, job_script, job_name)
    wrapper = GlueWrapper.new(@glue_client, @logger)
    setup_crawler(wrapper, crawler_name, db_name, db_prefix, data_source)
    query_database(wrapper, crawler_name, db_name)
    create_and_run_job(wrapper, job_script, job_name, db_name)
  end
```

```
  private

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

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

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

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

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

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

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

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

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

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

  puts 'Demo complete.'
end
```

에서 작업 실행 중에 데이터를 추출, 변환 및 로드하는 AWS Glue 데 사용하는 ETL 스크립트를 생성합니다.

```
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()
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 다음 주제를 참조하십시오.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables

- ListJobs
- StartCrawler
- StartJobRun

작업

`CreateCrawler`

다음 코드 예시는 `CreateCrawler`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 CreateCrawler를 참조하세요.

`CreateJob`

다음 코드 예시는 `CreateJob`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

```
def initialize(glue_client, logger)
  @glue_client = glue_client
  @logger = logger
end

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 CreateJob을 참조하세요.

`DeleteCrawler`

다음 코드 예시는 `DeleteCrawler`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteCrawler를 참조하세요.

`DeleteDatabase`

다음 코드 예시는 `DeleteDatabase`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteDatabase를 참조하세요.

`DeleteJob`

다음 코드 예시는 `DeleteJob`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteJob을 참조하세요.

`DeleteTable`

다음 코드 예시는 `DeleteTable`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteTable을 참조하세요.

`GetCrawler`

다음 코드 예시는 `GetCrawler`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 GetCrawler를 참조하세요.

`GetDatabase`

다음 코드 예시는 `GetDatabase`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 GetDatabase를 참조하세요.

`GetJobRun`

다음 코드 예시는 `GetJobRun`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 GetJobRun을 참조하세요.

`GetJobRuns`

다음 코드 예시는 `GetJobRuns`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 GetJobRuns를 참조하세요.

`GetTables`

다음 코드 예시는 `GetTables`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 GetTables를 참조하세요.

`ListJobs`

다음 코드 예시는 `ListJobs`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListJobs를 참조하세요.

`StartCrawler`

다음 코드 예시는 `StartCrawler`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 StartCrawler를 참조하세요.

`StartJobRun`

다음 코드 예시는 `StartJobRun`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 StartJobRun을 참조하세요.

SDK for Ruby를 사용한 IAM 예제

다음 코드 예제에서는 IAM과 AWS SDK for Ruby 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 시작하기
- 기본 사항
- 작업

시작하기

Hello IAM

다음 코드 예제에서는 IAM 사용을 시작하는 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

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

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

    paginator = @client.list_policies
    policies = []

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

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

if $PROGRAM_NAME == __FILE__
```

```
  iam_client = Aws::IAM::Client.new
  manager = IAMManager.new(iam_client)
  manager.list_policies
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListPolicies를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 사용자를 생성하고 역할을 수임하는 방법을 보여줍니다.

> **Warning**
>
> 보안 위험을 방지하려면 목적별 소프트웨어를 개발하거나 실제 데이터로 작업할 때 IAM 사용자를 인증에 사용하지 마세요. 대신 AWS IAM Identity Center과 같은 보안 인증 공급자를 통한 페더레이션을 사용하십시오.

- 권한이 없는 사용자를 생성합니다.
- 계정에 대한 Amazon S3 버킷을 나열할 수 있는 권한을 부여하는 역할을 생성합니다.
- 사용자가 역할을 수임할 수 있도록 정책을 추가합니다.
- 역할을 수임하고 임시 자격 증명 정보를 사용하여 S3 버킷을 나열한 후 리소스를 정리합니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon S3 버킷을 나열할 수 있는 권한을 부여하는 역할과 IAM 사용자를 생성합니다. 사용자는 역할을 수임할 수 있는 권한만 있습니다. 역할을 수임한 후 임시 자격 증명을 사용하여 계정의 버킷을 나열합니다.

```
# Wraps the scenario actions.
class ScenarioCreateUserAssumeRole
  attr_reader :iam_client

  # @param [Aws::IAM::Client] iam_client: The AWS IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Waits for the specified number of seconds.
  #
  # @param duration [Integer] The number of seconds to wait.
  def wait(duration)
    puts('Give AWS time to propagate resources...')
    sleep(duration)
  end

  # Creates a user.
  #
  # @param user_name [String] The name to give the user.
  # @return [Aws::IAM::User] The newly created user.
  def create_user(user_name)
    user = @iam_client.create_user(user_name: user_name).user
    @logger.info("Created demo user named #{user.user_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info('Tried and failed to create demo user.')
    @logger.info("\t#{e.code}: #{e.message}")
    @logger.info("\nCan't continue the demo without a user!")
    raise
  else
    user
  end

  # Creates an access key for a user.
  #
  # @param user [Aws::IAM::User] The user that owns the key.
  # @return [Aws::IAM::AccessKeyPair] The newly created access key.
  def create_access_key_pair(user)
    user_key = @iam_client.create_access_key(user_name: user.user_name).access_key
    @logger.info("Created accesskey pair for user #{user.user_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create access keys for user #{user.user_name}.")
```

```
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  else
    user_key
  end

  # Creates a role that can be assumed by a user.
  #
  # @param role_name [String] The name to give the role.
  # @param user [Aws::IAM::User] The user who is granted permission to assume the
role.
  # @return [Aws::IAM::Role] The newly created role.
  def create_role(role_name, user)
    trust_policy = {
      Version: '2012-10-17',
      Statement: [{
        Effect: 'Allow',
        Principal: { 'AWS': user.arn },
        Action: 'sts:AssumeRole'
      }]
    }.to_json
    role = @iam_client.create_role(
      role_name: role_name,
      assume_role_policy_document: trust_policy
    ).role
    @logger.info("Created role #{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create a role for the demo. Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  else
    role
  end

  # Creates a policy that grants permission to list S3 buckets in the account, and
  # then attaches the policy to a role.
  #
  # @param policy_name [String] The name to give the policy.
  # @param role [Aws::IAM::Role] The role that the policy is attached to.
  # @return [Aws::IAM::Policy] The newly created policy.
  def create_and_attach_role_policy(policy_name, role)
    policy_document = {
      Version: '2012-10-17',
      Statement: [{
```

```
        Effect: 'Allow',
        Action: 's3:ListAllMyBuckets',
        Resource: 'arn:aws:s3:::*'
      }]
    }.to_json
    policy = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document
    ).policy
    @iam_client.attach_role_policy(
      role_name: role.role_name,
      policy_arn: policy.arn
    )
    @logger.info("Created policy #{policy.policy_name} and attached it to role
#{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create a policy and attach it to role #{role.role_name}.
Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

  # Creates an inline policy for a user that lets the user assume a role.
  #
  # @param policy_name [String] The name to give the policy.
  # @param user [Aws::IAM::User] The user that owns the policy.
  # @param role [Aws::IAM::Role] The role that can be assumed.
  # @return [Aws::IAM::UserPolicy] The newly created policy.
  def create_user_policy(policy_name, user, role)
    policy_document = {
      Version: '2012-10-17',
      Statement: [{
        Effect: 'Allow',
        Action: 'sts:AssumeRole',
        Resource: role.arn
      }]
    }.to_json
    @iam_client.put_user_policy(
      user_name: user.user_name,
      policy_name: policy_name,
      policy_document: policy_document
    )
    puts("Created an inline policy for #{user.user_name} that lets the user assume
role #{role.role_name}.")
```

```
 rescue Aws::Errors::ServiceError => e
   @logger.info("Couldn't create an inline policy for user #{user.user_name}.
Here's why: ")
   @logger.info("\t#{e.code}: #{e.message}")
   raise
 end

 # Creates an Amazon S3 resource with specified credentials. This is separated into
a
 # factory function so that it can be mocked for unit testing.
 #
 # @param credentials [Aws::Credentials] The credentials used by the Amazon S3
resource.
 def create_s3_resource(credentials)
   Aws::S3::Resource.new(client: Aws::S3::Client.new(credentials: credentials))
 end

 # Lists the S3 buckets for the account, using the specified Amazon S3 resource.
 # Because the resource uses credentials with limited access, it may not be able to
 # list the S3 buckets.
 #
 # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
 def list_buckets(s3_resource)
   count = 10
   s3_resource.buckets.each do |bucket|
     @logger.info "\t#{bucket.name}"
     count -= 1
     break if count.zero?
   end
 rescue Aws::Errors::ServiceError => e
   if e.code == 'AccessDenied'
     puts('Attempt to list buckets with no permissions: AccessDenied.')
   else
     @logger.info("Couldn't list buckets for the account. Here's why: ")
     @logger.info("\t#{e.code}: #{e.message}")
     raise
   end
 end

 # Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
 # This is separated into a factory function so that it can be mocked for unit
testing.
 #
```

```
 # @param key_id [String] The ID of the access key used by the STS client.
 # @param key_secret [String] The secret part of the access key used by the STS
client.
 def create_sts_client(key_id, key_secret)
   Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
 end

 # Gets temporary credentials that can be used to assume a role.
 #
 # @param role_arn [String] The ARN of the role that is assumed when these
credentials
 #                          are used.
 # @param sts_client [AWS::STS::Client] An AWS STS client.
 # @return [Aws::AssumeRoleCredentials] The credentials that can be used to assume
the role.
 def assume_role(role_arn, sts_client)
   credentials = Aws::AssumeRoleCredentials.new(
     client: sts_client,
     role_arn: role_arn,
     role_session_name: 'create-use-assume-role-scenario'
   )
   @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
   credentials
 end

 # Deletes a role. If the role has policies attached, they are detached and
 # deleted before the role is deleted.
 #
 # @param role_name [String] The name of the role to delete.
 def delete_role(role_name)
   @iam_client.list_attached_role_policies(role_name:
role_name).attached_policies.each do |policy|
     @iam_client.detach_role_policy(role_name: role_name, policy_arn:
policy.policy_arn)
     @iam_client.delete_policy(policy_arn: policy.policy_arn)
     @logger.info("Detached and deleted policy #{policy.policy_name}.")
   end
   @iam_client.delete_role({ role_name: role_name })
   @logger.info("Role deleted: #{role_name}.")
 rescue Aws::Errors::ServiceError => e
   @logger.info("Couldn't detach policies and delete role #{role.name}. Here's
why:")
   @logger.info("\t#{e.code}: #{e.message}")
   raise
```

```
  end

  # Deletes a user. If the user has inline policies or access keys, they are deleted
  # before the user is deleted.
  #
  # @param user [Aws::IAM::User] The user to delete.
  def delete_user(user_name)
    user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
    user.each do |key|
      @iam_client.delete_access_key({ access_key_id: key.access_key_id, user_name:
 user_name })
      @logger.info("Deleted access key #{key.access_key_id} for user
 '#{user_name}'.")
    end

    @iam_client.delete_user(user_name: user_name)
    @logger.info("Deleted user '#{user_name}'.")
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting user '#{user_name}': #{e.message}")
  end
end

# Runs the IAM create a user and assume a role scenario.
def run_scenario(scenario)
  puts('-' * 88)
  puts('Welcome to the IAM create a user and assume a role demo!')
  puts('-' * 88)
  user = scenario.create_user("doc-example-user-#{Random.uuid}")
  user_key = scenario.create_access_key_pair(user)
  scenario.wait(10)
  role = scenario.create_role("doc-example-role-#{Random.uuid}", user)
  scenario.create_and_attach_role_policy("doc-example-role-policy-#{Random.uuid}",
 role)
  scenario.create_user_policy("doc-example-user-policy-#{Random.uuid}", user, role)
  scenario.wait(10)
  puts('Try to list buckets with credentials for a user who has no permissions.')
  puts('Expect AccessDenied from this call.')
  scenario.list_buckets(
    scenario.create_s3_resource(Aws::Credentials.new(user_key.access_key_id,
 user_key.secret_access_key))
  )
  puts('Now, assume the role that grants permission.')
  temp_credentials = scenario.assume_role(
```

```
    role.arn, scenario.create_sts_client(user_key.access_key_id,
 user_key.secret_access_key)
  )
  puts('Here are your buckets:')
  scenario.list_buckets(scenario.create_s3_resource(temp_credentials))
  puts("Deleting role '#{role.role_name}' and attached policies.")
  scenario.delete_role(role.role_name)
  puts("Deleting user '#{user.user_name}', policies, and keys.")
  scenario.delete_user(user.user_name)
  puts('Thanks for watching!')
  puts('-' * 88)
rescue Aws::Errors::ServiceError => e
  puts('Something went wrong with the demo.')
  puts("\t#{e.code}: #{e.message}")
end

run_scenario(ScenarioCreateUserAssumeRole.new(Aws::IAM::Client.new)) if
 $PROGRAM_NAME == __FILE__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 다음 주제를 참조하십시오.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

작업

`AttachRolePolicy`

다음 코드 예시는 `AttachRolePolicy`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예시 모듈은 역할 정책을 나열, 생성, 연결 및 분리합니다.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 AttachRolePolicy를 참조하세요.

`AttachUserPolicy`

다음 코드 예시는 `AttachUserPolicy`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
  # 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 AttachUserPolicy를 참조하세요.

`CreateAccessKey`

다음 코드 예시는 `CreateAccessKey`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예시 모듈은 액세스 키를 나열, 생성, 비활성화 및 삭제합니다.

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'AccessKeyManager'
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
```

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

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

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

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 CreateAccessKey를 참조하세요.

`CreateAccountAlias`

다음 코드 예시는 `CreateAccountAlias`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

계정 별칭을 나열하고, 생성하고, 삭제합니다.

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
```

```
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info('Account aliases are:')
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info('No account aliases found.')
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

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

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 CreateAccountAlias를 참조하세요.

`CreatePolicy`

다음 코드 예시는 `CreatePolicy`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예시 모듈은 역할 정책을 나열, 생성, 연결 및 분리합니다.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 CreatePolicy를 참조하세요.

`CreateRole`

다음 코드 예시는 `CreateRole`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
  # 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 CreateRole을 참조하세요.

`CreateServiceLinkedRole`

다음 코드 예시는 `CreateServiceLinkedRole`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 CreateServiceLinkedRole을 참조하세요.

`CreateUser`

다음 코드 예시는 `CreateUser`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 CreateUser를 참조하세요.

`DeleteAccessKey`

다음 코드 예시는 `DeleteAccessKey`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예시 모듈은 액세스 키를 나열, 생성, 비활성화 및 삭제합니다.

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'AccessKeyManager'
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
```

```
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

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

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

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteAccessKey를 참조하세요.

`DeleteAccountAlias`

다음 코드 예시는 `DeleteAccountAlias`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

계정 별칭을 나열하고, 생성하고, 삭제합니다.

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
```

```
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info('Account aliases are:')
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info('No account aliases found.')
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

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

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteAccountAlias를 참조하세요.

`DeleteRole`

다음 코드 예시는 `DeleteRole`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# Deletes a role and its attached policies.
#
# @param role_name [String] The name of the role to delete.
def delete_role(role_name)
  # Detach and delete attached policies
  @iam_client.list_attached_role_policies(role_name: role_name).each do |response|
    response.attached_policies.each do |policy|
      @iam_client.detach_role_policy({
                                       role_name: role_name,
                                       policy_arn: policy.policy_arn
                                     })
      # Check if the policy is a customer managed policy (not AWS managed)
      unless policy.policy_arn.include?('aws:policy/')
        @iam_client.delete_policy({ policy_arn: policy.policy_arn })
        @logger.info("Deleted customer managed policy #{policy.policy_name}.")
      end
    end
  end

  # Delete the role
  @iam_client.delete_role({ role_name: role_name })
  @logger.info("Deleted role #{role_name}.")
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Couldn't detach policies and delete role #{role_name}. Here's
why:")
  @logger.error("\t#{e.code}: #{e.message}")
  raise
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteRole을 참조하세요.

`DeleteServerCertificate`

다음 코드 예시는 `DeleteServerCertificate`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

서버 인증서를 나열하고, 업데이트하고, 삭제합니다.

```
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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteServerCertificate를 참조하세요.

`DeleteServiceLinkedRole`

다음 코드 예시는 `DeleteServiceLinkedRole`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

private

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

    sleep(3)
  end
end

# Handles deletion error
```

```
  #
  # @param e [Aws::Errors::ServiceError] The error encountered during deletion
  # @param role_name [String] The name of the role attempted to delete
  def handle_deletion_error(e, role_name)
    return if e.code == 'NoSuchEntity'

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteServiceLinkedRole을 참조하세요.

`DeleteUser`

다음 코드 예시는 `DeleteUser`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
  # 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteUser를 참조하세요.

`DeleteUserPolicy`

다음 코드 예시는 `DeleteUserPolicy`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
 # 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteUserPolicy를 참조하세요.

`DetachRolePolicy`

다음 코드 예시는 `DetachRolePolicy`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예시 모듈은 역할 정책을 나열, 생성, 연결 및 분리합니다.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DetachRolePolicy를 참조하세요.

`DetachUserPolicy`

다음 코드 예시는 `DetachUserPolicy`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
  # Detaches a policy from a user
  #
  # @param user_name [String] The name of the user
  # @param policy_arn [String] The ARN of the policy to detach
  # @return [Boolean] true if the policy was successfully detached, false otherwise
  def detach_user_policy(user_name, policy_arn)
    @iam_client.detach_user_policy(
      user_name: user_name,
      policy_arn: policy_arn
    )
    @logger.info("Policy '#{policy_arn}' detached from user '#{user_name}'
 successfully.")
    true
  rescue Aws::IAM::Errors::NoSuchEntity
```

```
    @logger.error('Error detaching policy: Policy or user does not exist.')
    false
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from user '#{user_name}': #{e.message}")
    false
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DetachUserPolicy를 참조하세요.

`GetAccountPasswordPolicy`

다음 코드 예시는 `GetAccountPasswordPolicy`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# Class to manage IAM account password policies
class PasswordPolicyManager
  attr_accessor :iam_client, :logger

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

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

```
    raise
  end
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 GetAccountPasswordPolicy를 참조하세요.

`GetPolicy`

다음 코드 예시는 `GetPolicy`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
 # 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 GetPolicy를 참조하세요.

GetRole

다음 코드 예시는 GetRole의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 GetRole을 참조하세요.

GetUser

다음 코드 예시는 GetUser의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
 # 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 GetUser을 참조하세요.

`ListAccessKeys`

다음 코드 예시는 `ListAccessKeys`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예시 모듈은 액세스 키를 나열, 생성, 비활성화 및 삭제합니다.

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'AccessKeyManager'
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

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

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

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListAccessKeys를 참조하세요.

`ListAccountAliases`

다음 코드 예시는 `ListAccountAliases`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

계정 별칭을 나열하고, 생성하고, 삭제합니다.

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info('Account aliases are:')
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info('No account aliases found.')
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

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

```
    false
  end

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListAccountAliases를 참조하세요.

`ListAttachedRolePolicies`

다음 코드 예시는 `ListAttachedRolePolicies`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예시 모듈은 역할 정책을 나열, 생성, 연결 및 분리합니다.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListAttachedRolePolicies를 참조하세요.

`ListGroups`

다음 코드 예시는 `ListGroups`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListGroups를 참조하세요.

`ListPolicies`

다음 코드 예시는 `ListPolicies`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예시 모듈은 역할 정책을 나열, 생성, 연결 및 분리합니다.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListPolicies를 참조하세요.

`ListRolePolicies`

다음 코드 예시는 `ListRolePolicies`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
  # 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListRolePolicies를 참조하세요.

`ListRoles`

다음 코드 예시는 `ListRoles`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListRoles를 참조하세요.

`ListSAMLProviders`

다음 코드 예시는 `ListSAMLProviders`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListSAMLProviders를 참조하세요.

`ListServerCertificates`

다음 코드 예시는 `ListServerCertificates`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

서버 인증서를 나열하고, 업데이트하고, 삭제합니다.

```
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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListServerCertificates를 참조하세요.

`ListUsers`

다음 코드 예시는 `ListUsers`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListUsers를 참조하세요.

`PutUserPolicy`

다음 코드 예시는 `PutUserPolicy`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 PutUserPolicy를 참조하세요.

`UpdateServerCertificate`

다음 코드 예시는 `UpdateServerCertificate`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

서버 인증서를 나열하고, 업데이트하고, 삭제합니다.

```
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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 UpdateServerCertificate를 참조하세요.

`UpdateUser`

다음 코드 예시는 `UpdateUser`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
  # 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 UpdateUser를 참조하세요.

SDK for Ruby를 사용한 Kinesis 예제

다음 코드 예제에서는 Kinesis와 AWS SDK for Ruby 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 서버리스 예제

서버리스 예제

Kinesis 트리거에서 간접적으로 Lambda 함수 간접 호출

다음 코드 예제에서는 Kinesis 스트림에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 Kinesis 페이로드를 검색하고, Base64에서 디코딩하고, 레코드 콘텐츠를 로깅합니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Ruby를 사용하여 Lambda로 Kinesis 이벤트를 사용합니다.

```
# 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
```

Kinesis 트리거로 Lambda 함수에 대한 배치 항목 실패 보고

다음 코드 예제는 Kinesis 스트림에서 이벤트를 수신하는 Lambda 함수에 대한 부분 배치 응답을 구현하는 방법을 보여줍니다. 이 함수는 응답으로 배치 항목 실패를 보고하고 나중에 해당 메시지를 다시 시도하도록 Lambda에 신호를 보냅니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Ruby를 사용하여 Lambda로 Kinesis 배치 항목 실패를 보고합니다.

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

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

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

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

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

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

AWS KMS SDK for Ruby를 사용한 예제

다음 코드 예제에서는를와 AWS SDK for Ruby 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS KMS.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`CreateKey`

다음 코드 예시는 `CreateKey`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
```

- API에 대한 세부 정보는 AWS SDK for Ruby API 참조의 CreateKey를 참조하세요.

`Decrypt`

다음 코드 예시는 `Decrypt`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
```

- API에 대한 세부 정보는 AWS SDK for Ruby API 참조의 Decrypt를 참조하세요.

`Encrypt`

다음 코드 예시는 `Encrypt`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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*')
```

- API에 대한 세부 정보는 AWS SDK for Ruby API 참조의 Encrypt를 참조하세요.

`ReEncrypt`

다음 코드 예시는 `ReEncrypt`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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*')
```

- 자세한 내용을 알아보려면AWS SDK for Ruby API 참조의 ReEncrypt를 참조하세요.

SDK for Ruby를 사용한 Lambda 예제

다음 코드 예제에서는 Lambda와 AWS SDK for Ruby 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시작하기

Hello Lambda

다음 코드 예제에서는 Lambda 사용을 시작하는 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-lambda'

# Creates an AWS Lambda client using the default credentials and configuration
def lambda_client
  Aws::Lambda::Client.new
```

```
end

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

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

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

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

list_lambda_functions if __FILE__ == $PROGRAM_NAME
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListFunctions를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- IAM 역할과 Lambda 함수를 생성하고 핸들러 코드를 업로드합니다.
- 단일 파라미터로 함수를 간접적으로 간접 호출하고 결과를 가져옵니다.
- 함수 코드를 업데이트하고 환경 변수로 구성합니다.
- 새 파라미터로 함수를 간접적으로 간접 호출하고 결과를 가져옵니다. 반환된 실행 로그를 표시합니다.
- 계정의 함수를 나열합니다.

자세한 내용은 콘솔로 Lambda 함수 생성을 참조하세요.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

로그를 작성할 수 있는 Lambda 함수에 대한 사전 요구 IAM 권한을 설정합니다.

```
# Get an AWS Identity and Access Management (IAM) role.
#
# @param iam_role_name: The name of the role to retrieve.
# @param action: Whether to create or destroy the IAM apparatus.
# @return: The IAM role.
def manage_iam(iam_role_name, action)
  case action
  when 'create'
    create_iam_role(iam_role_name)
  when 'destroy'
    destroy_iam_role(iam_role_name)
  else
    raise "Incorrect action provided. Must provide 'create' or 'destroy'"
  end
end

private

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

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

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

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

호출 파라미터로 제공된 숫자를 증가 시키는 Lambda 핸들러를 정의합니다.

```
require 'logger'

# A function that increments a whole number by one (1) and logs the result.
# Requires a manually-provided runtime parameter, 'number', which must be Int
#
# @param event [Hash] Parameters sent when the function is invoked
```

```
# @param context [Hash] Methods and properties that provide information
# about the invocation, function, and execution environment.
# @return incremented_number [String] The incremented number.
def lambda_handler(event:, context:)
  logger = Logger.new($stdout)
  log_level = ENV['LOG_LEVEL']
  logger.level = case log_level
                 when 'debug'
                   Logger::DEBUG
                 when 'info'
                   Logger::INFO
                 else
                   Logger::ERROR
                 end
  logger.debug('This is a debug log message.')
  logger.info('This is an info log message. Code executed successfully!')
  number = event['number'].to_i
  incremented_number = number + 1
  logger.info("You provided #{number.round} and it was incremented to
 #{incremented_number.round}")
  incremented_number.round.to_s
end
```

Lambda 함수를 배포 패키지로 압축합니다.

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

새 Lambda 함수를 생성합니다.

```
  # Deploys a Lambda function.
  #
  # @param function_name: The name of the Lambda function.
  # @param handler_name: The fully qualified name of the handler function.
  # @param role_arn: The IAM role to use for the function.
  # @param deployment_package: The deployment package that contains the function
code in .zip format.
  # @return: The Amazon Resource Name (ARN) of the newly created function.
  def create_function(function_name, handler_name, role_arn, deployment_package)
    response = @lambda_client.create_function({
                                                role: role_arn.to_s,
                                                function_name: function_name,
                                                handler: handler_name,
                                                runtime: 'ruby2.7',
                                                code: {
                                                  zip_file: deployment_package
                                                },
                                                environment: {
                                                  variables: {
                                                    'LOG_LEVEL' => 'info'
                                                  }
                                                }
                                              })
    @lambda_client.wait_until(:function_active_v2, { function_name: function_name })
do |w|
      w.max_attempts = 5
      w.delay = 5
    end
    response
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error creating #{function_name}:\n #{e.message}")
  rescue Aws::Waiters::Errors::WaiterFailed => e
    @logger.error("Failed waiting for #{function_name} to activate:\n #{e.message}")
  end
```

선택적 런타임 파라미터를 사용하여 Lambda 함수를 간접 호출합니다.

```
  # 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
```

Lambda 함수의 구성을 업데이트하여 새 환경 변수를 삽입합니다.

```
  # 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
```

Lambda 함수의 코드를 다른 코드가 포함된 다른 배포 패키지로 업데이트하세요.

```
  # 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
```

내장 페이지네이터를 사용하여 기존의 모든 Lambda 함수를 나열합니다.

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

특정 Lambda 함수를 삭제합니다.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 다음 항목을 참조하세요.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - 간접 호출
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

작업

`CreateFunction`

다음 코드 예시는 `CreateFunction`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

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

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 CreateFunction을 참조하세요.

`DeleteFunction`

다음 코드 예시는 `DeleteFunction`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

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

  # Deletes a Lambda function.
  # @param function_name: The name of the function to delete.
  def delete_function(function_name)
    print "Deleting function: #{function_name}..."
    @lambda_client.delete_function(
      function_name: function_name
    )
    print 'Done!'.green
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error deleting #{function_name}:\n #{e.message}")
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteFunction을 참조하세요.

`GetFunction`

다음 코드 예시는 `GetFunction`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

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

  # Gets data about a Lambda function.
  #
  # @param function_name: The name of the function.
  # @return response: The function data, or nil if no such function exists.
  def get_function(function_name)
    @lambda_client.get_function(
      {
        function_name: function_name
      }
    )
  rescue Aws::Lambda::Errors::ResourceNotFoundException => e
    @logger.debug("Could not find function: #{function_name}:\n #{e.message}")
    nil
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 GetFunction을 참조하세요.

`Invoke`

다음 코드 예시는 `Invoke`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

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

  # Invokes a Lambda function.
  # @param function_name [String] The name of the function to invoke.
  # @param payload [nil] Payload containing runtime parameters.
  # @return [Object] The response from the function invocation.
  def invoke_function(function_name, payload = nil)
    params = { function_name: function_name }
    params[:payload] = payload unless payload.nil?
    @lambda_client.invoke(params)
  rescue Aws::Lambda::Errors::ServiceException => e
    @logger.error("There was an error executing #{function_name}:\n #{e.message}")
  end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 간접 호출을 참조하세요.

`ListFunctions`

다음 코드 예시는 `ListFunctions`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

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

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListFunctions를 참조하세요.

`UpdateFunctionCode`

다음 코드 예시는 `UpdateFunctionCode`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

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

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 UpdateFunctionCode를 참조하세요.

`UpdateFunctionConfiguration`

다음 코드 예시는 `UpdateFunctionConfiguration`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class LambdaWrapper
  attr_accessor :lambda_client, :cloudwatch_client, :iam_client

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

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

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 UpdateFunctionConfiguration을 참조하세요.

시나리오

고객 피드백 분석을 위한 애플리케이션 생성

다음 코드 예제에서는 고객 의견 카드를 분석하고, 원어에서 번역하고, 감정을 파악하고, 번역된 텍스트에서 오디오 파일을 생성하는 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Ruby

이 예제 애플리케이션은 고객 피드백 카드를 분석하고 저장합니다. 특히 뉴욕시에 있는 가상 호텔의 필요를 충족합니다. 호텔은 다양한 언어의 고객들로부터 물리적인 의견 카드의 형태로 피드백을 받습니다. 피드백은 웹 클라이언트를 통해 앱에 업로드됩니다. 의견 카드의 이미지가 업로드된 후 다음 단계가 수행됩니다.

- Amazon Textract를 사용하여 이미지에서 텍스트가 추출됩니다.
- Amazon Comprehend가 추출된 텍스트와 해당 언어의 감정을 파악합니다.
- 추출된 텍스트는 Amazon Translate를 사용하여 영어로 번역됩니다.
- Amazon Polly가 추출된 텍스트에서 오디오 파일을 합성합니다.

전체 앱은 AWS CDK를 사용하여 배포할 수 있습니다. 소스 코드와 배포 지침은 GitHub의 프로젝트를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Comprehend

- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

서버리스 예제

Lambda 함수를 사용하여 Amazon RDS 데이터베이스에 연결

다음 코드 예제는 RDS 데이터베이스에 연결하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 간단한 데이터베이스 요청을 하고 결과를 반환합니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Ruby를 사용하여 Lambda 함수에서 Amazon RDS 데이터베이스에 연결

```
# Ruby code here.

require 'aws-sdk-rds'
require 'json'
require 'mysql2'

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

  credentials = Aws::Credentials.new(
    ENV['AWS_ACCESS_KEY_ID'],
    ENV['AWS_SECRET_ACCESS_KEY'],
    ENV['AWS_SESSION_TOKEN']
```

```
  )
  rds_client = Aws::RDS::AuthTokenGenerator.new(
    region: region,
    credentials: credentials
  )

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

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

Kinesis 트리거에서 간접적으로 Lambda 함수 간접 호출

다음 코드 예제에서는 Kinesis 스트림에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 Kinesis 페이로드를 검색하고, Base64에서 디코딩하고, 레코드 콘텐츠를 로깅합니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Ruby를 사용하여 Lambda로 Kinesis 이벤트를 사용합니다.

```
# 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
```

DynamoDB 트리거에서 간접적으로 Lambda 함수 간접 호출

다음 코드 예제에서는 DynamoDB 스트림에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 DynamoDB 페이로드를 검색하고 레코드 콘텐츠를 로깅합니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Ruby를 사용하여 Lambda로 DynamoDB 이벤트 사용.

```
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
```

Amazon DocumentDB 트리거에서 간접적으로 Lambda 함수 호출

다음 코드 예제에서는 DocumentDB 변경 스트림에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 DocumentDB 페이로드를 검색하고 레코드 콘텐츠를 로깅합니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Ruby를 사용하여 Lambda로 Amazon DocumentDB 이벤트 소비

```
require 'json'

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

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

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

Amazon MSK 트리거를 사용하여 Lambda 함수 간접 호출

다음 코드 예제에서는 Amazon MSK 클러스터에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 MSK 페이로드를 검색하고 레코드 콘텐츠를 로깅합니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Ruby를 사용하여 Lambda로 Amazon MSK 이벤트 사용

```
require 'base64'

def lambda_handler(event:, context:)
  # Iterate through keys
  event['records'].each do |key, records|
    puts "Key: #{key}"

    # Iterate through records
    records.each do |record|
      puts "Record: #{record}"

      # Decode base64
      msg = Base64.decode64(record['value'])
      puts "Message: #{msg}"
    end
  end
end
```

Amazon S3 트리거를 사용하여 Lambda 함수 간접 호출

다음 코드 예제는 S3 버킷에 객체를 업로드하여 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 해당 함수는 이벤트 파라미터에서 S3 버킷 이름과 객체 키를 검색하고 Amazon S3 API를 호출하여 객체의 콘텐츠 유형을 검색하고 로깅합니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Ruby를 사용하여 Lambda로 S3 이벤트 사용.

```
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 SNS 트리거를 사용하여 Lambda 함수 간접 호출

다음 코드 예제에서는 SNS 주제의 메시지를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 함수는 이벤트 파라미터에서 메시지를 검색하고 각 메시지의 내용을 로깅합니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Ruby를 사용하여 Lambda로 SNS 이벤트를 사용합니다.

```
# 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 트리거에서 간접적으로 Lambda 함수 간접 호출

다음 코드 예제는 SQS 대기열에서 메시지를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 함수는 이벤트 파라미터에서 메시지를 검색하고 각 메시지의 내용을 로깅합니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Ruby를 사용하여 Lambda로 SQS 이벤트를 사용합니다.

```
# 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
```

Kinesis 트리거로 Lambda 함수에 대한 배치 항목 실패 보고

다음 코드 예제는 Kinesis 스트림에서 이벤트를 수신하는 Lambda 함수에 대한 부분 배치 응답을 구현하는 방법을 보여줍니다. 이 함수는 응답으로 배치 항목 실패를 보고하고 나중에 해당 메시지를 다시 시도하도록 Lambda에 신호를 보냅니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Ruby를 사용하여 Lambda로 Kinesis 배치 항목 실패를 보고합니다.

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

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

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

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

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

```
end
```

DynamoDB 트리거로 Lambda 함수에 대한 배치 항목 실패 보고

다음 코드 예제에서는 DynamoDB 스트림에서 이벤트를 수신하는 Lambda 함수에 대한 부분 배치 응답을 구현하는 방법을 보여줍니다. 이 함수는 응답으로 배치 항목 실패를 보고하고 나중에 해당 메시지를 다시 시도하도록 Lambda에 신호를 보냅니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Ruby를 사용하여 Lambda로 DynamoDB 배치 항목 실패 보고.

```
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
```

Amazon SQS 트리거로 Lambda 함수에 대한 배치 항목 실패 보고

다음 코드 예제는 SQS 대기열에서 이벤트를 수신하는 Lambda 함수에 대한 부분 배치 응답을 구현하는 방법을 보여줍니다. 이 함수는 응답으로 배치 항목 실패를 보고하고 나중에 해당 메시지를 다시 시도하도록 Lambda에 신호를 보냅니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Ruby를 사용하여 Lambda로 SQS 배치 항목 실패를 보고합니다.

```
# 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
```

SDK for Ruby를 사용한 Amazon MSK 예제

다음 코드 예제에서는 Amazon MSK와 AWS SDK for Ruby 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

서버리스 예제

Amazon MSK 트리거를 사용하여 Lambda 함수 간접 호출

다음 코드 예제에서는 Amazon MSK 클러스터에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 MSK 페이로드를 검색하고 레코드 콘텐츠를 로깅합니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Ruby를 사용하여 Lambda로 Amazon MSK 이벤트 사용

```
require 'base64'

def lambda_handler(event:, context:)
  # Iterate through keys
  event['records'].each do |key, records|
    puts "Key: #{key}"

    # Iterate through records
    records.each do |record|
      puts "Record: #{record}"
```

```
      # Decode base64
      msg = Base64.decode64(record['value'])
      puts "Message: #{msg}"
    end
  end
end
```

SDK for Ruby를 사용한 Amazon Polly 예제

다음 코드 예제에서는 AWS SDK for Ruby Amazon Polly에서를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`DescribeVoices`

다음 코드 예시는 `DescribeVoices`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-polly' # In v2: require 'aws-sdk'

begin
  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  # Get US English voices
  resp = polly.describe_voices(language_code: 'en-US')

  resp.voices.each do |v|
    puts v.name
    puts "  #{v.gender}"
    puts
  end
rescue StandardError => e
  puts 'Could not get voices'
  puts 'Error message:'
  puts e.message
end
```

- API에 대한 세부 정보는 AWS SDK for Ruby API 참조의 DescribeVoices를 참조하세요.

`ListLexicons`

다음 코드 예시는 `ListLexicons`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-polly' # In v2: require 'aws-sdk'
```

```
begin
  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  resp = polly.list_lexicons

  resp.lexicons.each do |l|
    puts l.name
    puts "  Alphabet:#{l.attributes.alphabet}"
    puts "  Language:#{l.attributes.language}"
    puts
  end
rescue StandardError => e
  puts 'Could not get lexicons'
  puts 'Error message:'
  puts e.message
end
```

- API에 대한 세부 정보는 AWS SDK for Ruby API 참조의 ListLexicons를 참조하세요.

`SynthesizeSpeech`

다음 코드 예시는 `SynthesizeSpeech`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-polly' # In v2: require 'aws-sdk'

begin
  # Get the filename from the command line
  if ARGV.empty?
    puts 'You must supply a filename'
```

```
    exit 1
  end

  filename = ARGV[0]

  # Open file and get the contents as a string
  if File.exist?(filename)
    contents = IO.read(filename)
  else
    puts "No such file: #{filename}"
    exit 1
  end

  # Create an Amazon Polly client using
  # credentials from the shared credentials file ~/.aws/credentials
  # and the configuration (region) from the shared configuration file ~/.aws/config
  polly = Aws::Polly::Client.new

  resp = polly.synthesize_speech({
                                   output_format: 'mp3',
                                   text: contents,
                                   voice_id: 'Joanna'
                                 })

  # Save output
  # Get just the file name
  #  abc/xyz.txt -> xyx.txt
  name = File.basename(filename)

  # Split up name so we get just the xyz part
  parts = name.split('.')
  first_part = parts[0]
  mp3_file = "#{first_part}.mp3"

  IO.copy_stream(resp.audio_stream, mp3_file)

  puts "Wrote MP3 content to: #{mp3_file}"
rescue StandardError => e
  puts 'Got error:'
  puts 'Error message:'
  puts e.message
end
```

- API에 대한 세부 정보는 AWS SDK for Ruby API 참조의 SynthesizeSpeech를 참조하세요.

시나리오

고객 피드백 분석을 위한 애플리케이션 생성

다음 코드 예제에서는 고객 의견 카드를 분석하고, 원어에서 번역하고, 감정을 파악하고, 번역된 텍스트에서 오디오 파일을 생성하는 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Ruby

이 예제 애플리케이션은 고객 피드백 카드를 분석하고 저장합니다. 특히 뉴욕시에 있는 가상 호텔의 필요를 충족합니다. 호텔은 다양한 언어의 고객들로부터 물리적인 의견 카드의 형태로 피드백을 받습니다. 피드백은 웹 클라이언트를 통해 앱에 업로드됩니다. 의견 카드의 이미지가 업로드된 후 다음 단계가 수행됩니다.

- Amazon Textract를 사용하여 이미지에서 텍스트가 추출됩니다.
- Amazon Comprehend가 추출된 텍스트와 해당 언어의 감정을 파악합니다.
- 추출된 텍스트는 Amazon Translate를 사용하여 영어로 번역됩니다.
- Amazon Polly가 추출된 텍스트에서 오디오 파일을 합성합니다.

전체 앱은 AWS CDK를 사용하여 배포할 수 있습니다. 소스 코드와 배포 지침은 GitHub의 프로젝트를 참조하세요.

이 예제에서 사용되는 서비스

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

SDK for Ruby를 사용한 Amazon RDS 예제

다음 코드 예제에서는 AWS SDK for Ruby Amazon RDS에서를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시작하기

Hello Amazon RDS

다음 코드 예제에서는 Amazon RDS 사용을 시작하는 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-rds'
require 'logger'

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

  # Lists and prints all RDS DB instances in the current AWS account.
```

```
  def list_db_instances
    @logger.info('Listing RDS DB instances')

    paginator = @client.describe_db_instances
    instances = []

    paginator.each_page do |page|
      instances.concat(page.db_instances)
    end

    if instances.empty?
      @logger.info('No instances found.')
    else
      @logger.info("Found #{instances.count} instance(s):")
      instances.each do |instance|
        @logger.info(" * #{instance.db_instance_identifier}
 (#{instance.db_instance_status})")
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  rds_client = Aws::RDS::Client.new(region: 'us-west-2')
  manager = RDSManager.new(rds_client)
  manager.list_db_instances
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DescribeDBInstances를 참조하세요.

작업

`CreateDBSnapshot`

다음 코드 예시는 `CreateDBSnapshot`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 CreateDBSnapshot을 참조하세요.

`DescribeDBInstances`

다음 코드 예시는 `DescribeDBInstances`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DescribeDBInstances를 참조하세요.

`DescribeDBParameterGroups`

다음 코드 예시는 `DescribeDBParameterGroups`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DescribeDBParameterGroups를 참조하세요.

`DescribeDBParameters`

다음 코드 예시는 `DescribeDBParameters`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DescribeDBParameters 참조하세요.

`DescribeDBSnapshots`

다음 코드 예시는 `DescribeDBSnapshots`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DescribeDBSnapshots를 참조하세요.

서버리스 예제

Lambda 함수를 사용하여 Amazon RDS 데이터베이스에 연결

다음 코드 예제는 RDS 데이터베이스에 연결하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 간단한 데이터베이스 요청을 하고 결과를 반환합니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Ruby를 사용하여 Lambda 함수에서 Amazon RDS 데이터베이스에 연결

```
# Ruby code here.

require 'aws-sdk-rds'
require 'json'
require 'mysql2'

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

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

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

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

SDK for Ruby를 사용한 Amazon S3 예제

다음 코드 예제에서는 AWS SDK for Ruby Amazon S3에서를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시작하기

Hello Amazon S3

다음 코드 예제에서는 Amazon S3 사용을 시작하는 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# frozen_string_literal: true

# S3Manager is a class responsible for managing S3 operations
# such as listing all S3 buckets in the current AWS account.
class S3Manager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end
```

```
  # Lists and prints all S3 buckets in the current AWS account.
  def list_buckets
    @logger.info('Here are the buckets in your account:')

    response = @client.list_buckets

    if response.buckets.empty?
      @logger.info("You don't have any S3 buckets yet.")
    else
      response.buckets.each do |bucket|
        @logger.info("- #{bucket.name}")
      end
    end
  rescue Aws::Errors::ServiceError => e
    @logger.error("Encountered an error while listing buckets: #{e.message}")
  end
end

if $PROGRAM_NAME == __FILE__
  s3_client = Aws::S3::Client.new
  manager = S3Manager.new(s3_client)
  manager.list_buckets
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListBuckets를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 버킷을 만들고 버킷에 파일을 업로드합니다.
- 버킷에서 객체를 다운로드합니다.
- 버킷의 하위 폴더에 객체를 복사합니다.
- 버킷의 객체를 나열합니다.
- 버킷 객체와 버킷을 삭제합니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-s3'

# Wraps the getting started scenario actions.
class ScenarioGettingStarted
  attr_reader :s3_resource

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

  # Creates a bucket with a random name in the currently configured account and
  # AWS Region.
  #
  # @return [Aws::S3::Bucket] The newly created bucket.
  def create_bucket
    bucket = @s3_resource.create_bucket(
      bucket: "amzn-s3-demo-bucket-#{Random.uuid}",
      create_bucket_configuration: {
        location_constraint: 'us-east-1' # NOTE: only certain regions permitted
      }
    )
    puts("Created demo bucket named #{bucket.name}.")
  rescue Aws::Errors::ServiceError => e
    puts('Tried and failed to create demo bucket.')
    puts("\t#{e.code}: #{e.message}")
    puts("\nCan't continue the demo without a bucket!")
    raise
  else
    bucket
  end

  # Requests a file name from the user.
  #
```

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

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

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

  # Copies an Amazon S3 object to a subfolder within the same bucket.
  #
```

```
  # @param source_object [Aws::S3::Object] The source object to copy.
  # @return [Aws::S3::Object, nil] The destination object.
  def copy_object(source_object)
    dest_object = nil
    puts("\nDo you want to copy #{source_object.key} to a subfolder in your bucket
(y/n)? ")
    answer = gets.chomp.downcase
    if answer == 'y'
      dest_object = source_object.bucket.object("demo-folder/#{source_object.key}")
      dest_object.copy_from(source_object)
      puts("Copied #{source_object.key} to #{dest_object.key}.")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't copy #{source_object.key}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  else
    dest_object
  end

  # Lists the objects in an Amazon S3 bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket to query.
  def list_objects(bucket)
    puts("\nYour bucket contains the following objects:")
    bucket.objects.each do |obj|
      puts("\t#{obj.key}")
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't list the objects in bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end

  # Deletes the objects in an Amazon S3 bucket and deletes the bucket.
  #
  # @param bucket [Aws::S3::Bucket] The bucket to empty and delete.
  def delete_bucket(bucket)
    puts("\nDo you want to delete all of the objects as well as the bucket (y/n)? ")
    answer = gets.chomp.downcase
    if answer == 'y'
      bucket.objects.batch_delete!
      bucket.delete
      puts("Emptied and deleted bucket #{bucket.name}.\n")
```

```
    end
  rescue Aws::Errors::ServiceError => e
    puts("Couldn't empty and delete bucket #{bucket.name}.")
    puts("\t#{e.code}: #{e.message}")
    raise
  end
end

# Runs the Amazon S3 getting started scenario.
def run_scenario(scenario)
  puts('-' * 88)
  puts('Welcome to the Amazon S3 getting started demo!')
  puts('-' * 88)

  bucket = scenario.create_bucket
  s3_object = scenario.upload_file(bucket)
  scenario.download_file(s3_object)
  scenario.copy_object(s3_object)
  scenario.list_objects(bucket)
  scenario.delete_bucket(bucket)

  puts('Thanks for watching!')
  puts('-' * 88)
rescue Aws::Errors::ServiceError
  puts('Something went wrong with the demo!')
end

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

- API 세부 정보는 AWS SDK for Ruby API 참조의 다음 항목을 참조하세요.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

작업

`CopyObject`

다음 코드 예시는 `CopyObject`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

객체를 복사합니다.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectCopyWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                        copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket and rename it with the
 target key.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's why:
 #{e.message}"
```

```
  end
end

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

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

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

run_demo if $PROGRAM_NAME == __FILE__
```

객체를 복사하고 대상 객체에 서버 측 암호화를 추가합니다.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectCopyEncryptWrapper
  attr_reader :source_object

  # @param source_object [Aws::S3::Object] An existing Amazon S3 object. This is
 used as the source object for
  #                                        copy actions.
  def initialize(source_object)
    @source_object = source_object
  end

  # Copy the source object to the specified target bucket, rename it with the target
 key, and encrypt it.
  #
  # @param target_bucket [Aws::S3::Bucket] An existing Amazon S3 bucket where the
 object is copied.
```

```
  # @param target_object_key [String] The key to give the copy of the object.
  # @return [Aws::S3::Object, nil] The copied object when successful; otherwise,
 nil.
  def copy_object(target_bucket, target_object_key, encryption)
    @source_object.copy_to(bucket: target_bucket.name, key: target_object_key,
 server_side_encryption: encryption)
    target_bucket.object(target_object_key)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't copy #{@source_object.key} to #{target_object_key}. Here's why:
 #{e.message}"
  end
end

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

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

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

run_demo if $PROGRAM_NAME == __FILE__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 CopyObject를 참조하세요.

`CreateBucket`

다음 코드 예시는 `CreateBucket`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket actions.
class BucketCreateWrapper
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An Amazon S3 bucket initialized with a name.
 This is a client-side object until
  #                                 create is called.
  def initialize(bucket)
    @bucket = bucket
  end

  # Creates an Amazon S3 bucket in the specified AWS Region.
  #
  # @param region [String] The Region where the bucket is created.
  # @return [Boolean] True when the bucket is created; otherwise, false.
  def create?(region)
    @bucket.create(create_bucket_configuration: { location_constraint: region })
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't create bucket. Here's why: #{e.message}"
    false
  end

  # Gets the Region where the bucket is located.
  #
  # @return [String] The location of the bucket.
  def location
    if @bucket.nil?
      'None. You must create a bucket before you can get its location!'
    else
      @bucket.client.get_bucket_location(bucket: @bucket.name).location_constraint
    end
```

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

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

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

run_demo if $PROGRAM_NAME == __FILE__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 CreateBucket을 참조하세요.

`DeleteBucket`

다음 코드 예시는 `DeleteBucket`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
  # 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteBucket을 참조하세요.

`DeleteBucketCors`

다음 코드 예시는 `DeleteBucketCors`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteBucketCors를 참조하세요.

`DeleteBucketPolicy`

다음 코드 예시는 `DeleteBucketPolicy`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteBucketPolicy를 참조하세요.

`DeleteObjects`

다음 코드 예시는 `DeleteObjects`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteObjects를 참조하세요.

`GetBucketCors`

다음 코드 예시는 `GetBucketCors`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with an
 existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Gets the CORS configuration of a bucket.
  #
  # @return [Aws::S3::Type::GetBucketCorsOutput, nil] The current CORS configuration
 for the bucket.
  def cors
    @bucket_cors.data
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get CORS configuration for #{@bucket_cors.bucket.name}. Here's
 why: #{e.message}"
    nil
  end

end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 GetBucketCors를 참조하세요.

`GetBucketPolicy`

다음 코드 예시는 `GetBucketPolicy`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object configured
 with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Gets the policy of a bucket.
  #
  # @return [Aws::S3::GetBucketPolicyOutput, nil] The current bucket policy.
  def policy
    policy = @bucket_policy.data.policy
    policy.respond_to?(:read) ? policy.read : policy
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get the policy for #{@bucket_policy.bucket.name}. Here's why:
 #{e.message}"
    nil
  end

end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 GetBucketPolicy를 참조하세요.

`GetObject`

다음 코드 예시는 `GetObject`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

객체를 가져옵니다.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectGetWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object directly to a file.
  #
  # @param target_path [String] The path to the file where the object is downloaded.
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
 successful; otherwise nil.
  def get_object(target_path)
    @object.get(response_target: target_path)
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

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

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

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

run_demo if $PROGRAM_NAME == __FILE__
```

객체를 가져와 서버 측 암호화 상태를 보고합니다.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectGetEncryptionWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Gets the object into memory.
  #
  # @return [Aws::S3::Types::GetObjectOutput, nil] The retrieved object data if
 successful; otherwise nil.
  def object
    @object.get
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't get object #{@object.key}. Here's why: #{e.message}"
  end
end

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

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

  encryption = obj_data.server_side_encryption.nil? ? 'no' :
 obj_data.server_side_encryption
```

```
  puts "Object #{object_key} uses #{encryption} encryption."
end

run_demo if $PROGRAM_NAME == __FILE__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 GetObject를 참조하세요.

`HeadObject`

다음 코드 예시는 `HeadObject`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectExistsWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Checks whether the object exists.
  #
  # @return [Boolean] True if the object exists; otherwise false.
  def exists?
    @object.exists?
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't check existence of object #{@object.bucket.name}:#{@object.key}.
 Here's why: #{e.message}"
    false
  end
end
```

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

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

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

run_demo if $PROGRAM_NAME == __FILE__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 HeadObject를 참조하세요.

`ListBuckets`

다음 코드 예시는 `ListBuckets`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListBuckets를 참조하세요.

`ListObjectsV2`

다음 코드 예시는 `ListObjectsV2`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket actions.
class BucketListObjectsWrapper
```

```
  attr_reader :bucket

  # @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
  def initialize(bucket)
    @bucket = bucket
  end

  # Lists object in a bucket.
  #
  # @param max_objects [Integer] The maximum number of objects to list.
  # @return [Integer] The number of objects listed.
  def list_objects(max_objects)
    count = 0
    puts "The objects in #{@bucket.name} are:"
    @bucket.objects.each do |obj|
      puts "\t#{obj.key}"
      count += 1
      break if count == max_objects
    end
    count
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't list objects in bucket #{bucket.name}. Here's why: #{e.message}"
    0
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"

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

run_demo if $PROGRAM_NAME == __FILE__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListObjectsV2를 참조하세요.

`PutBucketCors`

다음 코드 예시는 `PutBucketCors`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket CORS configuration.
class BucketCorsWrapper
  attr_reader :bucket_cors

  # @param bucket_cors [Aws::S3::BucketCors] A bucket CORS object configured with an
 existing bucket.
  def initialize(bucket_cors)
    @bucket_cors = bucket_cors
  end

  # Sets CORS rules on a bucket.
  #
  # @param allowed_methods [Array<String>] The types of HTTP requests to allow.
  # @param allowed_origins [Array<String>] The origins to allow.
  # @returns [Boolean] True if the CORS rules were set; otherwise, false.
  def set_cors(allowed_methods, allowed_origins)
    @bucket_cors.put(
      cors_configuration: {
        cors_rules: [
          {
            allowed_methods: allowed_methods,
            allowed_origins: allowed_origins,
            allowed_headers: %w[*],
            max_age_seconds: 3600
          }
        ]
      }
    )
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't set CORS rules for #{@bucket_cors.bucket.name}. Here's why:
 #{e.message}"
```

```
    false
  end

end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 PutBucketCors를 참조하세요.

`PutBucketPolicy`

다음 코드 예시는 `PutBucketPolicy`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# Wraps an Amazon S3 bucket policy.
class BucketPolicyWrapper
  attr_reader :bucket_policy

  # @param bucket_policy [Aws::S3::BucketPolicy] A bucket policy object configured
 with an existing bucket.
  def initialize(bucket_policy)
    @bucket_policy = bucket_policy
  end

  # Sets a policy on a bucket.
  #
  def policy(policy)
    @bucket_policy.put(policy: policy)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't set the policy for #{@bucket_policy.bucket.name}. Here's why:
 #{e.message}"
    false
  end

end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 PutBucketPolicy를 참조하세요.

`PutBucketWebsite`

다음 코드 예시는 `PutBucketWebsite`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 bucket website actions.
class BucketWebsiteWrapper
  attr_reader :bucket_website

  # @param bucket_website [Aws::S3::BucketWebsite] A bucket website object
 configured with an existing bucket.
  def initialize(bucket_website)
    @bucket_website = bucket_website
  end

  # Sets a bucket as a static website.
  #
  # @param index_document [String] The name of the index document for the website.
  # @param error_document [String] The name of the error document to show for 4XX
 errors.
  # @return [Boolean] True when the bucket is configured as a website; otherwise,
 false.
  def set_website(index_document, error_document)
    @bucket_website.put(
      website_configuration: {
        index_document: { suffix: index_document },
        error_document: { key: error_document }
      }
    )
```

```
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't configure #{@bucket_website.bucket.name} as a website. Here's
 why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  index_document = "index.html"
  error_document = "404.html"

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

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

run_demo if $PROGRAM_NAME == __FILE__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 PutBucketWebsite를 참조하세요.

`PutObject`

다음 코드 예시는 `PutObject`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

관리형 업로더(Object.upload_file)를 사용하여 파일을 업로드합니다.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
```

```
class ObjectUploadFileWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  # Uploads a file to an Amazon S3 object by using a managed uploader.
  #
  # @param file_path [String] The path to the file to upload.
  # @return [Boolean] True when the file is uploaded; otherwise false.
  def upload_file(file_path)
    @object.upload_file(file_path)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't upload file #{file_path} to #{@object.key}. Here's why:
 #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-uploaded-file"
  file_path = "object_upload_file.rb"

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

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

run_demo if $PROGRAM_NAME == __FILE__
```

Object.put을 사용하여 파일을 업로드합니다.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
```

```
class ObjectPutWrapper
  attr_reader :object

  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object(source_file_path)
    File.open(source_file_path, 'rb') do |file|
      @object.put(body: file)
    end
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put #{source_file_path} to #{object.key}. Here's why:
 #{e.message}"
    false
  end
end

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

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

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

run_demo if $PROGRAM_NAME == __FILE__
```

Object.put을 사용하여 파일을 업로드하고 서버 측 암호화를 추가합니다.

```
require 'aws-sdk-s3'

# Wraps Amazon S3 object actions.
class ObjectPutSseWrapper
  attr_reader :object
```

```
  # @param object [Aws::S3::Object] An existing Amazon S3 object.
  def initialize(object)
    @object = object
  end

  def put_object_encrypted(object_content, encryption)
    @object.put(body: object_content, server_side_encryption: encryption)
    true
  rescue Aws::Errors::ServiceError => e
    puts "Couldn't put your content to #{object.key}. Here's why: #{e.message}"
    false
  end
end

# Example usage:
def run_demo
  bucket_name = "amzn-s3-demo-bucket"
  object_key = "my-encrypted-content"
  object_content = "This is my super-secret content."
  encryption = "AES256"

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

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

run_demo if $PROGRAM_NAME == __FILE__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 PutObject를 참조하세요.

시나리오

미리 서명된 URL 생성

다음 코드 예제는 Amazon S3에 대해 미리 서명된 URL을 생성하고 객체를 업로드하는 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-s3'
require 'net/http'

# Creates a presigned URL that can be used to upload content to an object.
#
# @param bucket [Aws::S3::Bucket] An existing Amazon S3 bucket.
# @param object_key [String] The key to give the uploaded object.
# @return [URI, nil] The parsed URI if successful; otherwise nil.
def get_presigned_url(bucket, object_key)
  url = bucket.object(object_key).presigned_url(:put)
  puts "Created presigned URL: #{url}"
  URI(url)
rescue Aws::Errors::ServiceError => e
  puts "Couldn't create presigned URL for #{bucket.name}:#{object_key}. Here's why:
 #{e.message}"
end

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

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

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

  case response
  when Net::HTTPSuccess
```

```
    puts 'Content uploaded!'
  else
    puts response.value
  end
end

run_demo if $PROGRAM_NAME == __FILE__
```

서버리스 예제

Amazon S3 트리거를 사용하여 Lambda 함수 간접 호출

다음 코드 예제는 S3 버킷에 객체를 업로드하여 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 해당 함수는 이벤트 파라미터에서 S3 버킷 이름과 객체 키를 검색하고 Amazon S3 API를 호출하여 객체의 콘텐츠 유형을 검색하고 로깅합니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Ruby를 사용하여 Lambda로 S3 이벤트 사용.

```
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
```

SDK for Ruby를 사용한 Amazon SES 예제

다음 코드 예제에서는 AWS SDK for Ruby Amazon SES에서를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`GetIdentityVerificationAttributes`

다음 코드 예시는 `GetIdentityVerificationAttributes`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

# Create client in us-west-2 region
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
client = Aws::SES::Client.new(region: 'us-west-2')

# Get up to 1000 identities
ids = client.list_identities({
                               identity_type: 'EmailAddress'
                             })

ids.identities.each do |email|
  attrs = client.get_identity_verification_attributes({
                                                        identities: [email]
                                                      })

  status = attrs.verification_attributes[email].verification_status

  # Display email addresses that have been verified
  puts email if status == 'Success'
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 GetIdentityVerificationAttributes를 참조하세요.

`ListIdentities`

다음 코드 예시는 `ListIdentities`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

# Create client in us-west-2 region
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
client = Aws::SES::Client.new(region: 'us-west-2')

# Get up to 1000 identities
ids = client.list_identities({
                               identity_type: 'EmailAddress'
                             })

ids.identities.each do |email|
  attrs = client.get_identity_verification_attributes({
                                                        identities: [email]
                                                      })

  status = attrs.verification_attributes[email].verification_status

  # Display email addresses that have been verified
  puts email if status == 'Success'
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ListIdentities를 참조하세요.

`SendEmail`

다음 코드 예시는 `SendEmail`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

# Replace sender@example.com with your "From" address.
# This address must be verified with Amazon SES.
sender = 'sender@example.com'

# Replace recipient@example.com with a "To" address. If your account
# is still in the sandbox, this address must be verified.
recipient = 'recipient@example.com'

# Specify a configuration set. To use a configuration
# set, uncomment the next line and line 74.
#   configsetname = "ConfigSet"

# The subject line for the email.
subject = 'Amazon SES test (AWS SDK for Ruby)'

# The HTML body of the email.
htmlbody =
  '<h1>Amazon SES test (AWS SDK for Ruby)</h1>'\
  '<p>This email was sent with <a href="https://aws.amazon.com/ses/">'\
  'Amazon SES</a> using the <a href="https://aws.amazon.com/sdk-for-ruby/">'\
  'AWS SDK for Ruby</a>.'

# The email body for recipients with non-HTML email clients.
textbody = 'This email was sent with Amazon SES using the AWS SDK for Ruby.'

# Specify the text encoding scheme.
encoding = 'UTF-8'

# Create a new SES client in the us-west-2 region.
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
ses = Aws::SES::Client.new(region: 'us-west-2')
```

```
# Try to send the email.
begin
  # Provide the contents of the email.
  ses.send_email(
    destination: {
      to_addresses: [
        recipient
      ]
    },
    message: {
      body: {
        html: {
          charset: encoding,
          data: htmlbody
        },
        text: {
          charset: encoding,
          data: textbody
        }
      },
      subject: {
        charset: encoding,
        data: subject
      }
    },
    source: sender
    # Uncomment the following line to use a configuration set.
    # configuration_set_name: configsetname,
  )

  puts "Email sent to #{recipient}"

# If something goes wrong, display an error message.
rescue Aws::SES::Errors::ServiceError => e
  puts "Email not sent. Error message: #{e}"
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 SendEmail을 참조하세요.

`VerifyEmailIdentity`

다음 코드 예시는 `VerifyEmailIdentity`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

# Replace recipient@example.com with a "To" address.
recipient = 'recipient@example.com'

# Create a new SES resource in the us-west-2 region.
# Replace us-west-2 with the AWS Region you're using for Amazon SES.
ses = Aws::SES::Client.new(region: 'us-west-2')

# Try to verify email address.
begin
  ses.verify_email_identity({
                              email_address: recipient
                            })

  puts "Email sent to #{recipient}"

# If something goes wrong, display an error message.
rescue Aws::SES::Errors::ServiceError => e
  puts "Email not sent. Error message: #{e}"
end
```

- API 세부 정보는 AWS SDK for Ruby SDK API 참조의 VerifyEmailIdentity를 참조하세요.

SDK for Ruby를 사용한 Amazon SES API v2 예제

다음 코드 예제에서는 Amazon SES API v2와 AWS SDK for Ruby 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

작업

`SendEmail`

다음 코드 예시는 `SendEmail`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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)
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 SendEmail을 참조하세요.

SDK for Ruby를 사용한 Amazon SNS 예제

다음 코드 예제에서는 AWS SDK for Ruby Amazon SNS에서를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업
- 서버리스 예제

작업

`CreateTopic`

다음 코드 예시는 `CreateTopic`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# This class demonstrates how to create an Amazon Simple Notification Service (SNS)
 topic.
class SNSTopicCreator
  # Initializes an SNS client.
  #
  # Utilizes the default AWS configuration for region and credentials.
  def initialize
    @sns_client = Aws::SNS::Client.new
  end

  # Attempts to create an SNS topic with the specified name.
  #
  # @param topic_name [String] The name of the SNS topic to create.
  # @return [Boolean] true if the topic was successfully created, false otherwise.
  def create_topic(topic_name)
    @sns_client.create_topic(name: topic_name)
    puts "The topic '#{topic_name}' was successfully created."
    true
  rescue Aws::SNS::Errors::ServiceError => e
    # Handles SNS service errors gracefully.
    puts "Error while creating the topic named '#{topic_name}': #{e.message}"
    false
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_name = 'YourTopicName' # Replace with your topic name
```

```
  sns_topic_creator = SNSTopicCreator.new

  puts "Creating the topic '#{topic_name}'..."
  unless sns_topic_creator.create_topic(topic_name)
    puts 'The topic was not created. Stopping program.'
    exit 1
  end
end
```

- 자세한 정보는 AWS SDK for Ruby 개발자 안내서를 참조하세요.
- API 세부 정보는 AWS SDK for Ruby API 참조의 CreateTopic을 참조하세요.

`ListSubscriptions`

다음 코드 예시는 `ListSubscriptions`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# This class demonstrates how to list subscriptions to an Amazon Simple Notification
 Service (SNS) topic
class SnsSubscriptionLister
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Lists subscriptions for a given SNS topic
  # @param topic_arn [String] The ARN of the SNS topic
  # @return [Types::ListSubscriptionsResponse] subscriptions: The response object
  def list_subscriptions(topic_arn)
    @logger.info("Listing subscriptions for topic: #{topic_arn}")
    subscriptions = @sns_client.list_subscriptions_by_topic(topic_arn: topic_arn)
    subscriptions.subscriptions.each do |subscription|
      @logger.info("Subscription endpoint: #{subscription.endpoint}")
```

```
    end
    subscriptions
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error listing subscriptions: #{e.message}")
    raise
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  sns_client = Aws::SNS::Client.new
  topic_arn = 'SNS_TOPIC_ARN' # Replace with your SNS topic ARN
  lister = SnsSubscriptionLister.new(sns_client)

  begin
    lister.list_subscriptions(topic_arn)
  rescue StandardError => e
    puts "Failed to list subscriptions: #{e.message}"
    exit 1
  end
end
```

- 자세한 정보는 AWS SDK for Ruby 개발자 안내서를 참조하세요.
- API 세부 정보는 AWS SDK for Ruby API 참조의 ListSubscriptions를 참조하세요.

`ListTopics`

다음 코드 예시는 `ListTopics`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

```
def list_topics?(sns_client)
  sns_client.topics.each do |topic|
    puts topic.arn
  rescue StandardError => e
    puts "Error while listing the topics: #{e.message}"
  end
end

def run_me
  region = 'REGION'
  sns_client = Aws::SNS::Resource.new(region: region)

  puts 'Listing the topics.'

  return if list_topics?(sns_client)

  puts 'The bucket was not created. Stopping program.'
  exit 1
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- 자세한 정보는 AWS SDK for Ruby 개발자 안내서를 참조하세요.
- API 세부 정보는 AWS SDK for Ruby API 참조의 ListTopics를 참조하세요.

`Publish`

다음 코드 예시는 `Publish`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# Service class for sending messages using Amazon Simple Notification Service (SNS)
```

```
class SnsMessageSender
  # Initializes the SnsMessageSender with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Sends a message to a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param message [String] The message to send
  # @return [Boolean] true if message was successfully sent, false otherwise
  def send_message(topic_arn, message)
    @sns_client.publish(topic_arn: topic_arn, message: message)
    @logger.info("Message sent successfully to #{topic_arn}.")
    true
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error while sending the message: #{e.message}")
    false
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = 'SNS_TOPIC_ARN' # Should be replaced with a real topic ARN
  message = 'MESSAGE'         # Should be replaced with the actual message content

  sns_client = Aws::SNS::Client.new
  message_sender = SnsMessageSender.new(sns_client)

  @logger.info('Sending message.')
  unless message_sender.send_message(topic_arn, message)
    @logger.error('Message sending failed. Stopping program.')
    exit 1
  end
end
```

- 자세한 정보는 AWS SDK for Ruby 개발자 안내서를 참조하세요.
- API 세부 정보는 AWS SDK for Ruby API 참조의 Publish를 참조하세요.

SetTopicAttributes

다음 코드 예시는 SetTopicAttributes의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# Service class to enable an SNS resource with a specified policy
class SnsResourceEnabler
  # Initializes the SnsResourceEnabler with an SNS resource client
  #
  # @param sns_resource [Aws::SNS::Resource] The SNS resource client
  def initialize(sns_resource)
    @sns_resource = sns_resource
    @logger = Logger.new($stdout)
  end

  # Sets a policy on a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource to include in the policy
  # @param policy_name [String] The name of the policy attribute to set
  def enable_resource(topic_arn, resource_arn, policy_name)
    policy = generate_policy(topic_arn, resource_arn)
    topic = @sns_resource.topic(topic_arn)

    topic.set_attributes({
                           attribute_name: policy_name,
                           attribute_value: policy
                         })
    @logger.info("Policy #{policy_name} set successfully for topic #{topic_arn}.")
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Failed to set policy: #{e.message}")
  end

  private

  # Generates a policy string with dynamic resource ARNs
```

```
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource
  # @return [String] The policy as a JSON string
  def generate_policy(topic_arn, resource_arn)
    {
      Version: '2008-10-17',
      Id: '__default_policy_ID',
      Statement: [{
        Sid: '__default_statement_ID',
        Effect: 'Allow',
        Principal: { "AWS": '*' },
        Action: ['SNS:Publish'],
        Resource: topic_arn,
        Condition: {
          ArnEquals: {
            "AWS:SourceArn": resource_arn
          }
        }
      }]
    }.to_json
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = 'MY_TOPIC_ARN' # Should be replaced with a real topic ARN
  resource_arn = 'MY_RESOURCE_ARN' # Should be replaced with a real resource ARN
  policy_name = 'POLICY_NAME' # Typically, this is "Policy"

  sns_resource = Aws::SNS::Resource.new
  enabler = SnsResourceEnabler.new(sns_resource)

  enabler.enable_resource(topic_arn, resource_arn, policy_name)
end
```

- 자세한 정보는 AWS SDK for Ruby 개발자 안내서를 참조하세요.
- API 세부 정보는 AWS SDK for Ruby API 참조의 SetTopicAttributes를 참조하세요.

Subscribe

다음 코드 예시는 Subscribe의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

이메일 주소로 주제 구독.

```
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
```

- 자세한 정보는 AWS SDK for Ruby 개발자 안내서를 참조하세요.
- API 세부 정보는 AWS SDK for Ruby API 참조의 Subscribe를 참조하세요.

서버리스 예제

Amazon SNS 트리거를 사용하여 Lambda 함수 간접 호출

다음 코드 예제에서는 SNS 주제의 메시지를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 함수는 이벤트 파라미터에서 메시지를 검색하고 각 메시지의 내용을 로깅합니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Ruby를 사용하여 Lambda로 SNS 이벤트를 사용합니다.

```
# 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
```

SDK for Ruby를 사용한 Amazon SQS 예제

다음 코드 예제에서는 AWS SDK for Ruby Amazon SQS를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업
- 서버리스 예제

작업

`ChangeMessageVisibility`

다음 코드 예시는 `ChangeMessageVisibility`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-sqs' # v2: require 'aws-sdk'
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
sqs = Aws::SQS::Client.new(region: 'us-west-2')

begin
  queue_name = 'my-queue'
  queue_url = sqs.get_queue_url(queue_name: queue_name).queue_url

  # Receive up to 10 messages
  receive_message_result_before = sqs.receive_message({
                                                        queue_url: queue_url,
                                                        max_number_of_messages: 10
                                                      })

  puts "Before attempting to change message visibility timeout: received
 #{receive_message_result_before.messages.count} message(s)."

  receive_message_result_before.messages.each do |message|
    sqs.change_message_visibility({
                                    queue_url: queue_url,
                                    receipt_handle: message.receipt_handle,
                                    visibility_timeout: 30 # This message will not
 be visible for 30 seconds after first receipt.
                                  })
  end

  # Try to retrieve the original messages after setting their visibility timeout.
  receive_message_result_after = sqs.receive_message({
                                                       queue_url: queue_url,
                                                       max_number_of_messages: 10
                                                     })
```

```
  puts "\nAfter attempting to change message visibility timeout: received
 #{receive_message_result_after.messages.count} message(s)."
rescue Aws::SQS::Errors::NonExistentQueue
  puts "Cannot receive messages for a queue named '#{queue_name}', as it does not
 exist."
end
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ChangeMessageVisibility를 참조하세요.

`CreateQueue`

다음 코드 예시는 `CreateQueue`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# 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__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 CreateQueue를 참조하세요.

`DeleteQueue`

다음 코드 예시는 `DeleteQueue`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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)
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 DeleteQueue를 참조하세요.

`ListQueues`

다음 코드 예시는 `ListQueues`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
```

- API에 대한 세부 정보는 AWS SDK for Ruby API 참조의 ListQueues를 참조하세요.

`ReceiveMessage`

다음 코드 예시는 `ReceiveMessage`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 ReceiveMessage를 참조하세요.

SendMessage

다음 코드 예시는 SendMessage의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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__
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 SendMessage를 참조하세요.

`SendMessageBatch`

다음 코드 예시는 `SendMessageBatch`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'aws-sdk-sqs'
require 'aws-sdk-sts'

#
# @param sqs_client [Aws::SQS::Client] An initialized Amazon SQS client.
# @param queue_url [String] The URL of the queue.
# @param entries [Hash] The contents of the messages to be sent,
```

```
#   in the correct format.
# @return [Boolean] true if the messages were sent; otherwise, false.
# @example
#   exit 1 unless messages_sent?(
#     Aws::SQS::Client.new(region: 'us-west-2'),
#     'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue',
#     [
#       {
#         id: 'Message1',
#         message_body: 'This is the first message.'
#       },
#       {
#         id: 'Message2',
#         message_body: 'This is the second message.'
#       }
#     ]
#   )
def messages_sent?(sqs_client, queue_url, entries)
  sqs_client.send_message_batch(
    queue_url: queue_url,
    entries: entries
  )
  true
rescue StandardError => e
  puts "Error sending messages: #{e.message}"
  false
end

# Full example call:
# Replace us-west-2 with the AWS Region you're using for Amazon SQS.
def run_me
  region = 'us-west-2'
  queue_name = 'my-queue'
  entries = [
    {
      id: 'Message1',
      message_body: 'This is the first message.'
    },
    {
      id: 'Message2',
      message_body: 'This is the second message.'
    }
  ]
```

```
  sts_client = Aws::STS::Client.new(region: region)

  # For example:
  # 'https://sqs.us-west-2.amazonaws.com/111111111111/my-queue'
  queue_url = "https://sqs.#{region}.amazonaws.com/
#{sts_client.get_caller_identity.account}/#{queue_name}"

  sqs_client = Aws::SQS::Client.new(region: region)

  puts "Sending messages to the queue named '#{queue_name}'..."

  if messages_sent?(sqs_client, queue_url, entries)
    puts 'Messages sent.'
  else
    puts 'Messages not sent.'
  end
end
```

- API에 대한 세부 정보는 AWS SDK for Ruby API 참조의 SendMessageBatch를 참조하세요.

서버리스 예제

Amazon SQS 트리거에서 간접적으로 Lambda 함수 간접 호출

다음 코드 예제는 SQS 대기열에서 메시지를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 함수는 이벤트 파라미터에서 메시지를 검색하고 각 메시지의 내용을 로깅합니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Ruby를 사용하여 Lambda로 SQS 이벤트를 사용합니다.

```
# 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
```

Amazon SQS 트리거로 Lambda 함수에 대한 배치 항목 실패 보고

다음 코드 예제는 SQS 대기열에서 이벤트를 수신하는 Lambda 함수에 대한 부분 배치 응답을 구현하는 방법을 보여줍니다. 이 함수는 응답으로 배치 항목 실패를 보고하고 나중에 해당 메시지를 다시 시도하도록 Lambda에 신호를 보냅니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Ruby를 사용하여 Lambda로 SQS 배치 항목 실패를 보고합니다.

```
# 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 SDK for Ruby를 사용한 예제

다음 코드 예제에서는를와 AWS SDK for Ruby 함께 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS STS.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- 작업

작업

`AssumeRole`

다음 코드 예시는 `AssumeRole`의 사용 방법을 보여줍니다.

SDK for Ruby

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예 리포지토리에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
 # 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
```

- API 세부 정보는 AWS SDK for Ruby API 참조의 AssumeRole을 참조하세요.

SDK for Ruby를 사용한 Amazon Textract 예제

다음 코드 예제에서는 AWS SDK for Ruby Amazon Textract에서를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시나리오

고객 피드백 분석을 위한 애플리케이션 생성

다음 코드 예제에서는 고객 의견 카드를 분석하고, 원어에서 번역하고, 감정을 파악하고, 번역된 텍스트에서 오디오 파일을 생성하는 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Ruby

이 예제 애플리케이션은 고객 피드백 카드를 분석하고 저장합니다. 특히 뉴욕시에 있는 가상 호텔의 필요를 충족합니다. 호텔은 다양한 언어의 고객들로부터 물리적인 의견 카드의 형태로 피드백을 받습니다. 피드백은 웹 클라이언트를 통해 앱에 업로드됩니다. 의견 카드의 이미지가 업로드된 후 다음 단계가 수행됩니다.

- Amazon Textract를 사용하여 이미지에서 텍스트가 추출됩니다.
- Amazon Comprehend가 추출된 텍스트와 해당 언어의 감정을 파악합니다.
- 추출된 텍스트는 Amazon Translate를 사용하여 영어로 번역됩니다.
- Amazon Polly가 추출된 텍스트에서 오디오 파일을 합성합니다.

전체 앱은 AWS CDK를 사용하여 배포할 수 있습니다. 소스 코드와 배포 지침은 GitHub의 프로젝트를 참조하세요.

이 예제에서 사용되는 서비스

- Amazon Comprehend

- Lambda
- Amazon Polly
- Amazon Textract
- Amazon Translate

SDK for Ruby를 사용한 Amazon Translate 예제

다음 코드 예제에서는 Amazon Translate에서를 사용하여 작업을 수행하고 일반적인 시나리오 AWS SDK for Ruby 를 구현하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

시나리오

고객 피드백 분석을 위한 애플리케이션 생성

다음 코드 예제에서는 고객 의견 카드를 분석하고, 원어에서 번역하고, 감정을 파악하고, 번역된 텍스트에서 오디오 파일을 생성하는 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for Ruby

이 예제 애플리케이션은 고객 피드백 카드를 분석하고 저장합니다. 특히 뉴욕시에 있는 가상 호텔의 필요를 충족합니다. 호텔은 다양한 언어의 고객들로부터 물리적인 의견 카드의 형태로 피드백을 받습니다. 피드백은 웹 클라이언트를 통해 앱에 업로드됩니다. 의견 카드의 이미지가 업로드된 후 다음 단계가 수행됩니다.

- Amazon Textract를 사용하여 이미지에서 텍스트가 추출됩니다.
- Amazon Comprehend가 추출된 텍스트와 해당 언어의 감정을 파악합니다.
- 추출된 텍스트는 Amazon Translate를 사용하여 영어로 번역됩니다.
- Amazon Polly가 추출된 텍스트에서 오디오 파일을 합성합니다.

전체 앱은 AWS CDK를 사용하여 배포할 수 있습니다. 소스 코드와 배포 지침은 GitHub의 프로젝트를 참조하세요.

이 예제에서 사용되는 서비스

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

AWS SDK for Ruby 버전 1 또는 2에서 AWS SDK for Ruby 버전 3으로 마이그레이션

이 주제에는 Ruby용 AWS SDK 버전 1 또는 2에서 버전 3으로 마이그레이션하는 데 도움이 되는 세부 정보가 포함되어 있습니다.

Side-by-side 사용

AWS SDK for Ruby 버전 1 또는 2를 버전 3으로 교체할 필요는 없습니다. 동일한 애플리케이션에서는 이 둘을 함께 사용할 수 있습니다. 자세한 내용은 이 블로그 게시물을 참조하십시오.

간단하게 예를 들면 다음과 같습니다.

```
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
```

버전 3 SDK를 사용하기 위해 기존 작업 버전 1 또는 2 코드를 다시 쓸 필요는 없습니다. 올바른 마이그레이션 전략은 버전 3 SDK에 대해서만 새 코드를 쓰는 것입니다.

일반적인 차이

버전 3은 한 가지 중요한 면에서 버전 2와 다릅니다.

- 각 서비스는 별도의 gem으로 사용할 수 있습니다.

버전 2는 여러 가지 중요한 면에서 버전 1과 다릅니다.

- `Aws` 및 `AWS` 간에 루트 네임스페이스가 다릅니다. 따라서 항목별 사용이 가능합니다.
- `Aws.config` - 이제 메서드가 아닌 vanilla Ruby 해시입니다.
- 엄격한 생성자 옵션 - 버전 1에서 클라이언트나 리소스 객체를 생성할 때 알 수 없는 생성자 옵션이 무시됩니다. 버전 2에서는 알 수 없는 생성자 옵션이 `ArgumentError`를 트리거합니다. 예제:

```
# 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
```

클라이언트 차이

버전 2와 버전 3의 클라이언트 클래스 간에 차이가 없습니다.

버전 1과 버전 2 간에 클라이언트 클래스는 외부적인 차이가 가장 적습니다. 다수의 서비스 클라이언트에는 클라이언트 생성 후에 호환되는 인터페이스가 있습니다. 일부 중요한 차이는 다음과 같습니다.

- `Aws::S3::Client` - 버전 1 Amazon S3 클라이언트 클래스는 수작업으로 코딩되었습니다. 버전 2는 서비스 모델에서 생성됩니다. 메서드 이름과 입력이 버전 2에서 크게 다릅니다.
- `Aws::EC2::Client`- 버전 2는 출력 목록에 복수형 이름을 사용하고, 버전 1은 접미사 `_set`를 사용합니다. 예제:

```
# 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` - 버전 2는 구조화된 응답을 사용하는 반면, 버전 1은 vanilla Ruby 해시를 사용합니다.
- 서비스 클래스 이름 바꾸기 - 버전 2는 여러 서비스에 대해 다음과 같이 다른 이름을 사용합니다.
  - `AWS::SimpleWorkflow`는 `Aws::SWF`가 됩니다.
  - `AWS::ELB`는 `Aws::ElasticLoadBalancing`가 됩니다.
  - `AWS::SimpleEmailService`는 `Aws::SES`가 됩니다.
- 클라이언트 구성 옵션 - 버전 1 구성 옵션 중 일부가 버전 2에서 이름이 변경되었습니다. 다른 옵션은 제거되거나 바뀌었습니다. 주된 변경 사항은 다음과 같습니다.

- `:use_ssl`이 제거되었습니다. 버전 2는 모든 위치에서 SSL을 사용합니다. SSL을 비활성화하려면 `:endpoint`를 사용하는 `http://`를 구성해야 합니다.
- `:ssl_ca_file`가 이제 `:ssl_ca_bundle`입니다.
- `:ssl_ca_path`가 이제 `:ssl_ca_directory`입니다.
- `:ssl_ca_store` 추가.
- `:endpoint`가 호스트 이름 대신 정규화된 HTTP 또는 HTTPS URI여야 합니다.
- 각 서비스에 대한 `:*_port` 옵션이 제거되고 이제 `:endpoint`로 바뀌었습니다.
- `:user_agent_prefix`가 이제 `:user_agent_suffix`입니다.

리소스 차이

버전 2와 버전 3의 리소스 인터페이스 간에 차이가 없습니다.

버전 1과 버전 2의 리소스 인터페이스 간에 큰 차이가 있습니다. 버전 1은 전적으로 수작업 코딩되었지만, 버전 2 리소스 인터페이스는 모델에서 생성됩니다. 버전 2 리소스 인터페이스는 일관성이 훨씬 더 높습니다. 몇 가지 주요 시스템적인 차이는 다음과 같습니다.

- 분리된 리소스 클래스 - 버전 2에서는 서비스 이름이 클래스가 아닌 모듈입니다. 여기서는 모듈이 리소스 인터페이스입니다.

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

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

- 참조 리소스 - 버전 2 SDK는 모음과 개별 리소스 getter를 다음과 같은 두 개의 서로 다른 메서드로 분리합니다.

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

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

- 배치 작업 - 버전 1에서는 모든 배치 작업이 수작업 코딩된 유틸리티입니다. 버전 2에서는 다수의 배치 작업이 API를 통해 자동 생성된 일괄 처리 작업입니다. 버전 2 일괄 처리 인터페이스는 버전 1과 매우 다릅니다.

AWS SDK for Ruby의 보안

Amazon Web Services(AWS)에서 가장 우선순위가 높은 것이 클라우드 보안입니다. AWS 고객으로서 여러분은 가장 높은 보안 요구 사항을 충족하기 위해 설계된 데이터 센터 및 네트워크 아키텍처의 혜택을 받게 됩니다. 보안은 AWS 와 사용자 간의 공동 책임입니다. 공동 책임 모델은 이 사항을 클라우드 내 보안 및 클라우드의 보안으로 설명합니다.

클라우드 보안 AWS - 클라우드에서 제공되는 모든 서비스를 실행하는 인프라를 보호하고 안전하게 사용할 수 있는 서비스를 AWS 제공할 책임이 있습니다. 당사의 보안 책임은에서 최우선 순위이며 AWS, 타사 감사자는 AWS 규정 준수 프로그램의 일환으로 보안의 효과를 정기적으로 테스트하고 검증합니다.

클라우드의 보안 - 사용자의 책임은 사용 AWS 서비스 중인과 데이터의 민감도, 조직의 요구 사항 및 관련 법률 및 규정을 비롯한 기타 요인에 따라 결정됩니다.

주제

- AWS SDK for Ruby의 데이터 보호
- AWS SDK for Ruby의 Identity and Access Management
- AWS SDK for Ruby에 대한 규정 준수 검증
- AWS SDK for Ruby의 복원력
- AWS SDK for Ruby의 인프라 보안
- AWS SDK for Ruby에서 최소 TLS 버전 적용
- Amazon S3 암호화 클라이언트 마이그레이션(V1에서 V2로)
- Amazon S3 암호화 클라이언트 마이그레이션(V2에서 V3로)

AWS SDK for Ruby의 데이터 보호

AWS 공동 책임 모델의 데이터 보호에 적용됩니다. 이 모델에 설명된 대로 AWS 는 모든를 실행하는 글로벌 인프라를 보호할 책임이 있습니다 AWS 클라우드. 사용자는 이 인프라에 호스팅되는 콘텐츠에 대한 통제 권한을 유지할 책임이 있습니다. 사용하는 AWS 서비스 의 보안 구성과 관리 태스크에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 관한 자세한 내용은 데이터 프라이버시 FAQ를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 AWS 공동 책임 모델 및 GDPR 블로그 게시물을 참조하세요.

데이터 보호를 위해 자격 증명을 보호하고 AWS 계정 AWS IAM Identity Center 또는 AWS Identity and Access Management (IAM)를 사용하여 개별 사용자를 설정하는 것이 좋습니다. 이렇게 하면 개별 사

용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용합니다.
- SSL/TLS를 사용하여 AWS 리소스와 통신합니다. TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- 를 사용하여 API 및 사용자 활동 로깅을 설정합니다 AWS CloudTrail. CloudTrail 추적을 사용하여 AWS 활동을 캡처하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 CloudTrail 추적 작업을 참조하세요.
- 내부의 모든 기본 보안 제어와 함께 AWS 암호화 솔루션을 사용합니다 AWS 서비스.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용합니다.
- 명령줄 인터페이스 또는 API를 AWS 통해에 액세스할 때 FIPS 140-3 검증 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 연방 정보 처리 표준(FIPS) 140-3을 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 형식 텍스트 필드에 입력하지 않는 것이 좋습니다. 여기에는 또는 기타 AWS 서비스 에서 콘솔, API AWS CLI또는 AWS SDKs를 사용하여 작업하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 자격 증명을 URL에 포함해서는 안 됩니다.

AWS SDK for Ruby의 Identity and Access Management

AWS Identity and Access Management (IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어할 수 있도록 지원하는 Amazon Web Services(AWS) 서비스입니다. IAM 관리자는 어떤 사용자가 AWS 서비스리소스를 사용할 수 있는 인증(로그인) 및 권한(권한 있음)을 받을 수 있는지 제어합니다. IAM은 추가 비용 없이 사용할 수 AWS 서비스 있는 입니다.

AWS SDK for Ruby를 사용하여에 액세스하려면 AWS 계정과 AWS 자격 증명이 AWS필요합니다. AWS 계정의 보안을 강화하려면 AWS 계정 자격 증명 대신 IAM 사용자를 사용해 액세스 자격 증명을 제공하는 것이 좋습니다.

IAM 작업에 대한 자세한 내용은 IAM을 참조하세요.

IAM 사용자에 대한 개요와 IAM 사용자가 계정 보안에 중요한 이유는 Amazon Web Services 일반 참조에서 AWS 보안 자격 증명을 참조하세요.

AWS SDK for Ruby는 지원하는 특정 Amazon Web Services(AWS) 서비스를 통해 공동 책임 모델을 따릅니다. AWS 서비스 보안 정보는 규정 준수 프로그램의 규정 준수 작업 범위에 속하는 AWS 서비스 보안 설명서 페이지 및를 참조하세요. AWS 서비스AWS

AWS SDK for Ruby에 대한 규정 준수 검증

AWS SDK for Ruby는 지원하는 특정 Amazon Web Services(AWS) 서비스를 통해 공동 책임 모델을 따릅니다. AWS 서비스 보안 정보는 규정 준수 프로그램의 규정 준수 작업 범위에 속하는 AWS 서비스 보안 설명서 페이지 및를 참조하세요. AWS 서비스AWS

Amazon Web Services(AWS) 서비스의 보안 및 규정 준수는 타사 감사자가 여러 AWS 규정 준수 프로그램의 일환으로 평가합니다. 여기에는 SOC, PCI, FedRAMP, HIPAA 등이 포함됩니다. AWS 는 규정 준수 프로그램 제공 범위 내 서비스 AWS 서비스 에서 특정 규정 준수 프로그램의 범위 내 목록을 자주 업데이트합니다. AWS

를 사용하여 서드 파티 감사 보고서를 다운로드할 수 있습니다 AWS Artifact. 자세한 내용은 AWS 아티팩트에서 보고서 다운로드를 참조하세요.

AWS 규정 준수 프로그램에 대한 자세한 내용은 규정 AWS 준수 프로그램을 참조하세요.

AWS SDK for Ruby를 사용하여 AWS 서비스 에 액세스할 때의 규정 준수 책임은 데이터의 민감도, 조직의 규정 준수 목표 및 관련 법률과 규정에 따라 결정됩니다. 를 사용할 때 HIPAA, PCI 또는 FedRAMP와 같은 표준을 준수해야 AWS 서비스 하는 경우는 도움이 되는 리소스를 AWS 제공합니다.

- 보안 및 규정 준수 빠른 시작 가이드 - 아키텍처 고려 사항을 설명하고 보안 중심 및 규정 준수 중심 기준 환경을 배포하기 위한 단계를 제공하는 배포 가이드입니다 AWS.
- HIPAA 적격 서비스 참조 - HIPAA 적격 서비스가 나열되어 있습니다. 모두 HIPAA 자격이 AWS 서비스 있는 것은 아닙니다.
- AWS 규정 준수 리소스 - 업계 및 위치에 적용될 수 있는 워크북 및 가이드 모음입니다.
- AWS Config - 리소스 구성이 내부 관행, 업계 지침 및 규정을 얼마나 잘 준수하는지 평가하는 서비스입니다.
- AWS Security Hub - 보안 업계 표준 및 모범 사례 준수 여부를 확인하는 데 도움이 AWS 되는 내 보안 상태에 대한 포괄적인 보기입니다.

AWS SDK for Ruby의 복원력

Amazon Web Services(AWS) 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다.

AWS 리전 는 물리적으로 분리되고 격리된 여러 가용 영역을 제공하며,이 가용 영역은 지연 시간이 짧고 처리량이 높으며 중복성이 높은 네트워킹과 연결됩니다.

가용 영역을 사용하면 중단 없이 가용 영역 간에 자동으로 장애 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 복수 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 리전 및 가용 영역에 대한 자세한 내용은 AWS 글로벌 인프라를 참조하세요.

AWS SDK for Ruby는 지원하는 특정 Amazon Web Services(AWS) 서비스를 통해 공동 책임 모델을 따릅니다. AWS 서비스 보안 정보는 규정 준수 프로그램의 규정 준수 작업 범위에 속하는 AWS 서비스 보안 설명서 페이지 및를 참조하세요. AWS 서비스AWS

AWS SDK for Ruby의 인프라 보안

AWS SDK for Ruby는 지원하는 특정 Amazon Web Services(AWS) 서비스를 통해 공동 책임 모델을 따릅니다. AWS 서비스 보안 정보는 규정 준수 프로그램의 규정 준수 작업 범위에 속하는 AWS 서비스 보안 설명서 페이지 및를 참조하세요. AWS 서비스AWS

AWS 보안 프로세스에 대한 자세한 내용은 AWS: 보안 프로세스 개요 백서를 참조하세요.

AWS SDK for Ruby에서 최소 TLS 버전 적용

AWS SDK for Ruby와 간의 통신 AWS 은 SSL(Secure Sockets Layer) 또는 TLS(전송 계층 보안)를 사용하여 보호됩니다. 모든 SSL 버전과 1.2 이전 버전의 TLS에는 통신 보안을 손상시킬 수 있는 취약성이 있습니다 AWS. 따라서 TLS 버전 1.2 이상을 지원하는 Ruby 버전과 함께 AWS SDK for Ruby를 사용하고 있는지 확인해야 합니다.

Ruby는 OpenSSL 라이브러리를 사용하여 HTTP 연결을 보호합니다. 시스템 패키지 관리자(`yum`, `apt` 등), 공식 설치 프로그램 또는 Ruby 관리자(rbenv, RVM 등)를 통해 설치된 지원되는 Ruby 버전(1.9.3 이상)에는 일반적으로 TLS 1.2를 지원하는 OpenSSL 1.0.1 이상이 통합되어 있습니다.

OpenSSL 1.0.1 이상에서 지원되는 Ruby 버전과 함께 사용할 경우 AWS SDK for Ruby는 TLS 1.2를 선호하며 클라이언트와 서버 모두에서 지원되는 최신 버전의 SSL 또는 TLS를 사용합니다. 이 값은 항상 TLS 1.2 이상입니다 AWS 서비스. SDK는 Ruby `Net::HTTP` 클래스를 `use_ssl=true`와 함께 사용합니다.

OpenSSL 버전 확인

Ruby 설치 시 OpenSSL 1.0.1 이상을 사용하고 있는지 확인하려면 다음 명령을 입력하십시오.

```
ruby -r openssl -e 'puts OpenSSL::OPENSSL_VERSION'
```

OpenSSL 버전을 얻는 또 다른 방법은 `openssl` 실행 파일을 직접 쿼리하는 것입니다. 먼저 다음 명령을 사용하여 적절한 실행 파일을 찾습니다.

```
ruby -r rbconfig -e 'puts RbConfig::CONFIG["configure_args"]'
```

출력에는 OpenSSL 설치 위치를 가리키는 `--with-openssl-dir=/path/to/openssl`이 포함되어야 합니다. 이 경로를 기록해 둡니다. OpenSSL 버전을 확인하려면 다음 명령을 입력합니다.

```
cd /path/to/openssl
bin/openssl version
```

모든 Ruby 설치에서 후자의 방법이 작동하지 않을 수 있습니다.

TLS 지원 업그레이드

Ruby 설치에서 사용하는 OpenSSL 버전이 1.0.1 미만인 경우 Ruby 설치 안내서에 설명된 대로 시스템 패키지 관리자, Ruby 설치 관리자 또는 Ruby 관리자를 사용하여 Ruby 또는 OpenSSL 설치를 업그레이드합니다. 소스에서 Ruby를 설치하는 경우 최신 OpenSSL을 먼저 설치한 다음 `./configure`를 실행할 때 `--with-openssl-dir=/path/to/upgraded/openssl`을 전달합니다.

Amazon S3 암호화 클라이언트 마이그레이션(V1에서 V2로)

> **Note**
>
> S3 암호화 클라이언트의 V2를 사용 중이고 V3로 마이그레이션하려는 경우 섹션을 참조하세요Amazon S3 암호화 클라이언트 마이그레이션(V2에서 V3로).

이 주제에서는 Amazon Simple Storage Service(S3) 암호화 클라이언트 버전 1(V1) 에서 버전 2(V2)로 애플리케이션을 마이그레이션하고 마이그레이션 프로세스 전반에 걸쳐 애플리케이션 가용성을 보장하는 방법을 보여줍니다.

마이그레이션 개요

이 마이그레이션은 다음 두 단계로 진행됩니다.

1. 새 형식을 읽도록 기존 클라이언트를 업데이트합니다. 먼저 SDK AWS for Ruby의 업데이트된 버전을 애플리케이션에 배포합니다. 이렇게 하면 기존 V1 암호화 클라이언트에서 새 V2 클라이언트가 작성한 객체를 해독할 수 있습니다. 애플리케이션에서 다중 AWS SDKs 사용하는 경우 각 SDK를 별도로 업그레이드해야 합니다.

2. 암호화 및 복호화 클라이언트를 V2로 마이그레이션합니다. 모든 V1 암호화 클라이언트가 새 형식을 읽을 수 있게 되면 기존 암호화 및 복호화 클라이언트를 각각의 V2 버전으로 마이그레이션할 수 있습니다.

새 형식을 읽기 위한 기존 클라이언트 업데이트

V2 암호화 클라이언트는 이전 버전의 클라이언트에서 지원하지 않는 암호화 알고리즘을 사용합니다. 마이그레이션의 첫 번째 단계는 V1 복호화 클라이언트를 최신 SDK 릴리스로 업데이트하는 것입니다. 이 단계를 완료하면 애플리케이션의 V1 클라이언트가 V2 암호화 클라이언트로 암호화된 객체를 해독할 수 있습니다. AWS SDK for Ruby의 각 메이저 버전에 대한 세부 정보는 아래를 참조하세요.

AWS SDK for Ruby 버전 3 업데이트

버전 3은 AWS SDK for Ruby의 최신 버전입니다. 이 마이그레이션을 완료하려면 `aws-sdk-s3`의 버전 1.76.0 이상을 사용해야 합니다.

명령줄에서 설치

`aws-sdk-s3` gem을 설치하는 프로젝트의 경우 버전 옵션을 사용하여 최소 버전 1.76.0이 설치되어 있는지 확인합니다.

```
gem install aws-sdk-s3 -v '>= 1.76.0'
```

Gemfile 사용

Gemfile을 사용하여 종속성을 관리하는 프로젝트의 경우 `aws-sdk-s3` gem의 최소 버전을 1.76.0으로 설정합니다. 예제:

```
gem 'aws-sdk-s3', '>= 1.76.0'
```

1. Gemfile을 수정합니다.
2. `bundle update aws-sdk-s3`을 실행합니다. 버전을 확인하려면 `bundle info aws-sdk-s3`을 실행합니다.

AWS SDK for Ruby 버전 2 업그레이드

AWS SDK for Ruby 버전 2는 2021년 11월 21일에 유지 관리 모드로 전환됩니다. 이 마이그레이션을 완료하려면 aws-sdk gem의 버전 2.11.562 이상을 사용해야 합니다.

명령줄에서 설치

aws-sdk gem을 설치하는 프로젝트의 경우 명령줄에서 버전 옵션을 사용하여 최소 버전 2.11.562가 설치되어 있는지 확인합니다.

```
gem install aws-sdk -v '>= 2.11.562'
```

Gemfile 사용

Gemfile을 사용하여 종속성을 관리하는 프로젝트의 경우 aws-sdk gem의 최소 버전을 2.11.562로 설정합니다. 예제:

```
gem 'aws-sdk', '>= 2.11.562'
```

1. Gemfile을 수정합니다. Gemfile.lock 파일이 있는 경우 해당 파일을 삭제하거나 업데이트합니다.
2. bundle update aws-sdk을 실행합니다. 버전을 확인하려면 bundle info aws-sdk을 실행합니다.

암호화 및 복호화 클라이언트를 V2로 마이그레이션

새 암호화 형식을 읽도록 클라이언트를 업데이트한 후 애플리케이션을 V2 암호화 및 복호화 클라이언트로 업데이트할 수 있습니다. 다음 단계는 V1에서 V2로 코드를 성공적으로 마이그레이션하는 방법을 보여줍니다.

V2 암호화 클라이언트를 사용하도록 코드를 업데이트하기 전에 이전 단계를 따르고 aws-sdk-s3 gem 버전 2.11.562 이상을 사용하고 있는지 확인합니다.

> **Note**
>
> AES-GCM으로 해독할 때는 해독된 데이터를 사용하기 전에 전체 객체를 끝까지 읽습니다. 이는 암호화되었던 객체이므로 객체가 수정되지 않았는지 확인하기 위함입니다.

V2 암호화 클라이언트 구성

EncryptionV2::Client에는 추가 구성이 필요합니다. 자세한 구성 정보는 EncryptionV2::Client 설명서 또는 이 주제의 뒷부분에 제공된 예제를 참조하세요.

1. 키 랩 방법 및 콘텐츠 암호화 알고리즘은 클라이언트 구성 시 지정해야 합니다. 새 `EncryptionV2::Client`를 만들 때는 `key_wrap_schema` 및 `content_encryption_schema`에 대한 값을 제공해야 합니다.

`key_wrap_schema` -를 사용하는 경우 AWS KMS로 설정해야 합니다`:kms_context`. 대칭(AES) 키를 사용하는 경우 이 값을 `:aes_gcm`으로 설정해야 합니다. 비대칭(RSA) 키를 사용하는 경우 이 값을 `:rsa_oaep_sha1`으로 설정해야 합니다.

`content_encryption_schema` - 이 값은 :aes_gcm_no_padding으로 설정해야 합니다.

2. security_profile은 클라이언트 구성 시 지정해야 합니다. 새 `EncryptionV2::Client`를 만들 때는 `security_profile`에 대한 값을 제공해야 합니다. security_profile 파라미터는 이전 V1 `Encryption::Client`를 사용하여 작성된 객체를 읽기 위한 지원을 결정합니다. :v2 및 :v2_and_legacy라는 두 값이 있습니다. 마이그레이션을 지원하려면 `security_profile`을 :v2_and_legacy로 설정합니다. :v2는 새 애플리케이션 개발에만 사용하세요.

3. AWS KMS key ID는 기본적으로 적용됩니다. V1,에서 클라이언트`Encryption::Client`를 생성하는 데 `kms_key_id` 사용된가에 제공되지 않았습니다 AWS KMS `Decrypt call`.는 메타데이터에서이 정보를 가져와 대칭 사이퍼텍스트 BLOB에 추가할 AWS KMS 수 있습니다. V2, E`ncryptionV2::Client`에서는 kms_key_id가 AWS KMS Decrypt 호출로 전달되고 객체를 암호화하는 데 사용된 키와 일치하지 않으면 호출이 실패합니다. 이전에 코드가 특정 `kms_key_id`를 설정하지 않았다면 클라이언트 생성 시 `kms_key_id: :kms_allow_decrypt_with_any_cmk`를 설정하거나 `get_object` 호출 시 `kms_allow_decrypt_with_any_cmk: true`를 설정합니다.

예: 대칭(AES) 키 사용

사전 마이그레이션

```
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)
```

마이그레이션 후

```
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
```

예: kms_key_id AWS KMS 와 함께 사용

사전 마이그레이션

```
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)
```

마이그레이션 후

```
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
```

예: kms_key_id AWS KMS 없이 사용

사전 마이그레이션

```
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)
```

마이그레이션 후

```
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
```

마이그레이션 후 대안

S2 암호화 클라이언트를 사용하여 객체를 읽고 해독만 하는 경우(작성 및 암호화하지 않음) 이 코드를 사용합니다.

```
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
```

Amazon S3 암호화 클라이언트 마이그레이션(V2에서 V3로)

> **Note**
>
> S3 암호화 클라이언트의 V1을 사용하는 경우 V3로 마이그레이션하기 전에 먼저 V2로 마이그레이션해야 합니다. V3 V1에서 V2로 마이그레이션하는 방법에 대한 지침은 Amazon S3 암호화 클라이언트 마이그레이션(V1에서 V2로) 섹션을 참조하세요.

이 주제에서는 Amazon Simple Storage Service(Amazon S3) 암호화 클라이언트 버전 2(V2)에서 버전 3(V3)으로 애플리케이션을 마이그레이션하고 마이그레이션 프로세스 전반에 걸쳐 애플리케이션 가용성을 보장하는 방법을 보여줍니다. V3는 보안을 강화하고 데이터 키 변조로부터 보호하기 위해 키 커밋 및 커밋 정책이 포함된 AES GCM을 도입합니다.

마이그레이션 개요

Amazon S3 암호화 클라이언트 버전 3에는 보안 강화를 위한 키 커밋이 포함된 AES GCM이 도입되었습니다. 이 새로운 암호화 알고리즘은 데이터 키 변조에 대한 보호를 제공하고 암호화된 데이터의 무결성을 보장합니다. V3로 마이그레이션하려면 프로세스 전반에 걸쳐 애플리케이션 가용성과 데이터 접근성을 유지하기 위한 신중한 계획이 필요합니다.

이 마이그레이션은 다음 두 단계로 진행됩니다.

1. 새 형식을 읽도록 기존 클라이언트를 업데이트합니다. 먼저 SDK AWS for Ruby의 업데이트된 버전을 애플리케이션에 배포합니다. 이렇게 하면 기존 V2 암호화 클라이언트가 새 V3 클라이언트가 작성한 객체를 복호화할 수 있습니다. 애플리케이션에서 다중 AWS SDKs 사용하는 경우 각 SDK를 별도로 업그레이드해야 합니다.

2. 암호화 및 복호화 클라이언트를 V3로 마이그레이션합니다. 모든 V2 암호화 클라이언트가 새 형식을 읽을 수 있게 되면 기존 암호화 및 복호화 클라이언트를 해당 V3 버전으로 마이그레이션할 수 있습니다. 여기에는 커밋 정책 구성 및 새 클라이언트 구성 옵션을 사용하도록 코드 업데이트가 포함됩니다.

아직 V1에서 V2로 마이그레이션하지 않은 경우 먼저 해당 마이그레이션을 완료해야 합니다. V1에서 V2로 마이그레이션하는 방법에 대한 자세한 지침은 Amazon S3 암호화 클라이언트 마이그레이션(V1에서 V2로) 섹션을 참조하세요.

V3 기능 이해

Amazon S3 암호화 클라이언트 버전 3에는 커밋 정책과 키 커밋이 포함된 AES GCM이라는 두 가지 주요 보안 기능이 도입되었습니다. 이러한 기능을 이해하는 것은 마이그레이션 전략을 계획하고 암호화된 데이터의 보안을 보장하는 데 필수적입니다.

약정 정책

커밋 정책은 암호화 및 복호화 작업 중에 암호화 클라이언트가 키 커밋을 처리하는 방법을 제어합니다. 키 커밋은 암호화된 데이터를 암호화하는 데 사용된 정확한 키로만 복호화할 수 있도록 하여 특정 유형의 암호화 공격으로부터 보호합니다.

V3 암호화 클라이언트는 세 가지 커밋 정책 옵션을 지원합니다.

`FORBID_ENCRYPT_ALLOW_DECRYPT`

이 정책은 키 커밋 없이 객체를 암호화하고 키 커밋이 있는 객체와 없는 객체의 복호화를 허용합니다.

- 암호화 동작: 객체는 V2와 동일한 알고리즘 제품군을 사용하여 키 커밋 없이 암호화됩니다.

- 복호화 동작: 키 커밋을 사용하거나 사용하지 않고 암호화된 객체를 복호화할 수 있습니다.
- 보안 영향:이 정책은 키 커밋을 적용하지 않으며 변조를 허용할 수 있습니다. 이 정책으로 암호화된 객체는 키 커밋의 향상된 보안 보호의 이점을 누릴 수 없습니다. V2 암호화 동작과의 호환성을 유지해야 하는 경우에만 마이그레이션 중에이 정책을 사용합니다.
- 버전 호환성:이 정책으로 암호화된 객체는 S3 암호화 클라이언트의 모든 V2 및 V3 구현에서 읽을 수 있습니다. V3

REQUIRE_ENCRYPT_ALLOW_DECRYPT

이 정책은 키 커밋으로 객체를 암호화하고 키 커밋이 있는 객체와 없는 객체의 복호화를 허용합니다.

- 암호화 동작: 객체는 키 커밋과 함께 AES GCM을 사용하여 키 커밋으로 암호화됩니다.
- 복호화 동작: 키 커밋을 사용하거나 사용하지 않고 암호화된 객체를 복호화하여 이전 버전과의 호환성을 제공할 수 있습니다.
- 보안 영향: 새 객체는 키 커밋 보호의 이점을 활용하지만 키 커밋이 없는 기존 객체는 계속 읽을 수 있습니다. 이를 통해 마이그레이션 중에 보안과 이전 버전과의 호환성 간에 균형을 맞출 수 있습니다.
- 버전 호환성:이 정책으로 암호화된 객체는 S3V3 암호화 클라이언트의 V3 및 최신 V2 구현에서만 읽을 수 있습니다.

REQUIRE_ENCRYPT_REQUIRE_DECRYPT

이 정책은 키 커밋으로 객체를 암호화하고 키 커밋으로 암호화된 객체의 복호화만 허용합니다.

- 암호화 동작: 객체는 키 커밋과 함께 AES GCM을 사용하여 키 커밋으로 암호화됩니다.
- 복호화 동작: 키 커밋으로 암호화된 객체만 복호화할 수 있습니다. 키 커밋 없이 객체를 복호화하려는 시도는 실패합니다.
- 보안 영향:이 정책은 모든 작업에 키 커밋을 적용하여 최고 수준의 보안을 제공합니다. 모든 객체가 키 커밋으로 다시 암호화되고 모든 클라이언트가 V3로 업그레이드된 후에만이 정책을 사용합니다.
- 버전 호환성:이 정책으로 암호화된 객체는 S3V3 암호화 클라이언트의 V3 및 최신 V2 구현에서만 읽을 수 있습니다. 또한이 정책은 V2 또는 V1 클라이언트로 암호화된 객체를 읽는 것을 방지합니다.

> **Note**
>
> 마이그레이션을 계획REQUIRE_ENCRYPT_ALLOW_DECRYPT할 때 로 시작하여 이전 버전과의 호환성을 유지하면서 새 객체에 대한 키 커밋의 보안 이점을 얻습니다. 모든 객체를 다시 암호

화하고 모든 클라이언트를 V3로 업그레이드한 REQUIRE_ENCRYPT_REQUIRE_DECRYPT 후에만 로 이동합니다.

키 커밋이 있는 AES GCM

키 커밋(ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY)이 포함된 AES GCM은 V3에 도입된 새로운 암호화 알고리즘으로, 데이터 키 변조로부터 보호하여 보안을 강화합니다. 이 알고리즘의 작동 방식과 적용 시기를 이해하는 것이 마이그레이션을 계획하는 데 중요합니다.

가 이전 알고리즘과 **ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY** 다른 방법

이전 버전의 S3 암호화 클라이언트는 명령 파일의 데이터 키를 암호화하기 위해 키 커밋 없이 AES CBC 또는 AES GCM을 사용했습니다.는 암호화 프로세스에 암호화 커밋을 ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY 추가하여 암호화된 데이터를 특정 키에 바인딩합니다. 이렇게 하면 공격자가 명령 파일의 암호화된 데이터 키를 변조하여 클라이언트가 잘못된 키로 데이터를 복호화하는 것을 방지할 수 있습니다.

키 커밋이 없으면 공격자가 명령 파일의 암호화된 데이터 키를 수정하여 다른 키로 복호화하여 무단 액세스 또는 데이터 손상을 허용할 수 있습니다.는 암호화된 데이터 키가 암호화 중에 사용된 원본 키로만 복호화할 수 있도록 하여이 공격을 ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY 방지합니다.

버전 호환성

로 암호화된 객체는 S3 암호화 클라이언트의 V3 구현과 V3 형식 읽기 지원이 포함된 V2의 특정 전환 버전에서만 해독ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY할 수 있습니다. 이 전환 지원이 없는 V2 클라이언트는 로 암호화된 지침 파일을 해독할 수 없습니다ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY.

Warning

ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY (REQUIRE_ENCRYPT_ALLOW_DECRYPT 또는 REQUIRE_ENCRYPT_REQUIRE_DECRYPT 커밋 정책을 사용하여)를 사용하여 암호화를 활성화하기 전에 암호화된 객체를 읽어야 하는 모든 클라이언트가 V3 또는 V3 형식을 지원하는 전환 버전으로 업그레이드되었는지 확인합니다. 전환 지원이 없는 V2 클라이언트가 로 암호화된 객체를 읽으려고 하면 ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY복호화가 실패합니다.

마이그레이션 중에 FORBID_ENCRYPT_ALLOW_DECRYPT 커밋 정책을 사용하여 V3 클라이언트가 키 커밋으로 암호화된 객체를 읽을 수 있도록 허용ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY하면서 없이 계속 암호화할 수 있습니다. 이렇게 하면 먼저 모든 리더를 업그레이드한 다음 키 커밋을 사용하여 암호화로 전환하는 안전한 마이그레이션 경로가 제공됩니다.

새 형식을 읽도록 기존 클라이언트를 업데이트하세요

V3 암호화 클라이언트는 V2 클라이언트가 기본적으로 지원하지 않는 암호화 알고리즘과 키 커밋 기능을 사용합니다. 마이그레이션의 첫 번째 단계는 V2 복호화 클라이언트를 V3 암호화 객체를 읽을 수 있는 AWS SDK for Ruby 버전으로 업데이트하는 것입니다. 이 단계를 완료하면 애플리케이션의 V2 클라이언트가 V3 암호화 클라이언트로 암호화된 객체를 해독할 수 있습니다.

V3 클라이언트로 암호화된 객체(REQUIRE_ENCRYPT_ALLOW_DECRYPT 또는 REQUIRE_ENCRYPT_REQUIRE_DECRYPT 커밋 정책을 사용하는 객체)를 읽으려면 aws-sdk-s3젬 버전 1.93.0 이상을 사용해야 합니다. 이 버전에는 키 커밋이 있는 AES GCM으로 암호화된 객체의 복호화에 대한 지원이 포함되어 있습니다.

명령줄에서 설치

명령줄에서 aws-sdk-s3 Gem을 설치하는 프로젝트의 경우 버전 옵션을 사용하여 최소 버전 1.208.0이 설치되었는지 확인합니다.

```
gem install aws-sdk-s3 -v '>= 1.208.0'
```

Gemfile 사용

Gemfile을 사용하여 종속성을 관리하는 프로젝트의 경우 aws-sdk-s3 Gem의 최소 버전을 1.208.0으로 설정합니다. 예제:

```
gem 'aws-sdk-s3', '>= 1.208.0'
```

1. Gemfile을 수정하여 최소 버전을 지정합니다.
2. 를 실행bundle update aws-sdk-s3하여 Gem을 업데이트합니다.
3. 버전을 확인하려면 bundle info aws-sdk-s3을 실행합니다.

> **Note**
>
> 최신 버전으로 업데이트한 후 기존 V2 암호화 클라이언트는 V3 클라이언트로 암호화된 객체를 복호화할 수 있습니다. 그러나 다음 섹션에 설명된 대로 V3로 마이그레이션할 때까지 V2 알고리즘을 사용하여 새 객체를 계속 암호화합니다.

암호화 및 복호화 클라이언트를 V3로 마이그레이션

새 암호화 형식을 읽도록 클라이언트를 업데이트한 후 애플리케이션을 V3 암호화 및 복호화 클라이언트로 업데이트할 수 있습니다. 다음 단계에서는 V2에서 V3로 코드를 성공적으로 마이그레이션하는 방법을 보여줍니다.

V3 암호화 클라이언트를 사용하도록 코드를 업데이트하기 전에 이전 단계를 따르고 `aws-sdk-s3` Gem 버전 1.93.0 이상을 사용하고 있는지 확인합니다.

> **Note**
>
> AES-GCM으로 해독할 때는 해독된 데이터를 사용하기 전에 전체 객체를 끝까지 읽습니다. 이는 객체가 암호화된 이후 수정되지 않았는지 확인하기 위한 것입니다.

V3 클라이언트 구성

V3 암호화 클라이언트에는 키 커밋 동작과 이전 버전과의 호환성을 제어하는 새로운 구성 옵션이 도입되었습니다. 성공적인 마이그레이션을 위해서는 이러한 옵션을 이해하는 것이 필수적입니다.

commitment_policy

`commitment_policy` 파라미터는 암호화 및 복호화 작업 중에 암호화 클라이언트가 키 커밋을 처리하는 방법을 제어합니다. 이는 V3 클라이언트에 가장 중요한 구성 옵션입니다.

- `:require_encrypt_allow_decrypt` - 키 커밋으로 새 객체를 암호화하고 키 커밋 유무에 관계없이 객체의 복호화를 허용합니다. 이는 기존 V2 객체와의 이전 버전 호환성을 유지하면서 새 객체에 대한 보안을 강화하므로 마이그레이션에 권장되는 설정입니다.
- `:forbid_encrypt_allow_decrypt` - 키 커밋 없이 새 객체를 암호화하고(V2 알고리즘 사용) 키 커밋 유무에 관계없이 객체의 복호화를 허용합니다. 일부 클라이언트가 아직 V3 암호화 객체를 읽을 수 없는 경우와 같이 마이그레이션 중에 V2 암호화 동작을 유지해야 하는 경우에만이 설정을 사용합니다. V3

- `:require_encrypt_require_decrypt` - 키 커밋으로 새 객체를 암호화하고 키 커밋으로 암호화된 객체의 복호화만 허용합니다. 모든 객체를 키 커밋으로 다시 암호화하고 모든 클라이언트를 V3로 업그레이드한 후에만이 설정을 사용합니다.

security_profile

`security_profile` 파라미터는 이전 암호화 클라이언트 버전에서 작성된 객체 읽기에 대한 지원을 결정합니다. 이 파라미터는 마이그레이션 중에 이전 버전과의 호환성을 유지하는 데 필수적입니다.

- `:v3_and_legacy` - V3 클라이언트가 V1 및 V2 암호화 클라이언트로 암호화된 객체를 복호화할 수 있습니다. 마이그레이션 중에이 설정을 사용하여 V3 클라이언트가 기존의 모든 암호화된 객체를 읽을 수 있도록 합니다.
- `:v3` - V3 클라이언트가 V2 암호화 클라이언트로 암호화된 객체만 복호화하도록 허용합니다. 이미 모든 V1 객체를 V2 형식으로 마이그레이션한 경우이 설정을 사용합니다.
- 지정하지 않으면 클라이언트는 V3 클라이언트로 암호화된 객체만 해독합니다. 레거시 객체가 없는 새 애플리케이션 개발에만 사용합니다.

envelope_location

`envelope_location` 파라미터는 암호화 메타데이터(암호화된 데이터 키 포함)가 저장되는 위치를 결정합니다. 이 파라미터는 키 커밋이 있는 AES GCM으로 보호되는 객체에 영향을 줍니다.

- `:metadata` (기본값) - S3 객체의 메타데이터 헤더에 암호화 메타데이터를 저장합니다. 이는 기본 동작이며 대부분의 사용 사례에 권장됩니다. 메타데이터 스토리지를 사용하는 경우 키 커밋이 있는 AES GCM은 적용되지 않습니다.
- `:instruction_file` - 구성 가능한 접미사가 있는 별도의 S3 객체(지침 파일)에 암호화 메타데이터를 저장합니다. 지침 파일을 사용할 때 키 커밋이 있는 AES GCM은 암호화된 데이터 키가 변조되지 않도록 보호합니다. 데이터 키 자체에 대한 키 커밋에서 제공하는 추가 보안이 필요한 경우이 설정을 사용합니다.

를 사용할 때 선택적으로 `instruction_file_suffix` 파라미터를 지정하여 명령 파일 객체에 사용되는 접미사를 사용자 `:instruction_file`지정할 수 있습니다. 기본 접미사는 입니다`.instruction`.

각 구성 옵션을 사용해야 하는 경우

마이그레이션 중에 다음 권장 구성 전략을 따릅니다.

1. 초기 마이그레이션: commitment_policy: :require_encrypt_allow_decrypt 및를 설정합니다security_profile: :v3_and_legacy. 이렇게 하면 V3 클라이언트가 키 커밋으로 새 객체를 암호화하는 동시에 기존 V1 및 V2 객체를 모두 복호화할 수 있습니다.
2. 모든 클라이언트가 업그레이드된 후: 키 커밋 보호가 필요한 모든 객체를 다시 암호화할 security_profile: :v3_and_legacy 때까지 commitment_policy: :require_encrypt_allow_decrypt 및를 계속 사용합니다.
3. 전체 V3 적용: 모든 객체를 키 커밋으로 다시 암호화하고 더 이상 V1/V2 객체를 읽을 필요가 없는 경우에만 선택적으로 security_profile 파라미터를 전환commitment_policy: :require_encrypt_require_decrypt하고 제거할 수 있습니다(또는 V2 객체가 여전히 존재하는 :v2 경우 로 설정).

의 경우 변경할 특별한 이유가 없는 한 기존 스토리지 방법(:metadata 또는 :instruction_file)을 envelope_location계속 사용합니다. 현재 메타데이터 스토리지를 사용하고 있고 데이터 키에 대한 키 커밋이 있는 AES GCM의 추가 보안을 원하는 경우 로 전환할 수 있지만:instruction_file, 이렇게 하려면 이러한 객체를 읽는 모든 클라이언트를 업데이트해야 합니다.

암호화 및 복호화 클라이언트를 V3로 마이그레이션

새 암호화 형식을 읽도록 클라이언트를 업데이트한 후 애플리케이션을 V3 암호화 및 복호화 클라이언트로 업데이트할 수 있습니다. 다음 예제에서는 V2에서 V3로 코드를 성공적으로 마이그레이션하는 방법을 보여줍니다.

V3 암호화 클라이언트 사용

마이그레이션 전(V2)

```
require 'aws-sdk-s3'

# Create V2 encryption client with KMS
client = Aws::S3::EncryptionV2::Client.new(
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context,
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v2_and_legacy,
  commitment_policy: :forbid_encrypt_allow_decrypt
)

# Encrypt and upload object
```

```
client.put_object(bucket: 'my-bucket', key: 'my-object', body: 'secret data')

# Download and decrypt object
resp = client.get_object(bucket: 'my-bucket', key: 'my-object')
decrypted_data = resp.body.read
```

마이그레이션 중(이전 버전과의 호환성을 갖춘 V3)

```
require 'aws-sdk-s3'

# Create V3 encryption client with KMS
client = Aws::S3::EncryptionV3::Client.new(
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context,
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v3_and_legacy,
  commitment_policy: :require_encrypt_allow_decrypt
)

# Encrypt and upload object
client.put_object(bucket: 'my-bucket', key: 'my-object', body: 'secret data')

# Download and decrypt object
resp = client.get_object(bucket: 'my-bucket', key: 'my-object')
decrypted_data = resp.body.read
```

마이그레이션 후(V3)

```
require 'aws-sdk-s3'

# Create V3 encryption client with KMS
client = Aws::S3::EncryptionV3::Client.new(
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context,
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v3,
  # Use the commitment policy (REQUIRE_ENCRYPT_REQUIRE_DECRYPT)
  # This encrypts with key commitment and does not decrypt V2 objects
  commitment_policy: :require_encrypt_require_decrypt
)

# Encrypt and upload object
client.put_object(bucket: 'my-bucket', key: 'my-object', body: 'secret data')
```

```
# Download and decrypt object
resp = client.get_object(bucket: 'my-bucket', key: 'my-object')
decrypted_data = resp.body.read
```

V3의 주요 차이점은 commitment_policy 파라미터를 추가하는 것입니다. 로 설정:require_encrypt_require_decrypt하면 새 객체가 키 커밋으로 암호화되고 클라이언트가 키 커밋으로 암호화된 객체만 복호화하여 데이터 키 변조에 대한 보안을 강화할 수 있습니다.

put_object 호출 자체는 변경되지 않습니다. 모든 보안 개선 사항은 클라이언트 수준에서 구성됩니다.

추가 예제

이 섹션에서는 V2에서 V3로 마이그레이션하는 동안 유용할 수 있는 특정 마이그레이션 시나리오 및 구성 옵션에 대한 추가 예제를 제공합니다.

명령 파일과 메타데이터 스토리지 비교

S3 암호화 클라이언트는 암호화 메타데이터(암호화된 데이터 키 포함)를 S3 객체의 메타데이터 헤더 또는 별도의 명령 파일이라는 두 위치에 저장할 수 있습니다. 스토리지 방법 선택은 키 커밋 보호를 통해 AES GCM의 이점을 활용하는 객체에 영향을 미칩니다.

메타데이터 스토리지(기본값)

기본적으로 암호화 클라이언트는 암호화 메타데이터를 S3 객체의 메타데이터 헤더에 저장합니다. 이는 객체와 함께 암호화 메타데이터를 유지하고 별도의 명령 파일 객체를 관리할 필요가 없으므로 대부분의 사용 사례에 권장되는 접근 방식입니다.

```
require 'aws-sdk-s3'

# Create V3 encryption client with metadata storage (default)
client = Aws::S3::EncryptionV3::Client.new(
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context,
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v3_and_legacy,
  commitment_policy: :require_encrypt_allow_decrypt,
  envelope_location: :metadata # Explicitly set to metadata (this is the default)
)

# Encrypt and upload object
```

```
# Encryption metadata is stored in the object's metadata headers
client.put_object(bucket: 'my-bucket', key: 'my-object',body: 'secret data')
```

메타데이터 스토리지를 사용하는 경우 키 커밋이 있는 AES GCM은
암호화된 데이터 키에 적용되지 않습니다. 그러나 콘텐츠 암호화는
`commitment_policy: :require_encrypt_allow_decrypt` 또는를 사용할 때 키 커밋의 이점을 여전히 누릴 수 있습니다`:require_encrypt_require_decrypt`.

지침 파일 스토리지

또는 명령 파일이라는 별도의 S3 객체에 암호화 메타데이터를 저장하도록 암호화 클라이언트를 구성할 수 있습니다. V3와 함께 지침 파일을 사용하는 경우 암호화된 데이터 키는 키 커밋이 있는 AES GCM으로 보호되므로 데이터 키 변조에 대한 추가 보안을 제공합니다.

```
require 'aws-sdk-s3'

# Create V3 encryption client with instruction file storage
client = Aws::S3::EncryptionV3::Client.new(
  kms_key_id: kms_key_id,
  key_wrap_schema: :kms_context,
  content_encryption_schema: :aes_gcm_no_padding,
  security_profile: :v3_and_legacy,
  commitment_policy: :require_encrypt_allow_decrypt,
  envelope_location: :instruction_file, # Store metadata in separate instruction file
  instruction_file_suffix: '.instruction' # Optional: customize the suffix (default is
 '.instruction')
)

# Encrypt and upload object
# Encryption metadata is stored in a separate object: 'my-object.instruction'
client.put_object(bucket: 'my-bucket', key: 'my-object', body: 'secret data')

# When retrieving the object, the client automatically reads the instruction file
resp = client.get_object(bucket: 'my-bucket', key: 'my-object')
decrypted_data = resp.body.read
```

`envelope_location: :instruction_file`를 사용할 때 암호화 클라이언트는 두 개의 S3 객체를 생성합니다.

1. 암호화된 데이터 객체(예: `my-object`)
2. 암호화 메타데이터가 포함된 명령 파일(예: `my-object.instruction`)

`instruction_file_suffix` 파라미터를 사용하면 명령 파일에 사용되는 접미사를 사용자 지정할 수 있습니다. 기본값은 `.instruction`입니다.

각 스토리지 방법을 사용해야 하는 경우

- 대부분의 시나리오에 메타데이터 스토리지를 사용합니다. 암호화 메타데이터가 객체와 함께 이동하므로 객체 관리를 간소화합니다.
- 객체 메타데이터 크기가 문제가 되거나 암호화된 객체와 암호화 메타데이터를 분리해야 하는 경우 명령 파일 스토리지를 사용합니다. 지침 파일을 사용하려면 S3 객체(암호화된 객체와 해당 명령 파일)를 하나 대신 관리해야 합니다.

> **Warning**
>
> 메타데이터 스토리지에서 명령 파일 스토리지로(또는 그 반대로) 변경하는 경우 새 스토리지 방법으로 구성된 클라이언트는 이전 스토리지 방법으로 암호화된 기존 객체를 읽을 수 없습니다. 스토리지 방법을 신중하게 계획하고 애플리케이션 전반에서 일관성을 유지합니다.

문서 기록

아래 표에 이 안내서의 중요한 변경 사항이 설명되어 있습니다. 이 설명서에 대한 업데이트 알림을 받으려면 RSS feed를 구독하세요.

변경 사항	설명	날짜
콘텐츠 재구성	다른 AWS SDKs.	2025년 3월 29일
Observability	Ruby 관찰성에 대한 정보를 추가했습니다.	2025년 1월 24일
일반 업데이트	최소 필수 Ruby 버전을 v2.5로 업데이트했습니다. 리소스 링크를 업데이트했습니다.	2024년 11월 13일
목차 및 안내 예제	보다 포괄적인 코드 예제 리포지토리를 참조하도록 안내 예제를 제거했습니다.	2024년 7월 10일
목차	코드 예제에 더 쉽게 접근할 수 있도록 목차를 업데이트했습니다.	2023년 6월 1일
IAM 모범 사례 업데이트	IAM 모범 사례에 따라 가이드가 업데이트되었습니다. 자세한 내용은 IAM의 보안 모범 사례를 참조하세요. 시작하기에 업데이트합니다.	2023년 5월 8일
일반 업데이트	관련 외부 리소스의 시작 페이지 업데이트. v2.3에 필요한 최소 Ruby 버전도 업데이트되었습니다. 용어 업데이트를 반영하도록 AWS Key Management Service 섹션을 업데이트했습니다. 명확성을 위해 REPL 유	2022년 8월 8일

	틸리티의 사용 정보가 업데이트되었습니다.	
끊어진 링크 수정	끊어진 예제 링크를 수정했습니다. 중복된 팁 및 요령 페이지를 제거했습니다. Amazon EC2 예제 콘텐츠로 리디렉션되었습니다. 코드 예제 리포지토리의 GitHub에서 사용할 수 있는 코드 예제 목록이 포함되었습니다.	2022년 8월 3일
SDK 지표	더 이상 사용되지 않는 Enterprise Support용 SDK 지표 사용에 대한 정보를 제거했습니다.	2022년 1월 28일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.