

개발자 가이드

# AWS SDK for C++



# AWS SDK for C++: 개발자 가이드

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

Amazon 상표 및 트레이드 드레스는 Amazon 외 제품 또는 서비스와 함께 사용되어서는 안되며, 고객에게 혼동을 일으키거나 Amazon 브랜드 이미지를 떨어뜨리고 폄하하는 방식으로 이용할 수 없습니다. Amazon이 소유하지 않은 기타 모든 상표는 Amazon 계열사, 관련 업체 또는 Amazon의 지원 업체 여부에 상관없이 해당 소유자의 자산입니다.

# Table of Contents

|                                                    |    |
|----------------------------------------------------|----|
| AWS SDK for C++ 개발자 안내서 소개 .....                   | 1  |
| 추가 설명서 및 리소스 .....                                 | 1  |
| SDK 메이저 버전에 대한 유지 관리 및 지원 .....                    | 2  |
| 시작하기 .....                                         | 3  |
| 를 사용하여 인증 AWS .....                                | 3  |
| 콘솔 자격 증명 사용 .....                                  | 3  |
| IAM Identity Center 사용 .....                       | 4  |
| 세부 인증 정보 .....                                     | 5  |
| 소스에서 SDK 가져오기 .....                                | 6  |
| Windows에서 빌드 .....                                 | 7  |
| Linux/macOS에서 빌드 .....                             | 11 |
| 간단한 애플리케이션 생성 .....                                | 15 |
| 패키지 관리자에서 SDK 가져오기 .....                           | 20 |
| 사전 조건 .....                                        | 7  |
| vcpkg를 사용하여 SDK 가져오기 .....                         | 21 |
| 빌드 문제 해결 .....                                     | 22 |
| CMake 오류: "AWSSDK"에서 제공하는 패키지 구성 파일을 찾을 수 없음 ..... | 23 |
| CMake 오류: 로드 파일을 찾을 수 없음(SDK 버전 1.8 사용 중) .....    | 23 |
| CMake 오류: 로드 파일을 찾을 수 없음 .....                     | 24 |
| 런타임 오류: aws-*.dll을 찾을 수 없어 진행할 수 없음 .....          | 25 |
| 서비스 클라이언트 구성 .....                                 | 26 |
| SDK 구성 .....                                       | 27 |
| 외부에서 클라이언트 구성 .....                                | 28 |
| 코드 내 클라이언트 구성 .....                                | 29 |
| 선언 .....                                           | 30 |
| 설명 .....                                           | 32 |
| AWS 리전 .....                                       | 38 |
| 보안 인증 공급자 .....                                    | 38 |
| 자격 증명 공급자 체인 .....                                 | 39 |
| 명시적 자격 증명 공급자 .....                                | 41 |
| ID 캐싱 .....                                        | 42 |
| CMake 파라미터 .....                                   | 42 |
| 일반 CMake 변수 및 옵션 .....                             | 42 |
| Android CMake 변수 및 옵션 .....                        | 57 |

|                                                |     |
|------------------------------------------------|-----|
| 로깅 .....                                       | 60  |
| HTTP .....                                     | 64  |
| HttpClient 및 AWSClient에서 사용하는 입출력 스트림 제어 ..... | 64  |
| 사용자 지정 libcrypto 사용 .....                      | 65  |
| SDK for C++에 사용자 지정 libcrypto를 빌드하는 방법 .....   | 66  |
| Docker 이미지에 모두 통합 .....                        | 67  |
| SDK 사용 .....                                   | 70  |
| SDK 초기화 및 종료 .....                             | 70  |
| AWS 서비스 요청하기 .....                             | 71  |
| 비동기 프로그래밍 .....                                | 71  |
| 비동기식 SDK 메서드 .....                             | 71  |
| SDK 비동기식 메서드 호출 .....                          | 72  |
| 비동기 작업 완료 알림 .....                             | 74  |
| 유틸리티 모듈 .....                                  | 76  |
| HTTP 스택 .....                                  | 76  |
| 문자열 유틸리티 .....                                 | 77  |
| 해싱 유틸리티 .....                                  | 77  |
| JSON 파서 .....                                  | 77  |
| XML 파서 .....                                   | 77  |
| 메모리 관리 .....                                   | 78  |
| 메모리 할당 및 할당 해제 .....                           | 78  |
| STL 및 AWS 문자열과 벡터 .....                        | 79  |
| 남아 있는 문제 .....                                 | 80  |
| 네이티브 SDK 개발자 및 메모리 제어 .....                    | 81  |
| 오류 처리 .....                                    | 81  |
| AWS 서비스 호출 .....                               | 83  |
| 코드 예제 시작하기 .....                               | 83  |
| 코드 예제의 구조 .....                                | 83  |
| Visual Studio에서 코드 예제 빌드 및 디버깅 .....           | 84  |
| 런타임 오류 해결 시작하기 .....                           | 86  |
| 안내 예제 .....                                    | 89  |
| Amazon CloudWatch 예제 .....                     | 90  |
| Amazon DynamoDB 예제 .....                       | 105 |
| Amazon EC2 예제 .....                            | 121 |
| Amazon S3 예제 .....                             | 146 |
| Amazon SQS 예제 .....                            | 179 |

|                                |     |
|--------------------------------|-----|
| 코드 예제 .....                    | 195 |
| ACM .....                      | 196 |
| 작업 .....                       | 196 |
| API Gateway .....              | 226 |
| 시나리오 .....                     | 226 |
| Aurora .....                   | 227 |
| 시작하기 .....                     | 227 |
| 기본 사항 .....                    | 230 |
| 작업 .....                       | 196 |
| 시나리오 .....                     | 226 |
| Auto Scaling .....             | 271 |
| 시작 .....                       | 227 |
| 기본 사항 .....                    | 230 |
| 작업 .....                       | 196 |
| CloudTrail .....               | 301 |
| 작업 .....                       | 196 |
| CloudWatch .....               | 306 |
| 작업 .....                       | 196 |
| CloudWatch Logs .....          | 317 |
| 작업 .....                       | 196 |
| CodeBuild .....                | 321 |
| 작업 .....                       | 196 |
| Amazon Cognito 자격 증명 공급자 ..... | 326 |
| 시작하기 .....                     | 227 |
| 작업 .....                       | 196 |
| 시나리오 .....                     | 226 |
| DynamoDB .....                 | 349 |
| 시작하기 .....                     | 227 |
| 기본 사항 .....                    | 230 |
| 작업 .....                       | 196 |
| 시나리오 .....                     | 226 |
| Amazon EC2 .....               | 428 |
| 시작하기 .....                     | 227 |
| 작업 .....                       | 196 |
| EventBridge .....              | 462 |
| 작업 .....                       | 196 |

|                          |     |
|--------------------------|-----|
| AWS Glue .....           | 466 |
| 시작하기 .....               | 227 |
| 기본 사항 .....              | 230 |
| 작업 .....                 | 196 |
| HealthImaging .....      | 505 |
| 시작하기 .....               | 227 |
| 작업 .....                 | 196 |
| 시나리오 .....               | 226 |
| IAM .....                | 539 |
| 시작하기 .....               | 227 |
| 기본 사항 .....              | 230 |
| 작업 .....                 | 196 |
| AWS IoT .....            | 581 |
| 시작하기 .....               | 227 |
| 기본 사항 .....              | 230 |
| 작업 .....                 | 196 |
| AWS IoT data .....       | 623 |
| 작업 .....                 | 196 |
| Lambda .....             | 625 |
| 시작하기 .....               | 227 |
| 기본 사항 .....              | 230 |
| 작업 .....                 | 196 |
| 시나리오 .....               | 226 |
| MediaConvert .....       | 650 |
| 작업 .....                 | 196 |
| Amazon RDS .....         | 658 |
| 시작하기 .....               | 227 |
| 기본 사항 .....              | 230 |
| 작업 .....                 | 196 |
| 시나리오 .....               | 226 |
| Amazon RDS 데이터 서비스 ..... | 695 |
| 시나리오 .....               | 226 |
| Amazon Rekognition ..... | 696 |
| 시작하기 .....               | 227 |
| 작업 .....                 | 196 |
| 시나리오 .....               | 226 |

|                                           |     |
|-------------------------------------------|-----|
| Amazon S3 .....                           | 702 |
| 시작하기 .....                                | 227 |
| 기본 사항 .....                               | 230 |
| 작업 .....                                  | 196 |
| 시나리오 .....                                | 226 |
| Secrets Manager .....                     | 784 |
| 작업 .....                                  | 196 |
| Amazon SES .....                          | 786 |
| 작업 .....                                  | 196 |
| 시나리오 .....                                | 226 |
| Amazon SNS .....                          | 808 |
| 시작하기 .....                                | 227 |
| 작업 .....                                  | 196 |
| 시나리오 .....                                | 226 |
| Amazon SQS .....                          | 850 |
| 시작하기 .....                                | 227 |
| 작업 .....                                  | 196 |
| 시나리오 .....                                | 226 |
| AWS STS .....                             | 886 |
| 작업 .....                                  | 196 |
| Amazon Transcribe Streaming .....         | 888 |
| 작업 .....                                  | 196 |
| 시나리오 .....                                | 226 |
| 보안 .....                                  | 896 |
| 데이터 보호 .....                              | 896 |
| ID 및 액세스 관리 .....                         | 897 |
| 고객 .....                                  | 898 |
| 보안 인증을 통한 인증 .....                        | 898 |
| 정책을 사용하여 액세스 관리 .....                     | 900 |
| AWS 서비스에서 IAM을 사용하는 방식 .....              | 901 |
| AWS 보안 인증 및 액세스 문제 해결 .....               | 901 |
| 규정 준수 검증 .....                            | 903 |
| 복원성 .....                                 | 904 |
| 인프라 보안 .....                              | 904 |
| 최소 TLS 버전 적용 .....                        | 905 |
| 모든 플랫폼에서 libcurl을 사용하여 특정 TLS 버전 적용 ..... | 905 |

|                                            |        |
|--------------------------------------------|--------|
| Windows에서 특정 TLS 버전 적용 .....               | 906    |
| Amazon S3 암호화 클라이언트 마이그레이션(V1에서 V2로) ..... | 909    |
| 마이그레이션 개요 .....                            | 909    |
| 새 형식을 읽기 위한 기존 클라이언트 업데이트 .....            | 910    |
| 암호화 및 복호화 클라이언트를 V2로 마이그레이션 .....          | 911    |
| 추가 예제 .....                                | 913    |
| Amazon S3 암호화 클라이언트 마이그레이션(V2에서 V3로) ..... | 916    |
| 마이그레이션 개요 .....                            | 916    |
| V3 개념 이해 .....                             | 916    |
| 새 형식을 읽도록 기존 클라이언트를 업데이트하세요 .....          | 918    |
| 암호화 및 복호화 클라이언트를 V3로 마이그레이션 .....          | 920    |
| 추가 예제 .....                                | 923    |
| 문서 이력 .....                                | 926    |
| .....                                      | cmxxix |

# AWS SDK for C++ 개발자 안내서 소개

AWS SDK for C++ 개발자 안내서에 오신 것을 환영합니다.

AWS SDK for C++는 Amazon Web Services(AWS)를 위한 최신 C++(버전 C++ 11 이상) 인터페이스를 제공합니다. 거의 모든 AWS 기능에 대해 고수준 및 저수준 API를 모두 제공하여 종속성을 최소화하고 Windows, macOS, Linux 및 모바일에서 플랫폼 이식성을 제공합니다.

## [AWS SDK for C++ 시작하기](#)

### Note

AWS IoT SDK 및 `aws-iot-device-sdk-cpp`는 이 SDK와 별개입니다. AWS IoT Device SDK for C++ v2는 GitHub의 [aws-iot-device-sdk-cpp-v2](#)에서 제공합니다. AWS IoT에 대한 자세한 내용은 AWS IoT 개발자 안내서에서 [AWS IoT란 무엇입니까?](#)를 참조하세요.

## 추가 설명서 및 리소스

이 설명서 외에도, 다음은 AWS SDK for C++ 개발자를 위한 귀중한 온라인 리소스입니다.

- [AWS SDK 및 도구 참조 안내서](#): AWS SDK에 공통적으로 적용되는 설정, 기능 및 기타 기본 개념이 포함되어 있습니다.
- GitHub:
  - [SDK 소스](#)
  - [SDK 문제](#)
- [AWS SDK for C++ API 레퍼런스](#)
- [AWS C++ 개발자 블로그](#)
- [AWS 코드 샘플 카탈로그](#)
- [SDK 라이선스](#)
- 동영상: [Introducing the AWS SDK for C++ from AWS re:invent 2015](#)

# 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 안내서](#)에서 다음 내용을 참조하세요.

- [AWS SDK 및 도구 유지 관리 정책](#)
- [AWS SDK 및 도구 버전 지원 매트릭스](#)

# AWS SDK for C++ 시작하기

AWS SDK for C++는 Amazon Web Services에 연결하는 데 사용할 수 있는 모듈화된 플랫폼 간 오픈 소스 라이브러리입니다.

AWS SDK for C++는 [CMake](#)를 사용하여 비디오 게임, 시스템, 모바일 및 임베디드 디바이스를 비롯한 여러 도메인에서 다양한 플랫폼을 지원합니다. CMake는 애플리케이션의 종속성을 관리하고 빌드 중인 플랫폼에 적합한 makefile을 생성하는 데 사용할 수 있는 빌드 도구입니다. CMake는 플랫폼 또는 애플리케이션에 사용되지 않는 빌드 부분을 제거합니다.

코드를 실행하여 AWS 리소스에 액세스하기 전에 코드의 AWS 인증 방식을 설정해야 합니다.

- [AWS SDK for C++를 AWS 사용하여 로 인증](#)

코드에서 AWS SDK for C++를 사용하려면 SDK 소스를 직접 빌드하거나 패키지 관리자를 사용하여 SDK 실행 파일을 얻습니다.

- [소스 코드에서 AWS SDK for C++ 가져오기](#)
- [패키지 관리자에서 AWS SDK for C++ 가져오기](#)

CMake와 관련된 빌드 문제가 발생하는 경우 [AWS SDK for C++ 빌드 문제 해결](#) 섹션을 참조하세요.

## AWS SDK for C++를 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. 모든 주요 인증 방법에 대한 선택 사항과 SDK용으로 구성하는 방법에 대한 지침은 AWS SDK 및 도구 참조 안내서에서 [인증 및 액세스](#)를 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 흐름 후는 AWS CLI 및 AWS SDKs AWS Tools for PowerShell 와 같은 로컬 개발 도구에서 작동하는 임시 자격 증명을 AWS 생성합니다. 이 기능은 특히 장기 액세스 키를 관리하는 것보다 대화형 인증을 선호하는 경우 AWS CLI 자격 증명을 구성하고 관리하는 프로세스를 간소화합니다.

이 방법을 선택하는 경우 지침에 따라 AWS CLI를 사용하여 콘솔 자격 증명으로 로그인합니다. 자세한 내용은 [콘솔 자격 증명을 사용하여 AWS 로컬 개발을 위해 로그인](#)을 참조하세요.

AWS CLI로 설정하면 [기본 자격 증명 공급자 체인](#)이 AWS CLI에서 캐시한 로그인 토큰을 사용하여 요청을 자동으로 시작합니다.

## IAM Identity Center 사용

이 방법에는 구성이 쉽고 AWS 액세스 포털 AWS CLI에 정기적으로 로그인하기 위한 설치가 포함됩니다.

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 CLI를 사용하는입니다.
- SDK에서 참조할 수 있는 구성 값 세트가 포함된 [default] 프로필이 있는 [shared AWSconfig file](#)입니다. 이 파일의 위치를 찾으려면 AWS SDK 및 도구 참조 가이드에서 [공유 파일의 위치](#)를 참조하세요.
- 공유 config 파일은 [region](#) 설정을 지정합니다. 이렇게 하면 SDK AWS 리전가 AWS 요청에 사용하는 기본값이 설정됩니다. 이 리전은 사용할 리전이 지정되지 않은 SDK 서비스 요청에 사용됩니다.
- SDK는 AWS에 요청을 보내기 전에 프로필의 [SSO token provider configuration](#)을 사용하여 보안 인증을 얻습니다. IAM Identity Center 권한 세트에 연결된 IAM 역할인 sso\_role\_name 값은 애플리케이션에 AWS 서비스 사용되는데 대한 액세스를 허용해야 합니다.

다음 샘플 config 파일은 SSO 토큰 공급자 구성으로 설정된 기본 프로필을 보여줍니다. 프로필의 sso\_session 설정은 이름이 지정된 [sso-session section](#)을 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 C++ 는 IAM Identity Center 인증을 사용하기 위해 애플리케이션에 추가 패키지(예: SSO 및 SS00IDC)를 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 서비스 필요합니다. 구성된 세션 길이에 따라 결국 액세스가 만료되고 SDK에 인증 오류가 발생합니다. AWS 액세스 포털에 로그인하려면에서 다음 명령을 실행합니다 AWS CLI.

```
aws sso login
```

기본 프로필이 설정되어 있으므로 `--profile` 옵션으로 명령을 호출할 필요가 없습니다. SSO 토큰 공급자 구성에서 명명된 프로필을 사용하는 경우 `aws sso login --profile named-profile` 명령을 사용합니다.

이미 활성 세션이 있는지 테스트하려면 다음 AWS CLI 명령을 실행합니다.

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

이 명령에 대한 응답은 공유 config 파일에 구성된 IAM Identity Center 계정 및 권한 집합을 보고해야 합니다.

### 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 증명에 있어야 합니다. 조직의 구성원인 인간 사용자는 작업 인력 ID라고도 하며, 이는 개발자인 사용자를 의미합니다. 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 집중식 액세스 관리를 위해 AWS IAM Identity Center (IAM Identity Center)를 사용하여 계정에 대한 액세스와 해당 계정 내의 권한을 관리하는 것이 좋습니다. 더 많은 대안을 보려면 다음을 참조하세요.

- 모범 사례에 대해 자세히 알아보려면 IAM 사용 설명서에서 [IAM의 보안 모범 사례](#)를 참조하세요.
- 단기 AWS 자격 증명을 생성하려면 IAM 사용 설명서의 [임시 보안 자격 증명을 참조하세요](#).
- 다른 AWS SDK for C++ 자격 증명 공급자에 대한 자세한 내용은 SDK 및 도구 참조 안내서의 [표준화된 자격 증명 공급자](#)를 참조하세요. AWS SDKs

## 소스 코드에서 AWS SDK for C++ 가져오기

먼저 소스에서 SDK를 빌드한 다음 로컬에 설치하면 코드에서 AWS SDK for C++를 사용할 수 있습니다.

### 프로세스 개요

| 일반 프로세스                                                                                                                                                                                                                                             | 자세한 프로세스                                                                                                                                                |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------|
| SDK 소스 빌드 및 설치 <ol style="list-style-type: none"> <li>1. CMake를 사용하여 SDK용 빌드 파일을 생성합니다.</li> <li>2. SDK를 빌드합니다.</li> <li>3. SDK를 설치합니다.</li> </ol>                                                                                                  | 먼저 소스에서 SDK를 빌드한 다음 설치합니다. <ul style="list-style-type: none"> <li>• <a href="#">Windows에서 빌드</a></li> <li>• <a href="#">Linux/macOS에서 빌드</a></li> </ul> |
| SDK를 사용한 애플리케이션 빌드 <ol style="list-style-type: none"> <li>1. SDK를 사용하기 위해 직접 코드를 작성하거나 샘플 애플리케이션을 사용한 후 AWSSDK 패키지를 cmake 파일에 추가합니다.</li> <li>2. CMake를 사용하여 애플리케이션용 빌드 파일을 생성합니다.</li> <li>3. 애플리케이션을 빌드합니다.</li> <li>4. 애플리케이션을 실행합니다.</li> </ol> | 그런 다음 SDK를 사용하여 자체 애플리케이션을 개발합니다. <ul style="list-style-type: none"> <li>• <a href="#">간단한 애플리케이션 생성</a></li> </ul>                                     |

## Windows에서 AWS SDK for C++ 빌드

AWS SDK for C++를 설정하려면 소스에서 직접 SDK를 빌드하거나 패키지 관리자를 사용하여 라이브러리를 다운로드 할 수 있습니다.

SDK 소스는 서비스별로 개별 패키지로 구분됩니다. 전체 SDK를 설치하는 데 최대 1시간이 걸릴 수 있습니다. 프로그램에서 사용하는 특정 서비스 하위 집합만 설치하면 설치 시간이 단축되고 디스크 공간도 절약됩니다. 설치할 서비스를 선택하려면 프로그램에서 사용하는 각 서비스의 패키지 이름을 알아야 합니다. 패키지 디렉터리 목록은 GitHub의 [aws/aws-sdk-cpp](https://github.com/aws/aws-sdk-cpp)에서 확인할 수 있습니다. 패키지 이름은 서비스 디렉터리 이름의 접미사입니다.

```
aws-sdk-cpp\aws-cpp-sdk-<packageName>    # Repo directory name and packageName
aws-sdk-cpp\aws-cpp-sdk-s3                  # Example: Package name is s3
```

### 사전 조건

일부 대형 AWS 클라이언트를 빌드하려면 최소 4GB의 RAM이 필요합니다. Amazon EC2 인스턴스 유형 t2.micro, t2.small 및 기타 소형 인스턴스 유형에서는 메모리 부족으로 인해 SDK 빌드가 실패할 수 있습니다.

AWS SDK for C++를 사용하려면 다음 중 하나가 필요합니다.

- Microsoft Visual Studio 2015 이후 버전
- GNU Compiler Collection(GCC) 4.9 이상
- Clang 3.3 이후 버전

### curl을 사용하여 Windows용 SDK 빌드

Windows에서 SDK는 [WinHTTP](https://github.com/aws/win-http)를 기본 HTTP 클라이언트로 사용하여 빌드됩니다. 그러나 WinHTTP 1.0은 Amazon Transcribe 및 Amazon Lex와 같은 일부 AWS 서비스에 필요한 HTTP/2 양방향 스트리밍을 지원하지 않습니다. 따라서 SDK에 curl 지원을 추가하여 빌드해야 하는 경우가 있습니다. 사용할 수 있는 모든 curl 다운로드 옵션을 보려면 [curl Releases and Downloads](https://curl.haxx.se/#releases)를 참조하세요. SDK에 curl 지원을 추가하여 빌드하는 한 가지 방법은 다음과 같습니다.

curl 라이브러리 지원을 포함하여 SDK를 빌드하려면

1. [curl for Windows](https://curl.haxx.se/windows/)로 이동하여 Microsoft Windows용 curl 바이너리 패키지를 다운로드합니다.
2. 컴퓨터의 폴더(예: C:\curl)에 패키지의 압축을 풉니다.

3. [CA certificates extracted from Mozilla](#)로 이동하여 `cacert.pem` 파일을 다운로드합니다. 이 Privacy Enhanced Mail(PEM) 파일에는 보안 웹 사이트의 신뢰성을 검증하는 데 사용되는 유효한 디지털 인증서의 번들 포함되어 있습니다. 인증서는 GlobalSign 및 Verisign과 같은 인증 기관(CA)에서 배포합니다.
4. `cacert.pem` 파일을 이전 단계에서 압축을 푼 `bin` 하위 폴더(예: `C:\curl\bin`)로 이동합니다. 파일 이름을 `curl-ca-bundle.crt`로 변경합니다.

또한 다음 절차에서도 Microsoft Build Engine(MSBuild)이 `curl.dll`을 찾을 수 있어야 합니다. 따라서와 같은 Windows PATH 환경 변수에 `curl bin` 폴더 경로를 추가해야 합니다(예: `set PATH=%PATH%;C:\curl\bin`). SDK를 빌드하기 위해 새 명령 프롬프트를 열 때마다 이를 추가해야 합니다. 또는 Windows 시스템 설정에서 환경 변수를 전역적으로 설정하여 설정이 유지되도록 할 수도 있습니다.

다음 절차에 따라 소스에서 SDK를 빌드할 경우 SDK에 `curl`을 빌드하는 데 필요한 명령 구문은 5단계(빌드 파일 생성)를 참조하세요.

코드를 작성할 때 [코드에서 AWS SDK for C++ 서비스 클라이언트 구성](#)에서 `caFile`을 인증서 파일의 위치로 설정해야 합니다. Amazon Transcribe를 사용한 예제는 GitHub의 AWS 코드 예제 리포지토리에서 [transcribe-streaming](#) 섹션을 참조하세요.

## 소스에서 SDK 빌드

명령줄 도구를 사용하여 소스에서 SDK를 빌드할 수 있습니다. 이 방법을 사용하면 SDK 빌드를 사용자 지정할 수 있습니다. 사용 가능한 옵션에 대한 자세한 내용은 [CMake 파라미터](#)를 참조하세요. 주요 단계는 세 가지입니다. 첫째, CMake를 사용하여 파일을 빌드합니다. 둘째, MSBuild를 사용하여 운영 체제 및 빌드 도구 체인과 호환되는 SDK 바이너리를 빌드합니다. 셋째, 개발 시스템의 올바른 위치에 바이너리를 설치하거나 복사합니다.

소스에서 SDK를 빌드하려면

1. [CMake](#)(최소 버전 3.13)와 해당 플랫폼용 관련 빌드 도구를 설치합니다. `cmake`는 PATH에 추가하는 것이 좋습니다. CMake 버전을 확인하려면 명령 프롬프트를 열고 `cmake --version` 명령을 실행합니다.
2. 명령 프롬프트에서 SDK를 저장할 폴더로 이동합니다.
3. 최신 소스 코드를 가져옵니다.

버전 1.11의 경우 외부 종속성을 래핑하는 데 `git` 하위 모듈을 사용합니다. 여기에는 AWS SDK 및 도구 참조 안내서에 설명된 [CRT 라이브러리](#)가 포함됩니다.

GitHub의 [aws/aws-sdk-cpp](https://github.com/aws/aws-sdk-cpp)에서 SDK 소스를 다운로드하거나 복제합니다.

- Git으로 복제: HTTPS

```
git clone --recurse-submodules https://github.com/aws/aws-sdk-cpp
```

- Git으로 복제: SSH

```
git clone --recurse-submodules git@github.com:aws/aws-sdk-cpp.git
```

4. 생성된 빌드 파일은 SDK 소스 디렉터리 외부에 저장하는 것이 좋습니다. 빌드 파일을 저장할 새 디렉터리를 생성하고 해당 폴더로 이동합니다.

```
mkdir sdk_build
cd sdk_build
```

5. cmake를 실행하여 빌드 파일을 생성합니다. cmake 명령줄에서 디버그 또는 릴리스 버전을 빌드할지 여부를 지정합니다. 이 절차 전체에서 Debug를 선택하면 애플리케이션 코드의 디버그 구성이 실행됩니다. 이 절차 전체에서 Release를 선택하면 애플리케이션 코드의 릴리스 구성이 실행됩니다. Windows의 경우 SDK 설치 위치는 일반적으로 \Program Files (x86)\aws-cpp-sdk-all\입니다. 명령 구문:

```
{path to cmake if not in PATH} {path to source location of aws-sdk-cpp} -DCMAKE_BUILD_TYPE=[Debug | Release] -DCMAKE_PREFIX_PATH={path to install destination}
```

빌드 출력을 수정하는 자세한 방법은 [CMake 파라미터](#)를 참조하세요.

빌드 파일을 생성하려면 다음 중 하나를 수행합니다.

- 빌드 파일 생성(모든 AWS 서비스): 전체 SDK를 빌드하려면 cmake를 실행하고 디버그 또는 릴리스 버전을 빌드할지 여부를 지정합니다. 예:

```
cmake "..\aws-sdk-cpp" -DCMAKE_BUILD_TYPE=Debug -DCMAKE_PREFIX_PATH="C:\Program Files (x86)\aws-cpp-sdk-all"
```

- 빌드 파일 생성(AWS 서비스 하위 집합): SDK의 특정 서비스 또는 서비스 패키지만 빌드하려면 CMake [BUILD\\_ONLY](#) 파라미터를 추가하고, 서비스 이름은 세미콜론으로 구분합니다. 다음 예제는 Amazon S3 서비스 패키지만 빌드합니다.

```
cmake ..\aws-sdk-cpp -DCMAKE_BUILD_TYPE=Debug -DBUILD_ONLY="s3" -
DCMAKE_PREFIX_PATH="C:\Program Files (x86)\aws-cpp-sdk-all"
```

- 빌드 파일 생성(curl 사용): curl 사전 조건을 완료한 후 SDK에 curl 지원을 포함하려면 세 가지 추가 cmake 명령줄 옵션인 [FORCE\\_CURL](#), [CURL\\_INCLUDE\\_DIR](#), [CURL\\_LIBRARY](#)가 필요합니다. 예:

```
cmake ..\aws-sdk-cpp -DCMAKE_BUILD_TYPE=Debug -DFORCE_CURL=ON -
DCURL_INCLUDE_DIR='C:/curl/include'
-DCURL_LIBRARY='C:/curl/lib/libcurl.dll.a' -DCMAKE_PREFIX_PATH="C:\Program
Files (x86)\aws-cpp-sdk-all"
```

#### Note

Failed to build third-party libraries 오류가 발생하면 **cmake --version**을 실행하여 CMake 버전을 확인합니다. CMake 버전은 최소 3.13 이상을 사용해야 합니다.

6. SDK 바이너리를 빌드합니다. 전체 SDK를 빌드하는 경우 이 단계에 1시간 이상 소요될 수 있습니다. 명령 구문:

```
{path to cmake if not in PATH} --build . --config=[Debug | Release]
```

```
cmake --build . --config=Debug
```

#### Note

코드 실행을 진행할 수 없습니다. dll을 찾을 수 없습니다. 프로그램을 다시 설치하면 이 문제가 해결될 수 있습니다." 오류가 발생할 경우 cmake 명령을 다시 시도하세요.

7. 관리자 권한으로 명령 프롬프트를 열어 CMAKE\_PREFIX\_PATH 파라미터를 사용하여 앞서 지정한 위치에 SDK를 설치합니다. 명령 구문:

```
{path to cmake if not in PATH} --install . --config=[Debug | Release]
```

```
cmake --install . --config=Debug
```

## Windows에서 Android용 빌드

Android용 빌드를 수행하려면 cmake 명령줄에 `-DTARGET_ARCH=ANDROID`를 추가합니다. AWS SDK for C++에는 적절한 환경 변수(`ANDROID_NDK`)를 참조하여 필요한 항목을 제공하는 CMake 도구 체인 파일이 포함되어 있습니다.

Windows에서 Android용 SDK를 빌드하려면 Visual Studio(2015 이상) 개발자 명령 프롬프트에서 cmake를 실행해야 합니다. 또한 [NMAKE](#)가 설치되어 있어야 하며, 경로에 `git` 및 `patch` 명령이 포함되어 있어야 합니다. Windows 시스템에 git이 설치되어 있다면 `patch`는 일반적으로 형제 디렉터리(예: `.../Git/usr/bin/`)에서 찾을 수 있습니다. 이러한 요구 사항을 확인한 후에는 NMAKE를 사용하기 위해 cmake 명령줄이 약간 변경됩니다.

```
cmake -G "NMake Makefiles" -DTARGET_ARCH=ANDROID <other options> ..
```

NMAKE는 순차적으로 빌드합니다. 더 빠르게 빌드하려면 NMAKE 대신 JOM을 설치하고, 다음과 같이 cmake 간접 호출을 변경하는 것이 좋습니다.

```
cmake -G "NMake Makefiles JOM" -DTARGET_ARCH=ANDROID <other options> ..
```

예제 애플리케이션은 [Setting up an Android application with AWS SDK for C++](#)를 참조하세요.

## Linux/macOS에서 AWS SDK for C++ 빌드

AWS SDK for C++를 설정하려면 소스에서 직접 SDK를 빌드하거나 패키지 관리자를 사용하여 라이브러리를 다운로드 할 수 있습니다.

SDK 소스는 서비스별로 개별 패키지로 구분됩니다. 전체 SDK를 설치하는 데 최대 1시간이 걸릴 수 있습니다. 프로그램에서 사용하는 특정 서비스 하위 집합만 설치하면 설치 시간이 단축되고 디스크 공간도 절약됩니다. 설치할 서비스를 선택하려면 프로그램에서 사용하는 각 서비스의 패키지 이름을 알아야 합니다. 패키지 디렉터리 목록은 GitHub의 [aws/aws-sdk-cpp](#)에서 확인할 수 있습니다. 패키지 이름은 서비스 디렉터리 이름의 접미사입니다.

```
aws-sdk-cpp\aws-cpp-sdk-<packageName> # Repo directory name and packageName
aws-sdk-cpp\aws-cpp-sdk-s3             # Example: Package name is s3
```

## 사전 조건

일부 대형 AWS 클라이언트를 빌드하려면 최소 4GB의 RAM이 필요합니다. Amazon EC2 인스턴스 유형 t2.micro, t2.small 및 기타 소형 인스턴스 유형에서는 메모리 부족으로 인해 SDK 빌드가 실패할 수 있습니다.

AWS SDK for C++를 사용하려면 다음 중 하나가 필요합니다.

- GNU Compiler Collection(GCC) 4.9 이상
- Clang 3.3 이후 버전

## Linux 시스템의 추가 요구 사항

libcurl, libopenssl, libuuid, zlib용 헤더 파일(-dev 패키지)이 반드시 필요하며 Amazon Polly 지원을 위해 선택적으로 libpulse도 필요합니다. 시스템의 패키지 관리자를 사용하여 패키지를 찾을 수 있습니다.

Debian/Ubuntu 기반 시스템에 패키지를 설치하려면

- ```
sudo apt-get install libcurl4-openssl-dev libssl-dev uuid-dev zlib1g-dev libpulse-dev
```

Amazon Linux/Redhat/Fedora/CentOS 기반 시스템에 패키지를 설치하려면

- ```
sudo yum install libcurl-devel openssl-devel libuuid-devel pulseaudio-libs-devel
```

## 소스에서 SDK 빌드

vcpkg를 사용하는 대신 명령줄 도구를 사용하여 소스에서 SDK를 빌드할 수 있습니다. 이 방법을 사용하면 SDK 빌드를 사용자 지정할 수 있습니다. 사용 가능한 옵션에 대한 자세한 내용은 [CMake 파라미터](#)를 참조하세요.

소스에서 SDK를 빌드하려면

1. [CMake](#)(최소 버전 3.13)와 해당 플랫폼용 관련 빌드 도구를 설치합니다. cmake는 PATH에 추가하는 것이 좋습니다. CMake 버전을 확인하려면 명령 프롬프트를 열고 **cmake --version** 명령을 실행합니다.

2. 명령 프롬프트에서 SDK를 저장할 폴더로 이동합니다.
3. 최신 소스 코드를 가져옵니다.

버전 1.11의 경우 외부 종속성을 래핑하는 데 git 하위 모듈을 사용합니다. 여기에는 AWS SDK 및 도구 참조 안내서에 설명된 [CRT 라이브러리](#)가 포함됩니다.

GitHub의 [aws/aws-sdk-cpp](#)에서 SDK 소스를 다운로드하거나 복제합니다.

- Git으로 복제: HTTPS

```
git clone --recurse-submodules https://github.com/aws/aws-sdk-cpp
```

- Git으로 복제: SSH

```
git clone --recurse-submodules git@github.com:aws/aws-sdk-cpp.git
```

4. 생성된 빌드 파일은 SDK 소스 디렉터리 외부에 저장하는 것이 좋습니다. 빌드 파일을 저장할 새 디렉터리를 생성하고 해당 폴더로 이동합니다.

```
mkdir sdk_build  
cd sdk_build
```

5. cmake를 실행하여 빌드 파일을 생성합니다. cmake 명령줄에서 디버그 또는 릴리스 버전을 빌드 할지 여부를 지정합니다. 이 절차 전체에서 Debug를 선택하면 애플리케이션 코드의 디버그 구성이 실행됩니다. 이 절차 전체에서 Release를 선택하면 애플리케이션 코드의 릴리스 구성이 실행됩니다. 명령 구문:

```
{path to cmake if not in PATH} {path to source location of aws-sdk-cpp} -DCMAKE_BUILD_TYPE=[Debug | Release] -DCMAKE_PREFIX_PATH={path to install} -DCMAKE_INSTALL_PREFIX={path to install}
```

빌드 출력을 수정하는 자세한 방법은 [CMake 파라미터](#)를 참조하세요.

#### Note

대/소문자를 구분하지 않는 파일 시스템을 사용하여 Mac에서 빌드할 경우 빌드를 실행하는 디렉터리에서 pwd 명령의 출력을 확인합니다. pwd 출력에 /Users 및 Documents와 같이 디렉터리 이름이 대소문자를 혼용하여 표시되는지 확인합니다.

빌드 파일을 생성하려면 다음 중 하나를 수행합니다.

- 빌드 파일 생성(모든 AWS 서비스): 전체 SDK를 빌드하려면 `cmake`를 실행하고 디버그 또는 릴리스 버전을 빌드할지 여부를 지정합니다. 예:

```
cmake ../aws-sdk-cpp -DCMAKE_BUILD_TYPE=Debug -DCMAKE_PREFIX_PATH=/usr/local/ -
DCMAKE_INSTALL_PREFIX=/usr/local/
```

- 빌드 파일 생성(AWS 서비스 하위 집합): SDK의 특정 서비스 또는 서비스 패키지만 빌드하려면 CMake `BUILD_ONLY` 파라미터를 추가하고, 서비스 이름은 세미콜론으로 구분합니다. 다음 예제는 Amazon S3 서비스 패키지만 빌드합니다.

```
cmake ../aws-sdk-cpp -DCMAKE_BUILD_TYPE=Debug -DCMAKE_PREFIX_PATH=/usr/local/ -
DCMAKE_INSTALL_PREFIX=/usr/local/ -DBUILD_ONLY="s3"
```

#### Note

Failed to build third-party libraries 오류가 발생하면 `cmake --version`을 실행하여 CMake 버전을 확인합니다. CMake 버전은 최소 3.13 이상을 사용해야 합니다.

6. SDK 바이너리를 빌드합니다. 전체 SDK를 빌드하는 경우 이 작업에 1시간 이상 소요될 수 있습니다.

```
cmake --build . --config=Debug
```

7. SDK를 설치합니다. 설치하기로 선택한 위치에 따라 권한을 에스컬레이션해야 할 수 있습니다.

```
cmake --install . --config=Debug
```

## Linux에서 Android용 빌드

Android용 빌드를 수행하려면 `cmake` 명령줄에 `-DTARGET_ARCH=ANDROID`를 추가합니다. AWS SDK for C++에는 적절한 환경 변수(`ANDROID_NDK`)를 참조하여 필요한 항목을 제공하는 CMake 도구 체인 파일이 포함되어 있습니다. 예제 애플리케이션은 [Setting up an Android application with AWS SDK for C++](#)를 참조하세요.

# AWS SDK for C++를 사용하여 간단한 애플리케이션 생성

[CMake](#)는 애플리케이션의 종속성을 관리하고 빌드 중인 플랫폼에 적합한 makefile을 생성하는 데 사용하는 빌드 도구입니다. CMake를 사용하여 AWS SDK for C++로 프로젝트를 생성하고 빌드할 수 있습니다.

이 예제에서는 사용자가 소유한 Amazon S3 버킷을 보고합니다. 이 예제를 수행하기 위해 AWS 계정에 Amazon S3 버킷이 반드시 필요하지는 않지만, 최소한 하나 이상 보유할 경우 훨씬 더 유용하게 활용할 수 있습니다. 아직 버킷이 없는 경우 Amazon Simple Storage Service 사용 설명서에서 [버킷 만들기](#)를 참조하세요.

## 1단계: 코드 작성

이 예제는 하나의 소스 파일(hello\_s3.cpp)과 하나의 CMakeLists.txt 파일을 포함하는 하나의 폴더로 구성됩니다. 프로그램은 Amazon S3를 사용하여 스토리지 버킷 정보를 보고합니다. 이 코드는 GitHub의 [AWS 코드 예제 리포지토리](#)에서도 확인할 수 있습니다.

CMakeLists.txt 빌드 구성 파일에서 여러 옵션을 설정할 수 있습니다. 자세한 내용은 CMake 웹 사이트에서 [CMake 자습서](#)를 참조하세요.

### Note

심층 분석: CMAKE\_PREFIX\_PATH 설정

기본적으로 macOS, Linux, Android 및 기타 비 Windows 플랫폼에서 AWS SDK for C++는 /usr/local에 설치되고 Windows에서는 \Program Files (x86)\aws-cpp-sdk-all에 설치됩니다.

AWS SDK를 이러한 표준 위치에 설치하면 CMake가 필요한 리소스를 자동으로 찾습니다. 그러나 AWS SDK를 사용자 지정 위치에 설치하는 경우, SDK 빌드 결과 생성되는 다음 리소스를 찾을 수 있는 위치를 CMake에 알려야 합니다:

- AWSSDKConfig.cmake: 프로젝트에서 AWS SDK 라이브러리를 찾고 사용하는 방법을 CMake에 알려주는 구성 파일입니다. 이 파일이 없으면 CMake는 AWS SDK 헤더 파일을 찾거나, AWS SDK 라이브러리에 연결하거나, 적절한 컴파일러 플래그를 설정할 수 없습니다.
- (버전 1.8 이하) 종속성 위치: aws-c-event-stream, aws-c-common, aws-checksums

사용자 지정 설치 경로를 설정하려면:

```
cmake -DCMAKE_PREFIX_PATH=/path/to/your/aws-sdk-installation /path/to/project/you/are/building
```

사용자 지정 설치 시 CMAKE\_PREFIX\_PATH를 설정하지 않으면 CMake가 CMakeLists.txt에서 find\_package(AWSSDK)를 처리할 때 "AWSSDK를 찾을 수 없음"과 같은 오류가 발생하면서 빌드가 실패합니다.

### Note

심층 분석: Windows 런타임 라이브러리

프로그램을 실행하려면 프로그램의 실행 위치에 aws-c-common.dll, aws-c-event-stream.dll, aws-checksums.dll, aws-cpp-sdk-core.dll 등의 여러 DLL이 필요할 뿐만 아니라 프로그램의 구성 요소에 따른 특정 DLL도 필요합니다. 이 예제에서는 Amazon S3를 사용하므로 aws-cpp-sdk-s3도 필요합니다. CMakeLists.txt 파일의 두 번째 if 문은 이 요구 사항을 충족시키기 위해 설치 위치에서 실행 파일 위치로 해당 라이브러리를 복사합니다. AWSSDK\_CPY\_DYN\_LIBS는 AWS SDK for C++에서 정의한 매크로로, SDK의 DLL을 설치 위치에서 프로그램의 실행 가능한 위치로 복사합니다. 이러한 DLL이 실행 파일 위치에 없는 경우 '파일을 찾을 수 없음' 런타임 예외가 발생합니다. 이러한 오류가 발생할 경우 CMakeLists.txt 파일의 이 부분을 검토하여 고유한 환경에 필요한 변경 사항을 확인합니다.

폴더와 소스 파일을 생성하려면

1. 소스 파일을 보관할 hello\_s3 디렉터리 및/또는 프로젝트를 생성합니다.

### Note

Visual Studio에서 이 예제를 완료하려면: 새 프로젝트 만들기를 선택한 다음 CMake 프로젝트를 선택합니다. 프로젝트 이름을 hello\_s3로 지정합니다. 이 프로젝트 이름은 CMakeLists.txt 파일에 사용됩니다.

2. 해당 폴더 내에 다음 코드가 포함된 hello\_s3.cpp 파일을 추가합니다. 이 코드는 사용자가 소유한 Amazon S3 버킷을 보고합니다.

```
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
```

```
#include <iostream>
#include <aws/core/auth/AWSCredentialsProviderChain.h>
using namespace Aws;
using namespace Aws::Auth;

/*
 * A "Hello S3" starter application which initializes an Amazon Simple Storage
 * Service (Amazon S3) client
 * and lists the Amazon S3 buckets in the selected region.
 *
 * main function
 *
 * Usage: 'hello_s3'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        // You don't normally have to test that you are authenticated. But the S3
        // service permits anonymous requests, thus the s3Client will return "success" and 0
        // buckets even if you are unauthenticated, which can be confusing to a new user.
        auto provider = Aws::MakeShared<DefaultAWSCredentialsProviderChain>("alloc-
tag");
        auto creds = provider->GetAWSCredentials();
        if (creds.IsEmpty()) {
            std::cerr << "Failed authentication" << std::endl;
        }

        Aws::S3::S3Client s3Client(clientConfig);
        auto outcome = s3Client.ListBuckets();

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed with error: " << outcome.GetError() << std::endl;
            result = 1;
        } else {
```

```

        std::cout << "Found " << outcome.GetResult().GetBuckets().size()
                    << " buckets\n";
        for (auto &bucket: outcome.GetResult().GetBuckets()) {
            std::cout << bucket.GetName() << std::endl;
        }
    }
}

Aws::ShutdownAPI(options); // Should only be called once.
return result;
}

```

3. 프로젝트 이름, 실행 파일, 소스 파일 및 연결된 라이브러리를 지정하는 CMakeLists.txt 파일을 추가합니다.

```

# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS s3)

# Set this project's name.
project("hello_s3")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
                    # for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
        "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)

```

```
# Copy relevant AWS SDK for C++ libraries into the current binary directory for
running and debugging.

# set(BIN_SUB_DIR "/Debug") # if you are building from the command line you may
need to uncomment this
# and set the proper subdirectory to the executables' location.

AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
    hello_s3.cpp)

target_link_libraries(${PROJECT_NAME}
    ${AWSSDK_LINK_LIBRARIES})
```

## 2단계: CMake를 사용한 빌드

CMake는 CMakeLists.txt에 포함된 정보를 사용하여 실행 프로그램을 빌드합니다.

IDE의 표준 관행에 따라 애플리케이션을 빌드하는 것이 좋습니다.

명령줄에서 애플리케이션을 빌드하려면

1. **cmake**로 애플리케이션을 빌드할 디렉터리를 생성합니다.

```
mkdir my_project_build
```

2. 빌드 디렉터리로 변경하고 프로젝트의 소스 디렉터리 경로를 사용하여 **cmake**를 실행합니다.

```
cd my_project_build
cmake ../
```

3. **cmake**로 빌드 디렉터리를 생성한 후에는 **make**(Windows의 경우 **nmake**) 또는 MSBUILD(`msbuild ALL_BUILD.vcxproj` 또는 `cmake --build . --config=Debug`)를 사용하여 애플리케이션을 빌드할 수 있습니다.

### 3단계: 실행

이 애플리케이션을 실행하면 총 Amazon S3 버킷 수와 각 버킷의 이름을 나열하는 콘솔 출력이 표시됩니다.

IDE의 표준 관행에 따라 애플리케이션을 실행하는 것이 좋습니다.

#### Note

로그인하는 것을 잊지 마세요! IAM Identity Center를 사용하여 인증하는 경우 AWS CLI `aws sso login` 명령을 사용하여 로그인해야 합니다.

명령줄을 통해 프로그램을 실행하려면

1. 빌드 결과가 생성된 디버그 디렉터리로 변경합니다.
2. 실행 파일의 이름을 사용하여 프로그램을 실행합니다.

```
hello_s3
```

AWS SDK for C++를 사용하는 추가 예제는 [AWS SDK for C++를 사용하여 AWS 서비스를 호출하는 안내 예제](#) 섹션을 참조하세요.

## 패키지 관리자에서 AWS SDK for C++ 가져오기

#### Important

homebrew 또는 vcpkg 등의 패키지 관리자를 사용하는 경우:  
SDK for C++를 새 버전으로 업데이트한 후에는 SDK에 의존하는 라이브러리 또는 실행 파일을 다시 컴파일해야 합니다.

AWS SDK for C++를 설정하려면 소스에서 직접 SDK를 빌드하거나 패키지 관리자를 사용하여 라이브러리를 다운로드 할 수 있습니다.

SDK 소스는 서비스별로 개별 패키지로 구분됩니다. 전체 SDK를 설치하는 데 최대 1시간이 걸릴 수 있습니다. 프로그램에서 사용하는 특정 서비스 하위 집합만 설치하면 설치 시간이 단축되고 디스크 공간

도 절약됩니다. 설치할 서비스를 선택하려면 프로그램에서 사용하는 각 서비스의 패키지 이름을 알아야 합니다. 패키지 디렉터리 목록은 GitHub의 [aws/aws-sdk-cpp](https://github.com/aws/aws-sdk-cpp)에서 확인할 수 있습니다. 패키지 이름은 서비스 디렉터리 이름의 접미사입니다.

```
aws-sdk-cpp\aws-cpp-sdk-<packageName>    # Repo directory name and packageName
aws-sdk-cpp\aws-cpp-sdk-s3                # Example: Package name is s3
```

## 사전 조건

일부 대형 AWS 클라이언트를 빌드하려면 최소 4GB의 RAM이 필요합니다. Amazon EC2 인스턴스 유형 t2.micro, t2.small 및 기타 소형 인스턴스 유형에서는 메모리 부족으로 인해 SDK 빌드가 실패할 수 있습니다.

### Linux/macOS

Linux/macOS에서 AWS SDK for C++를 사용하려면 다음 중 하나가 필요합니다.

- GNU Compiler Collection(GCC) 4.9 이상
- Clang 3.3 이후 버전

### Windows

Windows에서 AWS SDK for C++를 사용하려면 다음 중 하나가 필요합니다.

- Microsoft Visual Studio 2015 이후 버전
- GNU Compiler Collection(GCC) 4.9 이상
- Clang 3.3 이후 버전

## vcpkg를 사용하여 SDK 가져오기

### Important

사용 가능한 vcpkg 배포는 외부 기여자가 지원하며 AWS를 통해 제공되지 않습니다. [소스에서 설치](#)하면 항상 최신 버전을 사용할 수 있습니다.

[vcpkg](#)은 외부 기여자들이 업데이트하고 유지 관리하는 패키지 관리자입니다. 이 패키지 관리자는 AWS를 통해 제공되지 않으므로 AWS SDK for C++의 최신 버전을 반영하지 않을 수 있습니다. AWS

에서 버전을 릴리스한 시점과 외부 패키지 관리자를 통해 사용할 수 있게 되는 시점 사이에 시차가 있을 수 있습니다. [소스에서 설치](#)하면 항상 최신 버전을 사용할 수 있습니다.

시스템에 [vcpkg](#)를 설치해야 합니다.

- [vcpkg](#) GitHub Readme의 지침에 따라 vcpkg를 다운로드하고 부트스트랩합니다. 메시지가 표시되면 다음 옵션을 대체합니다.
- 이러한 지침의 일부로 다음을 입력하도록 안내됩니다.

```
.\vcpkg\vcpkg install [packages to install]
```

전체 SDK를 설치하려면 `.\vcpkg\vcpkg install "aws-sdk-cpp[*]" --recurse`를 입력하거나, 설치할 SDK의 특정 서비스만 지정하려면 괄호 안에 패키지 이름을 추가합니다(예: `.\vcpkg\vcpkg install "aws-sdk-cpp[s3, ec2]" --recurse`).

출력에는 다음과 같은 메시지가 표시됩니다.

```
CMake projects should use: "-DCMAKE_TOOLCHAIN_FILE=C:/dev/vcpkg/vcpkg/scripts/buildsystems/vcpkg.cmake"
```

- 나중에 CMake에 사용할 수 있도록 전체 `-DCMAKE_TOOLCHAIN_FILE` 명령을 복사합니다. vcpkg GitHub Readme에는 이 명령을 도구 세트에서 사용할 위치도 안내되어 있습니다.
- vcpkg를 통해 설치한 빌드 구성 유형을 기록해 두어야 할 수도 있습니다. 콘솔 출력에는 빌드 구성과 SDK 버전이 표시됩니다. 다음 예제 출력은 빌드 구성이 "x86-windows"이고 설치된 AWS SDK for C++ 버전이 1.8임을 나타냅니다.

```
The following packages will be built and installed:
aws-sdk-cpp[core,dynamodb,kinesis,s3]:x86-windows -> 1.8.126#6
```

AWS SDK for C++를 설치한 후에는 SDK를 사용하여 자체 애플리케이션을 개발할 수 있습니다. [간단한 애플리케이션 생성](#)에 표시된 예제에서는 사용자가 소유한 Amazon S3 버킷을 보고합니다.

## AWS SDK for C++ 빌드 문제 해결

소스에서 AWS SDK for C++를 빌드할 때 다음과 같은 일반적인 빌드 문제가 발생할 수 있습니다.

### 주제

- [CMake 오류: "AWSSDK"에서 제공하는 패키지 구성 파일을 찾을 수 없음](#)
- [CMake 오류: 로드 파일을 찾을 수 없음\(SDK 버전 1.8 사용 중\)](#)
- [CMake 오류: 로드 파일을 찾을 수 없음](#)
- [런타임 오류: aws-\\*.dll을 찾을 수 없어 진행할 수 없음](#)

## CMake 오류: "AWSSDK"에서 제공하는 패키지 구성 파일을 찾을 수 없음

설치된 SDK를 찾을 수 없는 경우 CMake에서 다음 오류가 발생합니다.

```
1> [CMake] CMake Error at C:\CodeRepos\CMakeProject1\CMakeLists.txt:4 (find_package):
1> [CMake]   Could not find a package configuration file provided by "AWSSDK" with any
1> [CMake]   of the following names:
1> [CMake]
1> [CMake]     AWSSDKConfig.cmake
1> [CMake]     awssdk-config.cmake
1> [CMake]
1> [CMake]   Add the installation prefix of "AWSSDK" to CMAKE_PREFIX_PATH or set
1> [CMake]   "AWSSDK_DIR" to a directory containing one of the above files.  If
1> [CMake]   "AWSSDK"
1> [CMake]   provides a separate development package or SDK, be sure it has been
1> [CMake]   installed.
```

이 오류를 해결하려면 설치된 SDK(예: [Windows](#), [Linux/macOS](#) 등 SDK 설치 결과로 생성된 폴더)를 찾을 수 있는 위치를 CMake에 알립니다. CMakeLists.txt 파일에서 find\_package()의 첫 번째 호출 앞에 다음 명령을 삽입합니다. 예제는 [??? 섹션](#)을 참조하세요.

```
list(APPEND CMAKE_PREFIX_PATH "C:\\Program Files (x86)\\aws-cpp-sdk-all\\lib\\cmake")
```

## CMake 오류: 로드 파일을 찾을 수 없음(SDK 버전 1.8 사용 중)

설치된 라이브러리를 찾을 수 없는 경우 CMake에서 다음 오류가 발생합니다.

```
1> [CMake] include could not find load file:
1> [CMake]
1> [CMake]   C:/Program Files (x86)/aws-cpp-sdk-all/lib/aws-c-common/cmake/static/
1> [CMake]   aws-c-common-targets.cmake

1> [CMake] include could not find load file:
1> [CMake]
```

```
1> [CMake]      C:/Program Files (x86)/aws-cpp-sdk-all/lib/aws-checksums/cmake/static/
aws-checksums-targets.cmake
1> [CMake]      include could not find load file:
1> [CMake]
1> [CMake]      C:/Program Files (x86)/aws-cpp-sdk-all/lib/aws-checksums/cmake/static/
aws-checksums-targets.cmake
```

이 오류를 해결하려면 설치된 SDK(예: [Windows](#), [Linux/macOS](#) 등 SDK 설치 결과로 생성된 폴더)를 찾을 수 있는 위치를 CMake에 알립니다. CMakeLists.txt 파일에서 find\_package()의 첫 번째 호출 앞에 다음 명령을 삽입합니다. 예제는 [??? 섹션](#)을 참조하세요.

```
#Set the location of where Windows can find the installed libraries of the SDK.
if(MSVC)
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
"${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif()
```

이러한 종속성은 이후 버전에서 다르게 처리되므로 이 솔루션은 SDK v1.8에만 해당됩니다. 버전 1.9에서는 aws-sdk-cpp 및 aws-c-\* 라이브러리 사이에 중간 계층을 도입하여 이러한 문제를 해결합니다. 이 새 계층을 aws-crt-cpp라고 하며, SDK for C++의 git 하위 모듈입니다. aws-crt-cpp에는 aws-c-\* 라이브러리(aws-c-common, aws-checksums, aws-c-event-stream 등 포함)도 자체 git 하위 모듈로 포함되어 있습니다. 이를 통해 SDK for C++는 모든 CRT 라이브러리를 재귀적으로 가져오고 빌드 프로세스를 개선할 수 있습니다.

## CMake 오류: 로드 파일을 찾을 수 없음

설치된 라이브러리를 찾을 수 없는 경우 CMake에서 다음 오류가 발생합니다.

```
CMake Error at C:/Program Files (x86)/aws-cpp-sdk-all/lib/aws-c-auth/cmake/aws-c-auth-
config.cmake:11
    (include): include could not find load file:
    C:/Program Files (x86)/aws-cpp-sdk-all/lib/aws-c-auth/cmake/static/aws-c-auth-
targets.cmake
```

이 오류를 해결하려면 공유 라이브러리를 빌드하도록 CMake에 요청합니다. CMakeLists.txt 파일에서 find\_package()의 첫 번째 호출 앞에 다음 명령을 삽입합니다. 예제는 [??? 섹션](#)을 참조하세요.

```
set(BUILD_SHARED_LIBS ON CACHE STRING "Link to shared libraries by default.")
```

## 런타임 오류: `aws-*.dll`을 찾을 수 없어 진행할 수 없음

CMake는 필요한 DLL을 찾을 수 없는 경우 다음과 유사한 오류를 발생시킵니다.

```
The code execution cannot proceed because aws-cpp-sdk-[dynamodb].dll was not found.  
Reinstalling the program may fix this problem.
```

이 오류는 SDK for C++에 필요한 라이브러리 또는 실행 파일을 애플리케이션 실행 파일과 동일한 폴더에서 사용할 수 없기 때문에 발생합니다. 이 오류를 해결하려면 SDK 빌드 출력을 실행 파일 위치에 복사합니다. 오류의 특정 DLL 파일 이름은 사용 중인 AWS 서비스에 따라 달라집니다. 다음 중 하나를 수행합니다.

- AWS SDK for C++ 설치의 `/bin` 폴더 내용을 애플리케이션의 빌드 폴더로 복사합니다.
- `CMakeLists.txt` 파일에서 `AWSSDK_CPY_DYN_LIBS` 매크로를 사용하여 이들을 복사합니다.

이 매크로를 사용하여 복사를 수행하려면 `AWSSDK_CPY_DYN_LIBS(SERVICE_LIST "" ${CMAKE_CURRENT_BINARY_DIR})` 또는 `AWSSDK_CPY_DYN_LIBS(SERVICE_LIST "" ${CMAKE_CURRENT_BINARY_DIR}/${CMAKE_BUILD_TYPE})`에 대한 호출을 `CMakeLists.txt` 파일에 추가합니다. 예제는 [???](#) 섹션을 참조하세요.

빌드 환경에 맞는 올바른 복사 경로를 선택합니다. 명령줄을 통해 빌드하면 빌드 출력이 하위 폴더(`/Debug`)에 저장되는 경우가 많지만 Visual Studio 및 기타 IDE를 사용하면 그렇지 않은 경우가 많습니다. 출력 실행 파일의 위치를 확인하고 매크로가 해당 위치로 복사하고 있는지 확인합니다. 이러한 유형의 변경을 수행할 때는 다음 빌드의 시작점을 확보할 수 있도록 빌드 출력 디렉터리의 내용을 삭제하는 것이 좋습니다.

# AWS SDK for C++에서 서비스 클라이언트 구성

프로그래밍 방식으로 AWS 서비스에 액세스하기 위해 AWS SDK for C++는 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 서비스를 요청하려면 먼저 서비스 클라이언트를 생성해야 합니다. 코드에 사용된 각 AWS 서비스에는 해당 서비스와 상호 작용하기 위한 자체 라이브러리와 전용 유형이 있습니다. 클라이언트는 서비스가 제공하는 각 API 작업마다 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 안내서](#)에는 많은 AWS SDK에 공통적인 설정, 기능 및 기타 기본 개념도 포함되어 있습니다.

## 주제

- [AWS SDK for C++에서 Aws::SDKOptions를 사용한 일반 구성](#)
- [외부에서 SDK for C++ 서비스 클라이언트 구성](#)
- [코드에서 AWS SDK for C++ 서비스 클라이언트 구성](#)
- [AWS SDK for C++의 AWS 리전 설정](#)
- [AWS SDK for C++ 자격 증명 공급자 사용](#)
- [AWS SDK for C++ 빌드용 CMake 파라미터](#)
- [AWS SDK for C++에서 로깅 구성 및 사용](#)
- [AWS SDK for C++에서 HTTP 클라이언트 재정의](#)
- [AWS SDK for C++에서 HttpClient 및 AWSClient가 사용하는 입출력 스트림 제어](#)
- [AWS SDK for C++에서 사용자 지정 libcrypto 라이브러리 사용](#)

## AWS SDK for C++에서 **Aws::SDKOptions**를 사용한 일반 구성

[Aws::SDKOptions](#) 구조체에는 SDK 구성 옵션이 포함되어 있습니다. `Aws::SDKOptions`는 일반 SDK 구성에 중점을 두는 반면, [ClientConfiguration](#) 구조체는 AWS 서비스와의 통신 구성에 중점을 둡니다.

`Aws::SDKOptions`의 인스턴스는 [Aws::InitAPI](#) 및 [Aws::ShutdownAPI 메서드](#)에 전달됩니다. 두 메서드 모두에 동일한 인스턴스를 전송해야 합니다.

다음 샘플은 사용 가능한 몇 가지 옵션을 보여줍니다.

- 기본 로거를 사용하여 로깅 켜기

```
Aws::SDKOptions options;
options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Info;
Aws::InitAPI(options);
{
    // make your SDK calls here.
}
Aws::ShutdownAPI(options);
```

- 기본 HTTP 클라이언트 팩토리 재정의

```
Aws::SDKOptions options;
options.httpOptions.httpClientFactory_create_fn = [](){
    return Aws::MakeShared<MyCustomHttpClientFactory>(
        "ALLOC_TAG", arg1);
};
Aws::InitAPI(options);
{
    // make your SDK calls here.
}
Aws::ShutdownAPI(options);
```

### Note

`httpOptions`는 `std::shared_ptr` 대신 클로저(익명 함수 또는 Lambda 표현식이라고도 함)를 받습니다. 각 SDK 팩토리 함수는 팩토리 메모리 할당이 발생하는 시점에 메모리 관리자가 아직 설치되지 않았기 때문에 이러한 방식으로 작동합니다. 메서드에 클로저를 전달하

면 메모리 할당이 안전하게 수행될 수 있을 때 메모리 관리자가 호출되어 메모리 할당을 수행합니다. 이 절차를 수행하는 간단한 기법은 Lambda 표현식을 사용하는 것입니다.

- 글로벌 SIGPIPE 핸들러 사용

curl 및 OpenSSL을 사용하여 SDK for C++를 빌드하는 경우 신호 핸들러를 지정해야 합니다. 자체 사용자 지정 신호 핸들러를 사용하지 않는 경우 `installSigPipeHandler`를 `true`로 설정합니다.

```
Aws::SDKOptions options;
options.httpOptions.installSigPipeHandler = true;
Aws::InitAPI(options);
{
    // make your SDK calls here.
}
Aws::ShutdownAPI(options);
```

`installSigPipeHandler`가 `true`인 경우 SDK for C++에서는 SIGPIPE 신호를 무시하는 핸들러를 사용합니다. SIGPIPE에 대한 자세한 내용은 GNU 운영 체제 웹 사이트에서 [Operation Error Signals](#)를 참조하세요. curl 핸들러에 대한 자세한 내용은 curl 웹 사이트에 설명된 [CURLOPT\\_NOSIGNAL](#)을 참조하세요.

curl 및 OpenSSL의 기본 라이브러리는 원격 측이 연결을 종료할 때 SIGPIPE 신호를 전송하여 알릴 수 있습니다. 이러한 신호는 애플리케이션에서 처리해야 합니다. 이 curl 기능에 대한 자세한 내용은 curl 웹 사이트에서 [libcurl thread safety](#)를 참조하세요. 신호 핸들러는 각 애플리케이션에 전역적으로 적용되며 라이브러리는 SDK의 종속성이므로 이 동작은 SDK에 자동으로 내장되지 않습니다.

## 외부에서 SDK for C++ 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 프로필이라는 별도의 설정 세트를 유지하여 다양한 환경 또는 테스트에 대해 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 방법을 통해 애플리케이션을 구성하는 방법과 교차 SDK 설정에 대한 자세한 정보를 알아보려면 AWS SDK 및 도구 참조 안내서를 참조하세요. 환경 변수 또는 구성 파일에서 SDK가 확인할 수 있는 모든 설정을 보려면 AWS SDK 및 도구 참조 안내서에 나와 있는 [설정 참조](#)를 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 안내서에서 [환경 변수 지원](#)을 참조하세요.
  - 셸 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 안내서에 나와 있는 [공유 config 및 credentials 파일](#)을 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.

#### Note

환경 변수 또는 구성 파일에서 SDK가 확인할 수 있는 모든 설정을 보려면 AWS SDK 및 도구 참조 안내서에 나와 있는 [설정 참조](#)를 참조하세요.

## 코드에서 AWS SDK for C++ 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 구성에 대한 옵션이 있습니다.

AWS SDK for C++에는 애플리케이션에서 사용하는 AWS 서비스와 상호 작용하는 기능을 제공하는 AWS 서비스 클라이언트 클래스가 포함되어 있습니다. SDK for C++에서는 기본 클라이언트 구성을 변경할 수 있으며, 이는 다음과 같은 작업을 수행할 때 유용합니다.

- 프록시를 통해 인터넷에 연결
- 연결 제한 시간 및 요청 재시도 등 HTTP 전송 설정 변경
- TCP 소켓 버퍼 크기 힌트 지정

ClientConfiguration는 코드에서 인스턴스화하고 활용할 수 있는 SDK for C++의 구조입니다. 다음 스니펫은 이 클래스를 사용하여 프록시를 통해 Amazon S3에 액세스하는 방법을 보여줍니다.

```
Aws::Client::ClientConfiguration clientConfig;
clientConfig.proxyHost = "localhost";
clientConfig.proxyPort = 1234;
clientConfig.proxyScheme = Aws::Http::Scheme::HTTPS;
Aws::S3::S3Client(clientConfig);
```

## 구성 변수 선언

ClientConfiguration 구조체는 다음과 같이 멤버 변수를 선언합니다.

```
Aws::String accountId;
Aws::String accountIdEndpointMode = "preferred";
bool allowSystemProxy = false;
Aws::String appId;
Aws::String caPath;
Aws::String caFile;

struct {
    RequestChecksumCalculation requestChecksumCalculation =
        RequestChecksumCalculation::WHEN_SUPPORTED;
    ResponseChecksumValidation responseChecksumValidation =
        ResponseChecksumValidation::WHEN_SUPPORTED;
} checksumConfig;

ProviderFactories configFactories = ProviderFactories::defaultFactories;
long connectTimeoutMs = 1000;

struct CredentialProviderConfiguration {
    Aws::String profile;
    Aws::String region;
```

```

    struct {
        long metadataServiceNumAttempts = 1;
        long metadataServiceTimeout = 1;
        std::shared_ptr<RetryStrategy> imdsRetryStrategy;
        bool disableImdsV1;
        bool disableImds;
    } imdsConfig;
    struct STSCredentialsCredentialProviderConfiguration {
        Aws::String roleArn;
        Aws::String sessionName;
        Aws::String tokenFilePath;
        std::chrono::milliseconds retrieveCredentialsFutureTimeout =
std::chrono::seconds(10);
    } stsCredentialsProviderConfig;
} credentialProviderConfig;

bool disableExpectHeader = false;
bool disableIMDS = false;
bool disableImdsV1 = false;
bool enableClockSkewAdjustment = true;
Aws::CRT::Optional<bool> enableEndpointDiscovery;
bool enableHostPrefixInjection = true;
bool enableHttpClientTrace = false;
bool enableTcpKeepAlive = true;
Aws::String endpointOverride;
std::shared_ptr<Aws::Utils::Threading::Executor> executor = nullptr;
FollowRedirectsPolicy followRedirects;
Aws::Http::TransferLibType httpLibOverride;
Aws::Http::TransferLibPerformanceMode httpLibPerfMode =
    Http::TransferLibPerformanceMode::LOW_LATENCY;
long httpRequestTimeoutMs = 0;
unsigned long lowSpeedLimit = 1;
unsigned maxConnections = 25;
Aws::Utils::Array<Aws::String> nonProxyHosts;
Aws::String profileName;
Aws::String proxyCaFile;
Aws::String proxyCaPath;
Aws::Http::Scheme proxyScheme;
Aws::String proxyHost;
unsigned proxyPort = 0;
Aws::String proxyUserName;
Aws::String proxyPassword;
Aws::String proxySSLCertPath;
Aws::String proxySSLCertType;

```

```

Aws::String proxySSLKeyPath;
Aws::String proxySSLKeyType;
Aws::String proxySSLKeyPassword;
std::shared_ptr<Aws::Utils::RateLimits::RateLimiterInterface> readRateLimiter =
    nullptr;
Aws::String region;
Aws::Client::RequestCompressionConfig requestCompressionConfig;
long requestTimeoutMs = 0;
std::shared_ptr<RetryStrategy> retryStrategy = nullptr;
Aws::Http::Scheme scheme;
unsigned long tcpKeepAliveIntervalMs = 30000;
std::shared_ptr<smithy::components::tracing::TelemetryProvider> telemetryProvider;
Aws::String userAgent;
bool useDualStack = false;
bool useFIPS = false;
bool verifySSL = true;
Aws::Http::Version version = Http::Version::HTTP_VERSION_2TLS;

struct WinHTTPOptions {
    bool useAnonymousAuth = false;
} winHTTPOptions;

std::shared_ptr<Aws::Utils::RateLimits::RateLimiterInterface> writeRateLimiter =
    nullptr;

static Aws::String LoadConfigFromEnvOrProfile(const Aws::String& envKey, const
    Aws::String& profile,
    const Aws::String& profileProperty, const
    Aws::Vector<Aws::String>& allowedValues,
    const Aws::String& defaultValue);

```

## 구성 변수 설명

다음 목록은 클라이언트 동작을 사용자 지정하는 데 사용할 수 있는 ClientConfiguration 멤버 변수를 설명합니다.

### accountId

계정 기반 엔드포인트 라우팅의 AWS 계정 ID를 지정합니다. 111122223333 형식을 사용합니다. 계정 기반 엔드포인트 라우팅을 사용하면 일부 서비스의 요청 성능이 향상됩니다.

## accountIdEndpointMode

계정 기반 엔드포인트 라우팅 동작을 제어합니다. 유효한 값은 "required", "disabled", "preferred" 중 하나입니다. 기본값은 "preferred"입니다. 필요한 경우 계정 기반 엔드포인트 라우팅을 끄려면 "disabled"로 설정합니다.

## allowSystemProxy

HTTP 클라이언트에서 시스템 프록시 설정을 검색할지 여부를 제어합니다. 기본 설정은 false입니다. 자동 프록시 검색을 활성화하려면 true로 설정합니다.

## appId

애플리케이션별 식별자(선택 사항)를 지정합니다. 설정할 경우 이 값이 User-Agent 헤더에 App/{appId} 형식으로 추가됩니다. 이 값은 AWS\_SDK\_UA\_APP\_ID 환경 변수 또는 sdk\_ua\_app\_id 공유 구성 프로파일 속성을 사용하여 설정할 수 있습니다.

## caPath, caFile

HTTP 클라이언트에 SSL 인증서 트러스트 스토어를 찾을 위치를 지시합니다. 예제 트러스트 스토어는 OpenSSL c\_rehash 유틸리티로 준비된 디렉터리일 수 있습니다. 환경에서 symlink를 사용하지 않는 한 이러한 변수를 설정할 필요가 없습니다. Windows 및 macOS 시스템에서는 이 변수들이 아무런 영향을 미치지 않습니다.

## checksumConfig

체크섬 계산 및 검증 설정을 포함합니다. requestChecksumCalculation 및 responseChecksumValidation으로 구성되며 기본값은 WHEN\_SUPPORTED입니다.

## configFactories

클라이언트 유틸리티 클래스(Executor 및 RetryStrategy)를 초기화하기 위한 팩토리 메서드를 지정합니다. 재정의하지 않는 한 기본 팩토리가 사용됩니다.

## requestTimeoutMs 및 connectTimeoutMs

HTTP 요청의 제한 시간까지 대기할 시간(밀리초)을 지정합니다. 예를 들어, 대용량 파일을 전송할 경우에는 이 시간을 늘리는 것이 좋습니다.

## credentialProviderConfig

자격 증명 공급자에 대한 구성 설정을 포함합니다. 이 구조를 사용하여 SDK에서 AWS 자격 증명을 얻는 방법을 사용자 지정합니다.

## disableExpectHeader

CURL HTTP 클라이언트에만 적용됩니다. 기본적으로 CURL은 HTTP 헤더 수신 직후 서버가 오류로 응답하는 상황에서 HTTP 페이로드 전송을 방지하기 위해 HTTP 요청에 "Expect: 100-Continue" 헤더를 추가합니다. 이 동작으로 왕복 통신을 절약할 수 있으며 페이로드가 작고 네트워크 지연 시간이 우려되는 상황에서 유용합니다. 이 변수의 기본 설정은 false입니다. true로 설정하면 CURL에 HTTP 요청 헤더와 본문 페이로드를 모두 함께 전송하도록 지시합니다.

## disableIMDS

인스턴스 메타데이터 서비스(IMDS) 호출을 비활성화할지 여부를 제어합니다. 기본 설정은 false입니다. EC2 인스턴스 외부에서 실행될 때 IMDS 호출을 비활성화하려면 true로 설정합니다.

## disableImdsV1

IMDSv2 호출은 허용하면서 IMDSv1 호출을 비활성화할지 여부를 제어합니다. 기본 설정은 false입니다. 보안 강화를 위해 IMDSv1 호출만 비활성화하려면 true로 설정합니다.

## enableClockSkewAdjustment

매 HTTP 시도 후 클록 스큐를 조정할지 여부를 제어합니다. 기본 설정은 false입니다.

## enableEndpointDiscovery

엔드포인트 검색 사용 여부를 제어합니다. 기본적으로 리전 엔드포인트 또는 재정의된 엔드포인트가 사용됩니다. 엔드포인트 검색을 활성화하려면 이 변수를 true로 설정합니다.

## enableHostPrefixInjection

HTTP 호스트가 DiscoverInstances 요청에 "data-" 접두사를 추가할지 여부를 제어합니다. 기본적으로 이 동작은 활성화되어 있습니다. 비활성화하려면 이 변수를 false로 설정합니다.

## enableHttpClientTrace

디버깅을 위해 HTTP 클라이언트 추적을 활성화할지 여부를 제어합니다. 기본 설정은 false입니다. 자세한 HTTP 요청 및 응답 로깅을 활성화하려면 true로 설정합니다.

## enableTcpKeepAlive

TCP 연결 유지 패킷을 전송할지 여부를 제어합니다. 기본 설정은 true입니다.

tcpKeepAliveIntervalMs 변수와 함께 사용합니다. WinINet 및 IXMLHttpRequest2 클라이언트에는 이 변수를 적용할 수 없습니다.

## endpointOverride

서비스와 통신할 때 사용할 재정의 HTTP 엔드포인트를 지정합니다.

## executor

비동기식 Executor 핸들러의 구현을 참조합니다. 기본 동작은 각 비동기 호출마다 스레드를 생성하고 분리하는 것입니다. 이 동작을 변경하려면 Executor 클래스의 하위 클래스를 구현하고 해당 인스턴스를 이 변수에 할당합니다.

## followRedirects

HTTP 300 리디렉션 코드 처리 시 동작을 제어합니다.

## httpLibOverride

기본 HTTP 팩토리에서 반환하는 HTTP 구현을 지정합니다. Windows의 기본 HTTP 클라이언트는 WinHTTP입니다. 다른 모든 플랫폼의 기본 HTTP 클라이언트는 CURL입니다.

## httpLibPerfMode

HTTP 라이브러리 성능 모드를 지정합니다. 기본 설정은 LOW\_LATENCY입니다. 이 설정을 조정하여 다양한 성능 특성에 맞게 최적화할 수 있습니다.

## httpRequestTimeoutMs

HTTP 요청 제한 시간(밀리초)을 지정합니다. 기본값은 0(제한 시간 없음)입니다. 대용량 파일을 전송할 경우에는 이 값을 늘리는 것이 좋습니다.

## lowSpeedLimit

허용되는 최소 전송 속도(바이트/초)를 지정합니다. 전송 속도가 지정된 속도 아래로 떨어지면 전송 작업이 중단됩니다. 기본 설정은 1바이트/초입니다. 이 변수는 CURL 클라이언트에만 적용됩니다.

## maxConnections

단일 서버에 대한 최대 HTTP 연결 수를 지정합니다. 기본값은 25입니다. 대역폭이 합리적으로 지원하는 범위 외에는 허용되는 최대값이 없습니다.

## nonProxyHosts

프록시 설정을 우회해야 하는 호스트 이름 배열을 지정합니다. 프록시 구성에서 특정 호스트를 제외하려면 이 설정을 사용합니다.

## profileName

구성에 사용할 AWS 프로필 이름을 지정합니다. SDK는 AWS 구성 파일의 지정된 프로필에서 설정을 로드합니다.

## proxyCaFile

기본값과 다른 경우 프록시 연결용 인증 기관 파일의 경로를 지정합니다.

## proxyCaPath

기본값과 다른 경우 프록시 연결용 인증 기관 트러스트 스토어의 경로를 지정합니다.

proxyScheme, proxyHost, proxyPort, proxyUserName 및 proxyPassword

AWS와의 모든 통신에 대한 프록시를 설정하고 구성하는 데 사용됩니다. 이 기능이 유용할 수 있는 예시로는 Burp Suite와 함께 디버깅을 수행하거나, 프록시를 사용하여 인터넷에 연결하는 경우가 있습니다.

## proxySSLCertPath

클라이언트 인증서가 필요한 프록시 연결용 SSL 인증서 파일의 경로를 지정합니다.

## proxySSLCertType

프록시 연결에 사용할 SSL 인증서 유형을 지정합니다. 일반적인 유형으로는 PEM과 DER이 있습니다.

## proxySSLKeyPassword

키가 암호로 보호된 경우 프록시 연결에 사용되는 SSL 프라이빗 키의 암호를 지정합니다.

## proxySSLKeyPath

클라이언트 인증서가 필요한 프록시 연결용 SSL 프라이빗 키 파일의 경로를 지정합니다.

## proxySSLKeyType

프록시 연결에 사용할 SSL 프라이빗 키 유형을 지정합니다. 일반적인 유형으로는 PEM과 DER이 있습니다.

## writeRateLimiter 및 readRateLimiter

전송 계층에서 사용하는 대역폭을 스로틀링하는 데 사용되는 읽기 및 쓰기 속도 제한기 구현에 대한 참조입니다. 기본적으로 읽기 및 쓰기 속도는 스로틀링되지 않습니다. 스로틀링을 도입하려면 `RateLimiterInterface`의 하위 클래스를 구현하고 해당 인스턴스를 이 변수에 할당합니다.

## region

사용할 AWS 리전을 지정합니다(예: us-east-1). 기본적으로 사용되는 리전은 해당 AWS 자격 증명에 구성된 기본 리전입니다.

## requestCompressionConfig

요청 압축에 대한 구성 설정을 포함합니다. 이 구조를 사용하여 전송 전에 요청이 압축되는 시기와 방식을 제어합니다.

## retryStrategy

재시도 전략의 구현을 참조합니다. 기본 전략은 지수 백오프 정책을 구현합니다. 다른 전략을 수행하려면 `RetryStrategy` 클래스의 하위 클래스를 구현하고 해당 인스턴스를 이 변수에 할당합니다.

## scheme

URI 주소 지정 체계를 HTTP 또는 HTTPS로 지정합니다. 기본 체계는 HTTPS입니다.

## tcpKeepAliveIntervalMs

TCP 연결을 통해 연결 유지 패킷을 전송할 시간 간격(밀리초)을 지정합니다. 기본 간격은 30초입니다. 최소 설정은 15초입니다. WinINet 및 IXMLHttpRequest2 클라이언트에는 이 변수를 적용할 수 없습니다.

## telemetryProvider

지표 및 추적 데이터를 수집하기 위한 원격 측정 공급자 구현을 참조합니다. 관찰성 특성을 활성화하려면 이 설정을 구성합니다.

## userAgent

내부용입니다. 이 변수의 설정은 변경하지 않습니다.

## useDualStack

듀얼 스택 IPv4 및 IPv6 엔드포인트를 사용할지 여부를 제어합니다. 모든 AWS 서비스가 모든 리전에서 IPv6를 지원하는 것은 아닙니다.

## useFIPS

Federal Information Processing Standards(FIPS) 140-2로 검증된 암호화 모듈을 사용할지 여부를 제어합니다. 기본 설정은 `false`입니다. FIPS 규정 준수가 필요한 경우 `true`로 설정합니다.

## verifySSL

SSL 인증서 확인 여부를 제어합니다. 기본적으로 SSL 인증서를 확인합니다. 확인을 비활성화하려면 변수를 `false`로 설정합니다.

## version

요청에 사용할 HTTP 버전을 지정합니다. 기본 설정은 `HTTP_VERSION_2TLS`(TLS를 통한 HTTP/2)입니다.

## winHTTPOptions

Windows별 HTTP 구성 옵션을 포함합니다. 기본 설정이 false인 useAnonymousAuth를 포함합니다.

## AWS SDK for C++의 AWS 리전 설정

AWS 리전을 사용하여 특정 지역에서 작동하는 AWS 서비스에 액세스할 수 있습니다. 이는 중복성은 물론, 데이터와 애플리케이션을 고객과 고객의 사용자가 액세스할 위치에 가까운 곳에서 실행 상태로 유지하는 데도 유용할 수 있습니다.

### Important

대부분의 리소스는 특정 AWS 리전에 있으므로 SDK를 사용할 때 리소스의 올바른 리전을 제공해야 합니다.

공유 AWS config 파일 또는 환경 변수를 통해 기본 리전을 설정하는 방법에 대한 예는 AWS SDK 및 도구 참조 안내서에서 [AWS 리전](#) 섹션을 참조하세요.

AWS 요청에 사용할 AWS SDK for C++의 기본 AWS 리전을 설정해야 합니다. 이 기본값은 리전이 지정되지 않은 모든 SDK 서비스 메서드 직접 호출에 사용됩니다. SDK for C++에서는 [코드 내 클라이언트 구성](#)을 사용하여 기본 리전을 설정할 수도 있습니다.

## AWS SDK for C++ 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 액세스](#)를 참조하세요.

## 자격 증명 공급자 체인

클라이언트를 구성할 때 자격 증명 공급자를 명시적으로 지정하지 않으면 SDK for C++는 자격 증명을 제공할 수 있는 일련의 위치를 확인하는 자격 증명 공급자 체인을 사용합니다. 이러한 위치 중 하나에서 자격 증명을 찾으면 검색이 중지됩니다.

### 자격 증명 가져오기 순서

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 자격 증명 공급자 체인이라고 합니다.

체인의 각 단계마다 값을 설정하는 다양한 방법이 있습니다. 코드에서 직접 값을 설정하는 것이 항상 우선하며, 환경 변수로 설정한 다음 공유 AWS config 파일에서 설정합니다. 자세한 내용은 AWS SDK 및 도구 참조 안내서의 [Precedence of settings](#)를 참조하세요.

SDK는 공유 AWS config 및 credentials 파일의 [default] 프로필에서 자격 증명을 로드하려고 시도합니다. AWS\_PROFILE 환경 변수를 사용하여 [default]를 사용하는 대신 SDK가 로드할 명명된 프로파일을 선택할 수 있습니다. config 및 credentials 파일은 AWS SDKs 및 도구에서 공유됩니다. AWS SDKs 및 도구 참조 가이드에는 모든 AWS SDKs 및 도구에서 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 SDK for C++ AWS는 자격 증명 공급자를 다음 순서로 확인합니다.

#### 1. AWS 액세스 키(임시 및 장기 자격 증명)

SDK는 AWS\_ACCESS\_KEY\_ID 및 AWS\_SECRET\_ACCESS\_KEY, AWS\_SESSION\_TOKEN 환경 변수 또는 공유 AWS credentials 파일에서 자격 증명을 로드하려고 시도합니다.

- 이 공급자 구성에 대한 지침은 AWS SDK 및 도구 참조 안내서에서 [AWS 액세스 키](#)를 참조하세요.
- 이 공급자의 SDK 구성 속성에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서에서 [AWS 액세스 키](#)를 참조하세요.

#### 2. AWS STS 웹 자격 증명

액세스가 필요한 모바일 애플리케이션 또는 클라이언트 기반 웹 애플리케이션을 생성할 때 AWS AWS Security Token Service (AWS STS)는 퍼블릭 자격 증명 공급자(IdP)를 통해 인증된 페더레이션 사용자를 위한 임시 보안 자격 증명 세트를 반환합니다.

- 프로필에서 이를 지정하면 SDK 또는 도구가 AssumeRoleWithWebIdentity API 메서드를 사용하여 AWS STS 임시 자격 증명을 검색하려고 시도합니다. 이 메서드에 관한 자세한 정보는 AWS Security Token Service API 참조의 [AssumeRoleWithWebIdentity](#)를 참조하세요.
- 이 공급자 구성에 대한 지침은 AWS SDK 및 도구 참조 안내서의 [웹 자격 증명 또는 OpenID Connect로 페더레이션](#)을 참조하세요.
- 이 공급자의 SDK 구성 속성에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서에서 [역할 자격 증명 공급자 수임](#)을 참조하세요.

### 3. IAM Identity Center

IAM Identity Center를 사용하여 인증하는 경우 C++용 SDK가 AWS CLI 명령을 실행하여 설정한 Single Sign-On 토큰을 사용하는 경우입니다. SDK는 IAM Identity Center가 유효한 토큰으로 교환된 임시 자격 증명을 사용합니다. 그러면 AWS 서비스를 직접 호출할 때 이 임시 자격 증명이 사용됩니다. 이 프로세스에 관한 자세한 내용은 AWS SDK 및 도구 참조 안내서의 [AWS 서비스에 대한 SDK 자격 증명 해결 이해](#)를 참조하세요.

- 이 공급자 구성에 대한 지침은 AWS SDK 및 도구 참조 안내서의 [IAM Identity Center 인증](#)을 참조하세요.
- 이 공급자의 SDK 구성 속성에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서의 [IAM Identity Center 자격 증명 공급자](#)를 참조하세요.

### 4. AWS 로그인을 사용한 로그인 자격 증명 자격 증명 해석기

AWS 로그인 및 콘솔 자격 증명을 사용하여 인증하는 경우, SDK for C++가 `aws login` 또는 `CLaws login --profile`에서 실행하여 설정한 콘솔 자격 증명을 사용하는 경우입니다. SDK는 AWS 서비스를 호출할 때 이러한 자격 증명을 사용합니다.

- 이 프로세스에 대한 자세한 내용은 SDK 및 도구 참조 안내서의 [콘솔 자격 증명을 사용하여 AWS 로컬 개발을 위한 로그인](#)을 참조하세요. AWS SDKs

### 5. 외부 프로세스 공급자

이 공급자를 사용하면 온프레미스 자격 증명 저장소에서 자격 증명을 가져오거나, 온프레미스 ID 공급자와 통합하는 등 사용자 지정 구현을 제공할 수 있습니다.

- 이 공급자를 구성하는 한 가지 방법에 대한 지침은 AWS SDK 및 도구 참조 안내서에서 [IAM Roles Anywhere](#)를 참조하세요.
- 이 공급자의 SDK 구성 속성에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서에서 [프로세스 자격 증명 공급자](#)를 참조하세요.

### 6. Amazon ECS 및 Amazon EKS 컨테이너 자격 증명

Amazon Elastic Container Service 작업과 Kubernetes 서비스 계정은 이와 연결된 IAM 역할이 있을 수 있습니다. IAM 역할에서 부여된 권한은 작업에서 실행 중인 컨테이너 또는 포드의 컨테이너에 위임됩니다. 이 역할을 통해 컨테이너 내 SDK for C++ 애플리케이션 코드에 다른 AWS 서비스 서비스를 사용할 수 있습니다.

SDK는 Amazon ECS 및 Amazon EKS에서 자동으로 설정할 수 있는 `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` 또는 `AWS_CONTAINER_CREDENTIALS_FULL_URI` 환경 변수에서 자격 증명을 검색하려고 시도합니다.

- Amazon ECS에서 이 역할을 설정하는 방법에 관한 자세한 내용은 Amazon Elastic Container Service 개발자 안내서의 [Amazon ECS 작업 IAM 역할](#)을 참조하세요.
- Amazon EKS 설정 정보는 Amazon EKS 사용 설명서의 [Amazon EKS Pod Identity Agent 설정](#)을 참조하세요.
- 이 공급자의 SDK 구성 속성에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서의 [컨테이너 자격 증명 공급자](#)를 참조하세요.

## 7. Amazon EC2 인스턴스 메타데이터 서비스

IAM 역할을 생성하여 인스턴스에 연결합니다. 인스턴스의 SDK for C++ 애플리케이션은 역할이 제공하는 자격 증명을 인스턴스 메타데이터에서 검색하려고 시도합니다.

- 이 역할 설정과 메타데이터 사용에 대한 자세한 내용은 Amazon EC2 사용 설명서에서 [Amazon EC2의 IAM 역할](#) 및 [인스턴스 메타데이터 작업](#)을 참조하세요.
- 이 공급자의 SDK 구성 속성에 대한 자세한 내용은 AWS SDK 및 도구 참조 안내서의 [IMDS 자격 증명 공급자](#)를 참조하세요.

자격 증명 공급자 체인은 GitHub의 AWS SDK for C++ 소스 코드 [AWSCredentialsProviderChain](#)에 있는에서 검토할 수 있습니다.

새 사용자가 시작할 수 있도록 권장 접근 방식을 따른 경우 시작하기 주제 [AWS SDK for C++를 AWS 사용하여 로 인증](#) 중에 AWS 로그인 자격 증명 인증을 설정합니다. 상황에 따라 다른 인증 방법이 유용할 수 있습니다. 보안 위험을 방지하려면 항상 단기 보안 인증을 사용하는 것이 좋습니다. 다른 인증 방법 절차에 대해서는 AWS SDK 및 도구 참조 안내서의 [Authentication and access](#)를 참조하세요.

## 명시적 자격 증명 공급자

자격 증명 공급자 체인을 사용하여 인증 방법을 감지하는 대신 SDK에서 사용해야 하는 특정 자격 증명 공급자를 지정할 수 있습니다. 이 작업은 서비스 클라이언트의 생성자에 자격 증명을 제공하여 수행할 수 있습니다.

다음 예제에서는 체인을 사용하는 대신 임시 액세스 자격 증명을 직접 제공하여 Amazon Simple Storage Service 클라이언트를 생성합니다.

```
SDKOptions options;
Aws::InitAPI(options);
{
    const auto cred_provider =
    Aws::MakeShared<Auth::SimpleAWSCredentialsProvider>("TestAllocationTag",
        "awsAccessKeyId",
        "awsSecretKey",
        "sessionToken");
    S3Client client{cred_provider};
}
Aws::ShutdownAPI(options);
```

## ID 캐싱

SDK는 자격 증명 및 SSO 토큰과 같은 기타 자격 증명 유형을 캐싱합니다. 기본적으로 SDK는 지연 캐시 구현을 사용합니다. 이 구현은 첫 번째 요청 시 자격 증명을 로드하고 캐시한 후 만료 시점이 가까워지면 다른 요청 중에 이를 새로 고치려고 시도합니다. 동일한 [Aws::Client::ClientConfiguration](#)에서 생성된 클라이언트는 캐시를 공유합니다.

## AWS SDK for C++ 빌드용 CMake 파라미터

이 섹션에 나열된 [CMake](#) 파라미터를 사용하여 SDK 빌드 방식을 사용자 지정합니다.

이러한 옵션은 CMake GUI 도구를 사용하거나 명령줄에서 -D를 사용하여 설정할 수 있습니다. 예를 들면 다음과 같습니다.

```
cmake -DENABLE_UNITY_BUILD=ON -DREGENERATE_CLIENTS=1
```

## 일반 CMake 변수 및 옵션

다음은 SDK 소스 코드의 빌드 프로세스에 영향을 미치는 일반 **cmake** 변수 및 옵션입니다.

### Note

이러한 파라미터는 SDK for C++ 자체의 소스 코드를 빌드할 때 사용합니다.

## 주제

- [ADD\\_CUSTOM\\_CLIENTS](#)
- [AUTORUN\\_UNIT\\_TESTS](#)
- [AWS\\_AUTORUN\\_LD\\_LIBRARY\\_PATH](#)
- [AWS\\_SDK\\_WARNINGS\\_ARE\\_ERRORS](#)
- [AWS\\_USE\\_CRYPTO\\_SHARED\\_LIBS](#)
- [AWS\\_TEST\\_REGION](#)
- [BUILD\\_BENCHMARKS](#)
- [BUILD\\_DEPS](#)
- [BUILD\\_ONLY](#)
- [BUILD\\_OPTEL](#)
- [BUILD\\_SHARED\\_LIBS](#)
- [BYPASS\\_DEFAULT\\_PROXY](#)
- [CPP\\_STANDARD](#)
- [CURL\\_INCLUDE\\_DIR](#)
- [CURL\\_LIBRARY](#)
- [CUSTOM\\_MEMORY\\_MANAGEMENT](#)
- [DISABLE\\_INTERNAL\\_IMDSV1\\_CALLS](#)
- [ENABLE\\_ADDRESS\\_SANITIZER](#)
- [ENABLE\\_CURL\\_LOGGING](#)
- [ENABLE\\_HTTP\\_CLIENT\\_TESTING](#)
- [ENABLE\\_RTTI](#)
- [ENABLE\\_TESTING](#)
- [ENABLE\\_UNITY\\_BUILD](#)
- [ENABLE\\_VIRTUAL\\_OPERATIONS](#)
- [ENABLE\\_ZLIB\\_REQUEST\\_COMPRESSION](#)
- [FORCE\\_CURL](#)
- [FORCE\\_SHARED\\_CRT](#)
- [G](#)
- [MINIMIZE\\_SIZE](#)

- [NO\\_ENCRYPTION](#)
- [NO\\_HTTP\\_CLIENT](#)
- [REGENERATE\\_CLIENTS](#)
- [REGENERATE\\_DEFAULTS](#)
- [SIMPLE\\_INSTALL](#)
- [TARGET\\_ARCH](#)
- [USE\\_CRT\\_HTTP\\_CLIENT](#)
- [USE\\_IXML\\_HTTP\\_REQUEST\\_2](#)
- [USE\\_OPENSSL](#)
- [USE\\_TLS\\_V1\\_2](#)
- [USE\\_TLS\\_V1\\_3](#)

## ADD\_CUSTOM\_CLIENTS

API 정의를 기반으로 임의의 클라이언트를 빌드합니다. 정의를 `code-generation/api-definitions` 폴더에 배치한 후 이 인수를 **cmake**에 전달하세요. **cmake** 성 단계에서 클라이언트를 생성하여 빌드에 하위 디렉터리로 포함시킵니다. 이는 특히 [API Gateway](#) 서비스 중 하나를 사용하기 위해 C++ 클라이언트를 생성할 때 유용합니다. 예:

```
-
DADD_CUSTOM_CLIENTS="serviceName=myCustomService,version=2015-12-21;serviceName=someOtherService"
```

### Note

ADD\_CUSTOM\_CLIENTS 파라미터를 사용하려면 [Python 2.7](#), Java([JDK 1.8 이상](#)) 및 [Maven](#)이 설치되어 있고 PATH에 포함되어 있어야 합니다.

## AUTORUN\_UNIT\_TESTS

ON인 경우 빌드 후 단위 테스트를 자동으로 실행합니다.

값

ON | OFF

## 기본값

ON

## AWS\_AUTORUN\_LD\_LIBRARY\_PATH

CMake로 자동 실행되는 유닛 테스트를 위해 LD\_LIBRARY\_PATH에 추가할 경로입니다. 재정의된 종속성에 사용자 지정 런타임 라이브러리가 필요한 경우에 이 경로를 설정합니다.

값

(문자열) -

기본값

N/A

## AWS\_SDK\_WARNINGS\_ARE\_ERRORS

ON인 경우 컴파일러 경고를 오류로 처리합니다. 새 컴파일러나 흔하지 않은 컴파일러에서 오류가 관찰되는 경우 이를 OFF로 설정해 보세요.

값

ON | OFF

기본값

ON

## AWS\_USE\_CRYPTO\_SHARED\_LIBS

FindCrypto가 공유 암호화 라이브러리를 사용하도록 강제합니다(해당 라이브러리가 발견된 경우). 대신 [BUILD\\_SHARED\\_LIBS](#) 설정을 사용하려면 이 옵션을 OFF로 설정합니다.

값

ON | OFF

기본값

OFF

## AWS\_TEST\_REGION

통합 테스트에 사용할 AWS 리전입니다.

값

(문자열) -

기본값

N/A

## BUILD\_BENCHMARKS

ON인 경우 벤치마크 실행 파일을 빌드합니다.

값

ON | OFF

기본값

OFF

## BUILD\_DEPS

ON인 경우 타사 종속성을 빌드합니다.

값

ON | OFF

기본값

ON

## BUILD\_ONLY

사용할 클라이언트만 빌드합니다. `aws-cpp-sdk-transfer`와 같은 상위 수준 SDK로 설정된 경우 `BUILD_ONLY`는 하위 수준 클라이언트 종속성을 모두 확인합니다. 또한 선택한 프로젝트와 관련된 통합 및 유닛 테스트가 있는 경우, 이들도 빌드합니다. 이는 세미콜론(;) 문자로 값이 구분된 목록 인수입니다. 예:

```
-DBUILD_ONLY="s3;cognito-identity"
```

#### Note

핵심 SDK 모듈인 `aws-sdk-cpp-core`는 `BUILD_ONLY` 매개변수의 값과 상관없이 항상 빌드됩니다.

## BUILD\_OPTEL

ON인 경우 추적을 위한 오픈 텔레메트리 구현을 빌드합니다.

값

ON | OFF

기본값

OFF

## BUILD\_SHARED\_LIBS

기본 제공 CMake 옵션으로, 가시성을 위해 여기에 재표시됩니다. ON인 경우 공유 라이브러리를 빌드하며, 그렇지 않은 경우 정적 라이브러리만 빌드합니다.

#### Note

SDK에 동적으로 연결하려면 SDK를 사용하여 모든 빌드 대상에 대해 `USE_IMPORT_EXPORT` 기호를 정의해야 합니다.

값

ON | OFF

기본값

ON

## BYPASS\_DEFAULT\_PROXY

ON인 경우 `IXmlHttpRequest2`를 사용할 때 시스템의 기본 프록시 설정을 우회합니다.

값

ON | OFF

기본값

ON

## CPP\_STANDARD

C++ 14 및 17 코드 베이스와 함께 사용할 사용자 지정 C++ 표준을 지정합니다.

값

11 | 14 | 17

기본값

11

## CURL\_INCLUDE\_DIR

`libcurl` 헤더가 포함된 `curl include` 디렉터리의 경로입니다.

값

선택한 *include* 디렉터리의 문자열 경로입니다. 예: `D:/path/to/dir/with/curl/include`

기본값

N/A

## CURL\_LIBRARY

연결할 `curl` 라이브러리 파일의 경로입니다. 이 라이브러리는 애플리케이션의 요구 사항에 따라 정적 라이브러리 또는 가져오기 라이브러리일 수 있습니다.

값

curl 라이브러리 파일의 문자열 경로. 예: D:/path/to/static/libcurl/file/ie/libcurl.lib.a

기본값

N/A

## CUSTOM\_MEMORY\_MANAGEMENT

사용자 지정 메모리 관리자를 사용하려면 값을 1로 설정합니다. 모든 STL 유형에 사용자 지정 할당 인터페이스를 사용하도록 사용자 지정 할당기를 설치할 수 있습니다. 값을 0으로 설정하더라도 Windows에서 DLL 안전성을 위해 STL 템플릿 유형을 계속 사용할 수 있습니다.

정적 연결이 ON인 경우 사용자 지정 메모리 관리는 기본적으로 off(0)로 설정됩니다. 동적 연결이 ON인 경우 사용자 지정 메모리 관리는 기본적으로 on(1)으로 설정되며 교차 DLL 할당 및 할당 해제가 방지됩니다.

### Note

링커 불일치 오류를 방지하려면 빌드 시스템 전체에서 동일한 값(0 또는 1)을 사용해야 합니다.

SDK에서 수행한 할당을 처리할 자체 메모리 관리자를 설치하려면 SDK에 의존하는 모든 빌드 대상에 대해 -DCUSTOM\_MEMORY\_MANAGEMENT를 설정하고 USE\_AWS\_MEMORY\_MANAGEMENT를 정의해야 합니다.

## DISABLE\_INTERNAL\_IMDSV1\_CALLS

ON인 경우 [인스턴스 메타데이터 서비스](#)의 V1 API에 대한 내부 호출이 수행되지 않습니다. OFF인 경우 IMDSv2 호출이 실패하면 IMDSv2 호출이 IMDSv1 사용으로 대체됩니다. IMDSv1 및 IMDSv2에 대한 자세한 내용은 Amazon EC2 사용 설명서에서 [인스턴스 메타데이터 서비스를 사용하여 인스턴스 메타데이터에 액세스](#)를 참조하세요.

값

ON | OFF

기본값

OFF

## ENABLE\_ADDRESS\_SANITIZER

ON인 경우 gcc 또는 clang에 대해 Address Sanitizer를 켭니다.

값

ON | OFF

기본값

OFF

## ENABLE\_CURL\_LOGGING

ON인 경우 curl의 내부 로그를 SDK 로거로 파이프 처리합니다.

값

ON | OFF

기본값

OFF

## ENABLE\_HTTP\_CLIENT\_TESTING

ON인 경우 해당 HTTP 클라이언트 테스트 제품군을 빌드하고 실행합니다.

값

ON | OFF

기본값

OFF

## ENABLE\_RTTI

런타임 유형 정보(RTTI)를 활성화하도록 SDK를 빌드할지 여부를 제어합니다.

값

ON | OFF

## 기본값

ON

## ENABLE\_TESTING

SDK 빌드 중 유닛 테스트 및 통합 테스트 프로젝트를 빌드할지 여부를 제어합니다.

값

ON | OFF

## 기본값

ON

## ENABLE\_UNITY\_BUILD

ON인 경우 대부분의 SDK 라이브러리는 하나의 생성된 .cpp 파일로 빌드됩니다. 이렇게 하면 정적 라이브러리 크기가 상당히 줄어들고 컴파일 시간이 단축될 수 있습니다.

값

ON | OFF

## 기본값

OFF

## ENABLE\_VIRTUAL\_OPERATIONS

이 파라미터는 일반적으로 코드 생성을 위해 REGENERATE\_CLIENTS와 함께 작동합니다.

ENABLE\_VIRTUAL\_OPERATIONS가 ON이고 REGENERATE\_CLIENTS가 ON인 경우 서비스 클라이언트의 작업 관련 함수는 `virtual`로 표시됩니다.

ENABLE\_VIRTUAL\_OPERATIONS가 OFF이고 REGENERATE\_CLIENTS가 ON인 경우 `virtual`는 작업 함수에 추가되지 않으며 서비스 클라이언트 클래스는 `final`로 표시됩니다.

ENABLE\_VIRTUAL\_OPERATIONS가 OFF인 경우 SDK는 컴파일 시 gcc 및 clang에 대해 `-ffunction-sections` 및 `-fdata-sections` 컴파일러 플래그도 추가합니다.

자세한 내용은 GitHub의 [CMake 파라미터](#)를 참조하세요.

값

ON | OFF

기본값

ON

## ENABLE\_ZLIB\_REQUEST\_COMPRESSION

해당 기능을 지원하는 서비스의 경우 요청된 콘텐츠가 압축됩니다. 종속성이 사용 가능한 경우 기본적으로 활성화됩니다.

값

ON | OFF

기본값

ON

## FORCE\_CURL

Windows에만 해당됩니다. ON인 경우 기본 [WinHTTP](#) 데이터 전송 공급자 대신 curl 클라이언트를 강제로 사용합니다.

값

ON | OFF

기본값

OFF

## FORCE\_SHARED\_CRT

ON인 경우 SDK는 C 런타임에 동적으로 연결되며, 그렇지 않은 경우 BUILD\_SHARED\_LIBS 설정(이전 SDK 버전과의 이전 버전 호환성을 위해 필요한 경우가 있음)을 사용합니다.

값

ON | OFF

## 기본값

ON

## G

Visual Studio 솔루션 및 Xcode 프로젝트 등의 빌드 아티팩트를 생성합니다.

Windows 예:

```
-G "Visual Studio 12 Win64"
```

자세한 내용은 해당 플랫폼의 CMake 설명서를 참조하세요.

## MINIMIZE\_SIZE

[ENABLE\\_UNITY\\_BUILD](#)의 상위 집합입니다. ON인 경우 이 옵션은 ENABLE\_UNITY\_BUILD 및 추가 바이너리 크기 축소 설정을 활성화합니다.

## 값

ON | OFF

## 기본값

OFF

## NO\_ENCRYPTION

ON인 경우 기본 플랫폼별 암호화 구현이 라이브러리에 내장되지 않도록 합니다. 이 기능을 ON으로 설정하면 자체 암호화 구현을 주입할 수 있습니다.

## 값

ON | OFF

## 기본값

OFF

## NO\_HTTP\_CLIENT

ON인 경우 기본 플랫폼별 HTTP 클라이언트가 라이브러리에 내장되지 않도록 합니다. ON인 경우 자체 플랫폼별 HTTP 클라이언트 구현을 제공해야 합니다.

값

ON | OFF

기본값

OFF

## REGENERATE\_CLIENTS

ON인 경우 이 파라미터는 생성된 코드를 모두 삭제하고 `code-generation/api-definitions` 폴더에서 클라이언트 디렉터리를 생성합니다. 예:

```
-DREGENERATE_CLIENTS=1
```

### Note

REGENERATE\_CLIENTS 파라미터를 사용하려면 [Python 2.7](#), Java([JDK 1.8 이상](#)) 및 [Maven](#)이 설치되어 있고 PATH에 포함되어 있어야 합니다.

## REGENERATE\_DEFAULTS

ON인 경우 이 파라미터는 생성된 기본 코드를 모두 삭제하고 `code-generation/defaults` 폴더에서 다시 생성합니다. 예:

```
-DREGENERATE_DEFAULTS=1
```

### Note

REGENERATE\_DEFAULTS 파라미터를 사용하려면 [Python 2.7](#), Java([JDK 1.8 이상](#)) 및 [Maven](#)이 설치되어 있고 PATH에 포함되어 있어야 합니다.

## SIMPLE\_INSTALL

ON인 경우 설치 프로세스 중 bin/ 및 lib/ 아래에 플랫폼별 중간 디렉터리가 생성되지 않습니다. 단일 설치 디렉터리에서 다중 플랫폼 릴리스를 수행해야 하는 경우 OFF로 설정합니다.

값

ON | OFF

기본값

ON

## TARGET\_ARCH

모바일 플랫폼용으로 교차 컴파일하거나 빌드하려면 대상 플랫폼을 지정해야 합니다. 기본적으로 빌드는 호스트 운영 체제를 감지하여 감지된 운영 체제용으로 빌드합니다.

### Note

TARGET\_ARCH가 ANDROID인 경우 추가 옵션을 사용할 수 있습니다. [Android CMake 변수 및 옵션](#)을 참조하세요.

값

WINDOWS | LINUX | APPLE | ANDROID

## USE\_CRT\_HTTP\_CLIENT

ON인 경우 공통 런타임 HTTP 클라이언트가 사용되며, WinHttp 및 libcurl과 같은 레거시 시스템은 빌드되거나 포함되지 않습니다.

값

ON | OFF

기본값

OFF

## USE\_IXML\_HTTP\_REQUEST\_2

Windows에만 해당됩니다. ON인 경우 HTTP 스택에 com 객체 IXmlHttpRequest2를 사용합니다.

값

ON | OFF

기본값

OFF

## USE\_OPENSSL

ON인 경우 SDK는 OpenSSL을 사용하여 빌드하고, 그렇지 않은 경우 [awslabs/aws-lc](https://aws.amazon.com/blogs/aws/aws-lc/)를 사용합니다. AWS-LC는 AWS 암호화 팀이 AWS 및 고객을 위해 유지 관리하는 범용 암호화 라이브러리입니다. 이 파라미터를 OFF로 설정하면 시스템 기본 디렉터리에 OpenSSL 대신 AWS-LC가 설치됩니다. 시스템에 이미 OpenSSL이 설치되어 있는 경우에는 사용하지 마세요.

값

ON | OFF

기본값

ON

## USE\_TLS\_V1\_2

ON인 경우 HTTP 클라이언트에서 TLS 1.2를 적용합니다.

값

ON | OFF

기본값

ON

## USE\_TLS\_V1\_3

ON인 경우 HTTP 클라이언트에서 TLS 1.3을 적용합니다.

값

ON | OFF

기본값

OFF

## Android CMake 변수 및 옵션

SDK의 Android 빌드를 생성하는 경우([TARGET\\_ARCH](#)가 ANDROID로 설정된 경우) 다음 변수를 사용합니다.

주제

- [ANDROID\\_ABI](#)
- [ANDROID\\_BUILD\\_CURL](#)
- [ANDROID\\_BUILD\\_OPENSSL](#)
- [ANDROID\\_BUILD\\_ZLIB](#)
- [ANDROID\\_NATIVE\\_API\\_LEVEL](#)
- [ANDROID\\_STL](#)
- [ANDROID\\_TOOLCHAIN\\_NAME](#)
- [DISABLE\\_ANDROID\\_STANDALONE\\_BUILD](#)
- [NDK\\_DIR](#)

### ANDROID\_ABI

Android에만 해당됩니다. 코드를 출력할 Application Binary Interface(ABI)를 제어합니다.

#### Note

모든 유효한 Android ABI 값이 현재 지원되는 것은 아닙니다.

값

arm64 | armeabi-v7a | x86\_64 | x86 | mips64 | mips

## 기본값

armeabi-v7a

## ANDROID\_BUILD\_CURL

Android에만 해당됩니다. ON인 경우 curl도 빌드합니다.

값

ON | OFF

## 기본값

ON

## ANDROID\_BUILD\_OPENSSL

Android에만 해당됩니다. ON인 경우 Openssl도 빌드합니다.

값

ON | OFF

## 기본값

ON

## ANDROID\_BUILD\_ZLIB

Android에만 해당됩니다. ON인 경우 Zlib도 빌드합니다.

값

ON | OFF

## 기본값

ON

## ANDROID\_NATIVE\_API\_LEVEL

Android에만 해당됩니다. SDK가 빌드될 때 사용될 API 수준을 제어합니다. [ANDROID\\_STL](#)을 gnustdl로 설정한 경우 모든 API 수준을 선택할 수 있습니다. libc++를 사용하는 경우 최소 21 이상의 API 수준을 사용해야 합니다.

### 기본값

선택한 STL에 따라 다릅니다.

## ANDROID\_STL

Android에만 해당됩니다. SDK에서 사용하는 C++ 표준 라이브러리의 종류를 제어합니다.

### Important

gnustdl 옵션을 사용하는 경우 SDK 내에서 성능 문제가 발생할 수 있습니다. libc++\_shared 또는 libc++\_static을 사용하는 것이 좋습니다.

### 값

libc++\_shared | libc++\_static | gnustdl\_shared | gnustdl\_static

### 기본값

libc++\_shared

## ANDROID\_TOOLCHAIN\_NAME

Android에만 해당됩니다. SDK 빌드에 사용되는 컴파일러를 제어합니다.

### Note

Android NDK에서 GCC가 더 이상 권장되지 않으므로 기본값을 사용하는 것이 좋습니다.

### 기본값

standalone-clang

## DISABLE\_ANDROID\_STANDALONE\_BUILD

Android에만 해당됩니다. 기본적으로 Android 빌드는 NDK 스크립트를 통해 구성된 독립형 clang 기반 도구 체인을 사용합니다. 자체 도구 체인을 사용하려면 이 옵션을 ON으로 설정합니다.

값

ON | OFF

기본값

OFF

## NDK\_DIR

Android에만 해당됩니다. 빌드 시스템이 Android NDK를 찾을 때 사용할 재정의의 경로를 지정합니다. 기본적으로 빌드 시스템은 이 변수가 설정되지 않은 경우 환경 변수(ANDROID\_NDK)를 확인합니다.

## AWS SDK for C++에서 로깅 구성 및 사용

AWS SDK for C++에는 실행 중에 SDK가 수행한 작업의 레코드를 생성하는 구성 가능한 로깅이 포함되어 있습니다. 로깅을 활성화하려면 SDKOptions의 LogLevel을 애플리케이션에 적합한 상세 수준으로 설정합니다.

```
Aws::SDKOptions options;
options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Info;
```

선택할 수 있는 상세 수준은 7가지입니다. 기본값은 Off이며 이 경우 로그가 생성되지 않습니다. Trace는 가장 상세한 정보를 생성하며, Fatal은 치명적인 오류 상태만 보고하는 최소한의 메시지만 생성합니다.

애플리케이션에서 로깅이 활성화되면 SDK는 aws\_sdk\_<date>.log의 기본 이름 지정 패턴에 따라 실행 파일 디렉터리에 로그 파일을 생성합니다. 접두사 이름 지정 옵션으로 생성된 로그 파일은 로그 파일 보관 또는 삭제를 위해 매시간마다 롤오버됩니다.

SDK의 최신 버전은 기본 AWS 공통 런타임(CRT) 라이브러리에 점점 더 의존하고 있습니다. 이러한 라이브러리는 SDK 간 공통 기능과 기본 작업을 제공합니다. CRT 라이브러리에서 발생하는 모든 로그 메시지는 기본적으로 SDK for C++로 리디렉션됩니다. SDK for C++에 대해 지정하는 로그 수준 및 로깅 시스템이 CRT에도 적용됩니다.

위 예제에서 CRT는 `LogLevel::Info`를 상속하며 동일한 파일에 `Info` 수준으로 메시지를 로깅합니다.

출력을 별도의 로그 파일로 리디렉션하거나 CRT의 메시지에 대해 서로 다른 로그 수준을 설정하여 CRT 라이브러리에 대한 로깅을 개별적으로 제어할 수 있습니다. 로그를 과도하게 채우지 않도록 CRT 라이브러리의 상세 수준을 줄이는 것이 종종 유용합니다. 예를 들어, CRT 출력에 대해서만 로그 수준을 다음과 같이 `Warn`으로 설정할 수 있습니다.

```
options.loggingOptions.crt_logger_create_fn =
    [](){ return
    Aws::MakeShared<Aws::Utils::Logging::DefaultCRTLogSystem>("CRTLogSystem",
    Aws::Utils::Logging::LogLevel::Warn); };
```

선택적으로 `InitializeAWSLogging` 메서드를 사용하면 `DefaultLogSystem`의 상세 수준과 로그 출력을 제어할 수 있습니다. 로그 파일 이름 접두사를 구성하거나 출력을 파일 대신 스트림으로 리디렉션할 수 있습니다.

```
Aws::Utils::Logging::InitializeAWSLogging(
    Aws::MakeShared<Aws::Utils::Logging::DefaultLogSystem>(
        "RunUnitTests", Aws::Utils::Logging::LogLevel::Trace, "aws_sdk_"));
```

또는 `DefaultLogSystem`을 사용하는 대신 이 메서드를 사용하여 자체 로깅 구현을 제공할 수도 있습니다.

```
InitializeAWSLogging(Aws::MakeShared<CustomLoggingSystem>());
```

`InitializeAWSLogging` 메서드를 호출한 경우 `ShutdownAWSLogging`를 호출하여 프로그램 종료 시 리소스를 해제할 수 있습니다.

```
Aws::Utils::Logging::ShutdownAWSLogging();
```

## 로깅을 사용한 통합 테스트 예제

```
#include <aws/external/gtest.h>

#include <aws/core/utils/memory/stl/AWSString.h>
#include <aws/core/utils/logging/DefaultLogSystem.h>
#include <aws/core/utils/logging/AWSLogging.h>
```

```
#include <iostream>

int main(int argc, char** argv)
{
    Aws::Utils::Logging::InitializeAWSLogging(
        Aws::MakeShared<Aws::Utils::Logging::DefaultLogSystem>(
            "RunUnitTests", Aws::Utils::Logging::LogLevel::Trace, "aws_sdk_"));
    ::testing::InitGoogleTest(&argc, argv);
    int exitCode = RUN_ALL_TESTS();
    Aws::Utils::Logging::ShutdownAWSLogging();
    return exitCode;
}
```

사용자 지정 로깅을 위한 **Aws::Utils::Logging::DefaultLogSystem**의 하위 클래스 예제

다음 코드는 AWS SDK for C++의 일부인 **Aws::Utils::Logging::DefaultLogSystem** 클래스를 하위 클래스로 분류하는 방법을 보여줍니다. 이 예제에서는 **ProcessFormattedStatement** 가상 함수를 재정의하여 로깅을 사용자 지정합니다.

**Aws::Utils::Logging::DefaultLogSystem**은 AWS SDK for C++의 여러 클래스 중 하나로, 사용자 지정 로깅을 위해 **Aws::Utils::Logging::LogSystemInterface**을 하위 클래스로 분류합니다.

```
class LogSystemOverride : public Aws::Utils::Logging::DefaultLogSystem {
public:
    explicit LogSystemOverride(Aws::Utils::Logging::LogLevel logLevel,
                               const Aws::String &logPrefix)
        : DefaultLogSystem(logLevel, logPrefix), mLogToStreamBuf(false) {}

    const Aws::Utils::Stream::SimpleStreamBuf &GetStreamBuf() const {
        return mStreamBuf;
    }

    void setLogToStreamBuf(bool logToStreamBuf) {
        mLogToStreamBuf = logToStreamBuf;
    }

protected:

    void ProcessFormattedStatement(Aws::String &&statement) override {
        if (mLogToStreamBuf) {
            std::lock_guard<std::mutex> lock(mStreamMutex);
            mStreamBuf.sputn(statement.c_str(), statement.length());
        }
    }
};
```

```

    }

    DefaultLogSystem::ProcessFormattedStatement(std::move(statement));
}

private:
    Aws::Utils::Stream::SimpleStreamBuf mStreamBuf;
    // Use a mutex when writing to the buffer because
    // ProcessFormattedStatement can be called from multiple threads.
    std::mutex mStreamMutex;
    std::atomic<bool> mLogToStreamBuf;
};

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Trace;
    auto logSystemOverride = Aws::MakeShared<LogSystemOverride>("AllocationTag",

options.loggingOptions.logLevel,

options.loggingOptions.defaultLogPrefix);
    options.loggingOptions.logger_create_fn = [logSystemOverride]() {
        return logSystemOverride;
    };

    Aws::InitAPI(options); // Call Aws::InitAPI only once in an application.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::S3::S3Client s3Client(clientConfig);

        logSystemOverride->setLogToStreamBuf(true);
        auto outcome = s3Client.ListBuckets();
        if (!outcome.IsSuccess()) {
            std::cerr << "ListBuckets error: " <<
                outcome.GetError().GetExceptionName() << " " <<
                outcome.GetError().GetMessage() << std::endl;
        }

        logSystemOverride->setLogToStreamBuf(false);
    }
}

```

```

        std::cout << "Log for ListBuckets" << std::endl;
        std::cout << logSystemOverride->GetStreamBuf().str() << std::endl;
    }

    Aws::ShutdownAPI(options);

    return 0;
}

```

[전체 예제](#)는 GitHub에서 확인하세요.

## AWS SDK for C++에서 HTTP 클라이언트 재정의

Windows의 기본 HTTP 클라이언트는 [WinHTTP](#)입니다. 다른 모든 플랫폼의 기본 HTTP 클라이언트는 [curl](#)입니다.

선택적으로 서비스 클라이언트의 생성자에 전달할 사용자 지정 HttpClientFactory를 생성하여 HTTP 클라이언트 기본값을 재정의할 수 있습니다. HTTP 클라이언트를 재정의하려면 curl 지원을 추가하여 SDK를 빌드해야 합니다. Curl 지원은 Linux와 macOS에서 기본적으로 빌드되지만, Windows에서는 추가 단계가 필요합니다. curl 지원을 추가하여 Windows에서 SDK를 빌드하는 방법에 대한 자세한 내용은 [Windows에서 AWS SDK for C++ 빌드](#) 섹션을 참조하세요.

## AWS SDK for C++에서 HttpClient 및 AWSClient가 사용하는 입출력 스트림 제어

기본적으로 모든 응답은 stringbuf에서 지원하는 입력 스트림을 사용합니다. 필요한 경우 기본 동작을 재정의할 수 있습니다. 예를 들어 Amazon S3 GetObject를 사용 중인데 로드하고 싶지 않은 경우 AmazonWebServiceRequest의 IStreamFactory를 통해 lambda를 전달하여 파일 스트림을 생성할 수 있습니다.

### 파일 스트림 요청 예

```

//! Use a custom response stream when downloading an object from an Amazon Simple
//! Storage Service (Amazon S3) bucket.
/*!
\param bucketName: The Amazon S3 bucket name.
\param objectKey: The object key.
\param filePath: File path for custom response stream.
\param clientConfiguration: AWS client configuration.

```

```

    \return bool: Function succeeded.
    */

bool AwsDoc::SdkCustomization::customResponseStream(const Aws::String &bucketName,
                                                    const Aws::String &objectKey,
                                                    const Aws::String &filePath,
                                                    const
                                                    Aws::Client::ClientConfiguration &clientConfiguration) {

    Aws::S3::S3Client s3_client(clientConfiguration);

    Aws::S3::Model::GetObjectRequest getObjectRequest;
    getObjectRequest.WithBucket(bucketName).WithKey(objectKey);

    getObjectRequest.SetResponseStreamFactory([filePath]() {
        return Aws::New<Aws::FStream>(
            "FStreamAllocationTag", filePath, std::ios_base::out);
    });

    Aws::S3::Model::GetObjectOutcome getObjectOutcome = s3_client.GetObject(
        getObjectRequest);

    if (getObjectOutcome.IsSuccess()) {
        std::cout << "Successfully retrieved object to file " << filePath << std::endl;
    }
    else {
        std::cerr << "Error getting object. "
            << getObjectOutcome.GetError().GetMessage() << std::endl;
    }

    return getObjectOutcome.IsSuccess();
}

```

### Note

GitHub에 더 많은 내용이 있습니다. 전체 예제는 [AWS 코드 예제 리포지토리](#)에서 확인하세요.

## AWS SDK for C++에서 사용자 지정 libcrypto 라이브러리 사용

기본적으로는 AWS SDK for C++는 전송 계층 보안을 위해 기본 시스템 암호화 라이브러리를 사용합니다. 그러나 소스에서 SDK를 빌드할 때 다른 libcrypto 라이브러리를 사용하도록 선택적으로 SDK for C

++를 구성할 수 있습니다. 즉, 모든 암호화 작업이 OpenSSL의 사용자 지정 구현으로 전환됩니다. 예를 들어 애플리케이션에서 FIPS 표준을 달성하기 위해 [AWS-LC](#) 라이브러리를 [FIPS 모드](#)로 사용할 수 있습니다.

## SDK for C++에 사용자 지정 libcrypto를 빌드하는 방법

### 1단계: libcrypto 라이브러리 빌드 또는 가져오기

[AWS-LC](#)는 libcrypto 라이브러리를 대체할 수 있는 한 가지 예이지만, OpenSSL 또는 OpenSSL과 동등한 배포판이라면 무엇이든 사용할 수 있습니다.

SDK for C++와 그 종속성인 CRT는 모두 암호화 기능에 libcrypto를 사용하며, 둘 다 동일한 방식으로 종속성을 처리해야 합니다. SDK for C++는 요청이 SDK의 CRT S3 기능을 사용하는지 여부에 따라 두 가지 다른 HTTP 클라이언트에 의존합니다. 특히 CRT는 시작 시점에 초기화되는 TLS 구현인 [s2n](#)에 의존합니다. SDK와 s2n 팀 모두 [BUILD\\_SHARED\\_LIBS](#) 값에 관계없이 공유 libcrypto 라이브러리 사용을 강제하는 cmake 파라미터를 제공합니다. 일반적으로 CRT HTTP 클라이언트와 일반 HTTP 클라이언트가 동일한 libcrypto를 사용하도록 설정하는 것이 바람직합니다. 이 경우 양쪽 모두 종속성 트리에서 OpenSSL을 참조해야 함을 의미합니다. SDK는 [AWS\\_USE\\_CRYPTOSHARED\\_LIBS](#)를 통해, s2n(CRT 기반 호출용)은 [S2N\\_USE\\_CRYPTOSHARED\\_LIBS](#)를 통해 이를 제공합니다. 두 라이브러리 간의 의존성 해결 방식은 동일하며 일반적으로 일치하도록 설정되지만, 명시적으로 다르게 설정할 수도 있습니다.

예를 들어 AWS-LC를 libcrypto 라이브러리로 사용하려면 다음과 같이 빌드합니다.

```
git clone --depth 1 -b fips-2022-11-02 https://github.com/aws/aws-lc && \
  cd aws-lc && \
  mkdir build && \
  cd build && \
  cmake -G Ninja \
    -DCMAKE_INSTALL_LIBDIR=lib \
    -DCMAKE_INSTALL_PREFIX=/lc-install .. && \
  cmake --build . && \
  cmake --install . && \
  rm -rf .//* && \
  cmake -G Ninja \
    -DBUILD_SHARED_LIBS=ON \
    -DCMAKE_INSTALL_LIBDIR=lib \
    -DCMAKE_INSTALL_PREFIX=/lc-install .. && \
  cmake --build . && \
  cmake --install .
```

## 2단계: 소스에서 curl 빌드하거나 libcrypto 라이브러리가 포함된 curl 배포 사용

SDK for C++를 사용하려면 HTTP 요청을 수행할 시스템에 HTTP 클라이언트가 설치되어 있어야 합니다. HTTP 클라이언트는 사용하려는 libcrypto로 빌드해야 합니다. HTTP 클라이언트는 TLS 작업을 담당하므로 libcrypto 라이브러리가 사용됩니다.

다음 예에서는 설치된 AWS-LC 버전을 사용하여 curl 라이브러리를 다시 빌드합니다.

```
git clone --depth 1 -b curl-8_5_0 https://github.com/curl/curl && \
  cd curl && \
  autoreconf -fi && \
  mkdir build && \
  cd build && \
  ../configure \
    --enable-warnings \
    --enable-werror \
    --with-openssl=/lc-install \
    --prefix=/curl-install && \
  make && \
  make install
```

## 3단계: libcrypto 및 curl 라이브러리를 사용하여 SDK 빌드

이제 이전에 생성한 libcrypto 및 curl 아티팩트를 사용하여 SDK for C++를 빌드할 수 있습니다. 이 SDK 빌드는 모든 암호화 기능에 사용자 지정 libcrypto 라이브러리를 사용합니다.

```
git clone --depth 1 --recurse-submodules https://github.com/aws/aws-sdk-cpp \
  cd aws-sdk-cpp && \
  mkdir build && \
  cd build && \
  cmake -G Ninja \
    -DCMAKE_PREFIX_PATH="/curl-install;/lc-install;" \
    -DBUILD_ONLY="s3" \
    -DCMAKE_INSTALL_PREFIX=/sdk-install \
    -DAUTORUN_UNIT_TESTS=OFF .. && \
  cmake --build . && \
  cmake --install .
```

## Docker 이미지에 모두 통합

다음 샘플 Docker 파일은 Amazon Linux 2023 환경에서 이러한 단계를 구현하는 방법을 보여줍니다.

```
# User AL2023 Base image
FROM public.ecr.aws/amazonlinux/amazonlinux:2023

# Install Dev Tools
RUN yum groupinstall -y "Development Tools"
RUN yum install -y cmake3 ninja-build

# Build and install AWS-LC on the fips branch both statically and dynamically.
RUN git clone --depth 1 -b fips-2022-11-02 https://github.com/aws/aws-lc && \
    cd aws-lc && \
    mkdir build && \
    cd build && \
    cmake -G Ninja \
        -DCMAKE_INSTALL_LIBDIR=lib \
        -DCMAKE_INSTALL_PREFIX=/lc-install .. && \
    cmake --build . && \
    cmake --install . && \
    rm -rf .//* && \
    cmake -G Ninja \
        -DBUILD_SHARED_LIBS=ON \
        -DCMAKE_INSTALL_LIBDIR=lib \
        -DCMAKE_INSTALL_PREFIX=/lc-install .. && \
    cmake --build . && \
    cmake --install .

# Build and install curl targeting AWS-LC as openssl
RUN git clone --depth 1 -b curl-8_5_0 https://github.com/curl/curl && \
    cd curl && \
    autoreconf -fi && \
    mkdir build && \
    cd build && \
    ../configure \
        --enable-warnings \
        --enable-werror \
        --with-openssl=/lc-install \
        --prefix=/curl-install && \
    make && \
    make install

# Build and install SDK using the Curl and AWS-LC targets previously built
RUN git clone --depth 1 --recurse-submodules https://github.com/aws/aws-sdk-cpp \
    cd aws-sdk-cpp && \
```

```
mkdir build && \\  
cd build && \\  
cmake -G Ninja \\  
    -DCMAKE_PREFIX_PATH="/curl-install;/lc-install;" \\  
    -DBUILD_ONLY="s3" \\  
    -DCMAKE_INSTALL_PREFIX=/sdk-install \\  
    -DAUTORUN_UNIT_TESTS=OFF .. && \\  
cmake --build . && \\  
cmake --install .
```

# AWS SDK for C++ 사용

이 섹션에서는 [AWS SDK for C++ 사용 시작하기](#)에서 다루는 내용 외에도 AWS SDK for C++의 일반적인 사용에 대한 정보를 제공합니다.

서비스별 프로그래밍 예제는 [AWS SDK for C++ 코드 예제](#)를 참조하세요.

## 주제

- [AWS SDK for C++ 초기화 및 종료](#)
- [AWS SDK for C++를 사용하여 AWS 서비스 요청하기](#)
- [AWS SDK for C++를 사용한 비동기 프로그래밍](#)
- [AWS SDK for C++에서 사용 가능한 유틸리티 모듈](#)
- [AWS SDK for C++의 메모리 관리](#)
- [AWS SDK for C++에서 오류 처리](#)

## AWS SDK for C++ 초기화 및 종료

AWS SDK for C++를 사용하는 애플리케이션은 반드시 이를 초기화해야 합니다. 마찬가지로 애플리케이션이 종료되기 전에 SDK를 종료해야 합니다. 두 작업 모두 초기화 및 종료 프로세스와 이후 SDK 호출에 영향을 미치는 구성 옵션을 지원합니다.

AWS SDK for C++를 사용하는 모든 애플리케이션에는 `aws/core/Aws.h` 파일이 포함되어야 합니다.

AWS SDK for C++를 초기화하려면 `Aws::InitAPI`를 호출해야 합니다. 애플리케이션이 종료되기 전에 SDK를 종료하려면 `Aws::ShutdownAPI`를 호출해야 합니다. 각 메서드는 [Aws::SDKOptions](#)의 인수를 허용합니다. SDK에 대한 다른 모든 호출은 이 두 메서드 호출 사이에 수행할 수 있습니다.

**Aws::InitAPI**와 **Aws::ShutdownAPI** 사이에 수행되는 모든 AWS SDK for C++ 호출은 종괄호 쌍으로 묶거나, 두 메서드 사이에 호출되는 함수를 통해 간접적으로 호출해야 합니다.

기본 스켈레톤 애플리케이션은 다음과 같습니다.

```
#include <aws/core/Aws.h>
int main(int argc, char** argv)
{
    Aws::SDKOptions options;
    Aws::InitAPI(options);
    {
```

```

    // make your SDK calls here.
}
Aws::ShutdownAPI(options);
return 0;
}

```

SDK for C++ 및 해당 종속성은 C++ 정적 객체를 사용하며 정적 객체 폐기 순서는 C++ 표준에 따라 결정되지 않습니다. 정적 변수 폐기 순서의 비결정성으로 인해 발생하는 메모리 문제를 방지하려면 **Aws::InitAPI** 및 **Aws::ShutdownAPI** 호출을 다른 정적 객체로 래핑하지 마세요.

## AWS SDK for C++를 사용하여 AWS 서비스 요청하기

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 서비스를 요청하려면 먼저 서비스 클라이언트를 생성하고 [구성](#)해야 합니다. 코드에 사용된 각 AWS 서비스에는 해당 서비스와 상호 작용하기 위한 자체 라이브러리와 전용 유형이 있습니다. 클라이언트는 서비스가 제공하는 각 API 작업마다 하나의 메서드를 노출합니다.

클라이언트 클래스의 네임스페이스는 `Aws::Service::ServiceClient` 규칙을 따릅니다. 예를 들어 AWS Identity and Access Management(IAM)의 클라이언트 클래스는 `Aws::IAM::IAMClient`이고 Amazon S3의 클라이언트 클래스는 `Aws::S3::S3Client`입니다.

모든 AWS 서비스의 모든 클라이언트 클래스는 스레드 세이프입니다.

클라이언트 클래스를 인스턴스화할 때 AWS 자격 증명을 제공해야 합니다. 자격 증명은 코드, 환경 또는 공유 AWS config 파일 및 공유 credentials 파일에서 제공할 수 있습니다. 자격 증명에 대한 자세한 내용은 [권장 IAM Identity Center 인증을 설정하기 위한 지침](#)을 참조하거나 [사용 가능한 다른 자격 증명 공급자](#)를 사용하세요.

## AWS SDK for C++를 사용한 비동기 프로그래밍

### 비동기식 SDK 메서드

많은 메서드에 대해 SDK for C++는 동기식 버전과 비동기식 버전을 모두 제공합니다. 이름에 Async 접미사가 포함된 메서드는 비동기식입니다. 예를 들어 Amazon S3 메서드 `PutObject`는 동기식이고 `PutObjectAsync`는 비동기식입니다.

모든 비동기 작업과 마찬가지로, 비동기 SDK 메서드는 기본 태스크가 완료되기 전에 반환됩니다. 예를 들어 PutObjectAsync 메서드는 파일을 Amazon S3 버킷에 업로드하는 작업을 완료하기 전에 반환됩니다. 업로드 작업이 계속되는 동안 애플리케이션은 다른 비동기 메서드 호출을 포함하여 다른 작업을 수행할 수 있습니다. 연결된 콜백 함수가 호출될 때 비동기 작업이 완료되었다는 알림이 애플리케이션에 전송됩니다.

다음 섹션에서는 PutObjectAsync 비동기식 메서드 호출을 보여주는 코드 예제를 설명합니다. 각 섹션은 예제의 [전체 소스 파일](#)에서 개별 부분을 집중적으로 다룹니다.

## SDK 비동기식 메서드 호출

일반적으로 SDK 메서드의 비동기 버전은 다음 인수를 허용합니다.

- 동기식 버전과 동일한 요청 유형 객체에 대한 참조.
- 응답 핸들러 콜백 함수에 대한 참조. 이 콜백 함수는 비동기 작업이 완료되면 호출됩니다. 인수 중 하나에는 작업의 결과가 포함됩니다.
- AsyncCallerContext 객체에 대한 shared\_ptr(선택 사항). 이 객체는 응답 핸들러 콜백에 전달됩니다. 여기에는 텍스트 정보를 콜백에 전달하는 데 사용할 수 있는 UUID 속성이 포함됩니다.

아래에 표시된 uploadFileAsync 메서드는 Amazon S3 버킷에 파일을 비동기적으로 업로드하는 SDK의 Amazon S3 PutObjectAsync 메서드를 설정하고 호출합니다.

함수는 S3Client 객체 및 PutObjectRequest 객체에 대한 참조를 수신합니다. 함수가 이러한 참조를 메인 함수로부터 수신하는 이유는 비동기 호출이 지속되는 전체 기간 동안 이 객체들이 존재하도록 보장해야 하기 때문입니다.

AsyncCallerContext 객체에 대한 shared\_ptr이 할당됩니다. 해당 객체의 UUID 속성은 Amazon S3 객체 이름으로 설정됩니다. 시연을 위해 응답 핸들러 콜백은 속성에 액세스하여 해당 값을 출력합니다.

PutObjectAsync 호출에는 응답 핸들러 콜백 함수 uploadFileAsyncFinished에 대한 참조 인수가 포함됩니다. 이 콜백 함수는 다음 섹션에서 자세히 살펴봅니다.

```
bool AwsDoc::S3::uploadFileAsync(const Aws::S3::S3Client &s3Client,
                                  Aws::S3::Model::PutObjectRequest &request,
                                  const Aws::String &bucketName,
                                  const Aws::String &fileName) {
    request.SetBucket(bucketName);
```

```

request.SetKey(fileName);

const std::shared_ptr<Aws::IOStream> input_data =
    Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                   fileName.c_str(),
                                   std::ios_base::in | std::ios_base::binary);

if (!*input_data) {
    std::cerr << "Error: unable to open file " << fileName << std::endl;
    return false;
}

request.SetBody(input_data);

// Create and configure the context for the asynchronous put object request.
std::shared_ptr<Aws::Client::AsyncCallerContext> context =
    Aws::MakeShared<Aws::Client::AsyncCallerContext>("PutObjectAllocationTag");
context->SetUUID(fileName);

// Make the asynchronous put object call. Queue the request into a
// thread executor and call the uploadFileAsyncFinished function when the
// operation has finished.
s3Client.PutObjectAsync(request, uploadFileAsyncFinished, context);

return true;
}

```

비동기 작업의 리소스는 작업이 완료될 때까지 존재해야 합니다. 예를 들어, 클라이언트 및 요청 객체는 애플리케이션에서 작업 완료 알림을 수신할 때까지 존재해야 합니다. 애플리케이션 자체는 비동기 작업이 완료되기 전까지 종료될 수 없습니다.

이러한 이유로 uploadFileAsync 메서드는 uploadFileAsync 메서드 내에서 S3Client 및 PutObjectRequest 객체를 생성하고 로컬 변수에 저장하는 대신, 이들 객체에 대한 참조를 허용합니다.

이 예시에서 PutObjectAsync 메서드는 비동기 작업을 시작한 직후 호출자에게 반환되므로, 업로드 작업이 진행되는 동안 호출 체인이 추가 작업을 수행할 수 있습니다.

클라이언트가 uploadFileAsync 메서드의 로컬 변수에 저장된 경우 메서드가 반환될 때 범위를 벗어나게 됩니다. 그러나 클라이언트 객체는 비동기 작업이 완료될 때까지 계속 존재해야 합니다.

## 비동기 작업 완료 알림

비동기 작업이 완료되면 애플리케이션 응답 핸들러 콜백 함수가 호출됩니다. 이 알림에는 작업 결과가 포함됩니다. 결과는 메서드의 동기식 대응 메서드에서 반환하는 것과 동일한 결과 유형 클래스에 포함됩니다. 코드 예제에서 결과는 PutObjectOutcome 객체에 있습니다.

예제의 응답 핸들러 콜백 함수 uploadFileAsyncFinished는 다음과 같습니다. 비동기 작업이 성공 또는 실패했는지 확인합니다. std::condition\_variable을 사용하여 애플리케이션 스레드에 비동기 작업이 완료되었음을 알립니다.

```
// A mutex is a synchronization primitive that can be used to protect shared
// data from being simultaneously accessed by multiple threads.
std::mutex AwsDoc::S3::upload_mutex;
```

```
// A condition_variable is a synchronization primitive that can be used to
// block a thread, or to block multiple threads at the same time.
// The thread is blocked until another thread both modifies a shared
// variable (the condition) and notifies the condition_variable.
std::condition_variable AwsDoc::S3::upload_variable;
```

```
void uploadFileAsyncFinished(const Aws::S3::S3Client *s3Client,
                             const Aws::S3::Model::PutObjectRequest &request,
                             const Aws::S3::Model::PutObjectOutcome &outcome,
                             const std::shared_ptr<const
Aws::Client::AsyncCallerContext> &context) {
    if (outcome.IsSuccess()) {
        std::cout << "Success: uploadFileAsyncFinished: Finished uploading '"
                    << context->GetUUID() << "'." << std::endl;
    } else {
        std::cerr << "Error: uploadFileAsyncFinished: " <<
                    outcome.GetError().GetMessage() << std::endl;
    }

    // Unblock the thread that is waiting for this function to complete.
    AwsDoc::S3::upload_variable.notify_one();
}
```

비동기 작업이 완료되면 이와 관련된 리소스를 해제할 수 있습니다. 원하는 경우 애플리케이션을 종료할 수도 있습니다.

다음 코드는 애플리케이션에서 `uploadFileAsync` 및 `uploadFileAsyncFinished` 메서드를 사용하는 방법을 보여줍니다.

애플리케이션에서는 비동기 작업이 완료될 때까지 `S3Client` 및 `PutObjectRequest` 객체가 계속 존재하도록 해당 객체를 할당합니다. `uploadFileAsync` 호출 후 애플리케이션은 원하는 작업을 수행할 수 있습니다. 단순화를 위해 이 예제에서는 `std::mutex` 및 `std::condition_variable`을 사용하여 응답 핸들러 콜백으로 업로드 작업 완료 알림이 전달될 때까지 대기합니다.

```
int main(int argc, char* argv[])
{
    if (argc != 3)
    {
        std::cout << R"(
Usage:
    run_put_object_async <file_name> <bucket_name>
Where:
    file_name - The name of the file to upload.
    bucket_name - The name of the bucket to upload the object to.
)" << std::endl;
        return 1;
    }

    const Aws::SDKOptions options;
    Aws::InitAPI(options);
    {
        const Aws::String fileName = argv[1];
        const Aws::String bucketName = argv[2];

        // A unique_lock is a general-purpose mutex ownership wrapper allowing
        // deferred locking, time-constrained attempts at locking, recursive
        // locking, transfer of lock ownership, and use with
        // condition variables.
        std::unique_lock<std::mutex> lock(AwsDoc::S3::upload_mutex);

        // Create and configure the Amazon S3 client.
        // This client must be declared here, as this client must exist
        // until the put object operation finishes.
        const Aws::S3::S3ClientConfiguration config;
        // Optional: Set to the AWS Region in which the bucket was created (overrides
        config file).
        // config.region = "us-east-1";

        const Aws::S3::S3Client s3Client(config);
```

```

// Create the request object.
// This request object must be declared here, because the object must exist
// until the put object operation finishes.
Aws::S3::Model::PutObjectRequest request;

AwsDoc::S3::uploadFileAsync(s3Client, request, bucketName, fileName);

std::cout << "main: Waiting for file upload attempt..." <<
    std::endl << std::endl;

// While the put object operation attempt is in progress,
// you can perform other tasks.
// This example simply blocks until the put object operation
// attempt finishes.
AwsDoc::S3::upload_variable.wait(lock);

std::cout << std::endl << "main: File upload attempt completed."
    << std::endl;
}
Aws::ShutdownAPI(options);

return 0;
}

```

[전체 예제](#)는 Github에서 확인하세요.

## AWS SDK for C++에서 사용 가능한 유틸리티 모듈

AWS SDK for C++에는 C++로 AWS 애플리케이션을 개발하는 복잡성을 줄여주는 다양한 [유틸리티 모듈](#)이 포함되어 있습니다.

### HTTP 스택

HTTP 스택은 연결 풀링을 제공하고, 스레드 세이프를 제공하며, 필요에 따라 재사용할 수 있습니다. 자세한 내용은 [AWS 클라이언트 구성](#)을 참조하세요.

|         |                                 |
|---------|---------------------------------|
| 헤더      | <a href="#">/aws/core/http/</a> |
| API 설명서 | <a href="#">Aws::Http</a>       |

## 문자열 유틸리티

핵심 문자열 함수입니다(예: trim, lowercase, 숫자 변환).

|         |                                              |
|---------|----------------------------------------------|
| 헤더      | <a href="#">aws/core/utils/StringUtils.h</a> |
| API 설명서 | <a href="#">Aws::Utils::StringUtils</a>      |

## 해싱 유틸리티

해싱 함수입니다(SHA256, MD5, Base64, SHA256\_HMAC).

|         |                                                |
|---------|------------------------------------------------|
| 헤더      | <a href="#">/aws/core/utils/HashingUtils.h</a> |
| API 설명서 | <a href="#">Aws::Utils::HashingUtils</a>       |

## JSON 파서

완벽하게 작동하면서도 경량인 JSON 파서입니다(*cJSON*의 얇은 래퍼).

|         |                                                       |
|---------|-------------------------------------------------------|
| 헤더      | <a href="#">/aws/core/utils/json/JsonSerializer.h</a> |
| API 설명서 | <a href="#">Aws::Utils::Json::JsonValue</a>           |

## XML 파서

경량 XML 파서입니다(*tinycl2*의 얇은 래퍼). [RAII 패턴](#)이 인터페이스에 추가되었습니다.

|         |                                                     |
|---------|-----------------------------------------------------|
| 헤더      | <a href="#">/aws/core/utils/xml/XmlSerializer.h</a> |
| API 설명서 | <a href="#">Aws::Utils::Xml</a>                     |

# AWS SDK for C++의 메모리 관리

AWS SDK for C++는 라이브러리에서 메모리 할당 및 할당 해제를 제어하는 방법을 제공합니다.

## Note

사용자 지정 메모리 관리는 정의된 컴파일 시간 상수 `USE_AWS_MEMORY_MANAGEMENT`를 사용하여 빌드된 라이브러리 버전을 사용하는 경우에만 사용할 수 있습니다. 컴파일 시간 상수 없이 빌드된 라이브러리 버전을 사용하는 경우에는 `InitializeAWSMemorySystem`과 같은 글로벌 메모리 시스템 함수가 작동하지 않으며, 대신 글로벌 `new` 및 `delete` 함수가 사용됩니다.

컴파일 시간 상수에 대한 자세한 내용은 [STL 및 AWS 문자열과 벡터](#)를 참조하세요.

## 메모리 할당 및 할당 해제

메모리를 할당하거나 할당 해제하려면

1. 하위 클래스 `MemorySystemInterface`: `aws/core/utils/memory/MemorySystemInterface.h`.

```
class MyMemoryManager : public Aws::Utils::Memory::MemorySystemInterface
{
public:
    // ...
    virtual void* AllocateMemory(
        std::size_t blockSize, std::size_t alignment,
        const char *allocationTag = nullptr) override;
    virtual void FreeMemory(void* memoryPtr) override;
};
```

## Note

필요에 따라 `AllocateMemory`에 대한 유형 서명을 변경할 수 있습니다.

2. `Aws::SDKOptions` 구조체를 사용하여 사용자 지정 메모리 관리자 사용을 구성합니다. 구조체의 인스턴스를 `Aws::InitAPI`로 전달합니다. 애플리케이션을 종료하기 전에 동일한 인스턴스로 `Aws::ShutdownAPI`를 호출하여 SDK를 종료해야 합니다.

```
int main(void)
{
    MyMemoryManager sdkMemoryManager;
    SDKOptions options;
    options.memoryManagementOptions.memoryManager = &sdkMemoryManager;
    Aws::InitAPI(options);

    // ... do stuff

    Aws::ShutdownAPI(options);

    return 0;
}
```

## STL 및 AWS 문자열과 벡터

메모리 관리자로 초기화되면 AWS SDK for C++는 모든 할당 및 할당 해제를 메모리 관리자에게 위임합니다. 메모리 관리자가 없는 경우 SDK는 글로벌 `new` 및 `delete`를 사용합니다.

사용자 지정 STL 할당기를 사용하는 경우 할당 정책과 일치하도록 모든 STL 객체의 유형 서명을 변경해야 합니다. SDK 구현 및 인터페이스에서는 STL이 주로 사용되므로, SDK 내에서 단일 접근 방식을 채택할 경우 기본 STL 객체를 SDK로 직접 전달하거나 STL 할당을 제어하는 것이 제한될 수 있습니다. 대안으로, 내부적으로는 사용자 지정 할당기를 사용하고 인터페이스에서는 표준 및 사용자 지정 STL 객체를 허용하는 하이브리드 접근 방식은 메모리 문제 조사에 더 어려움을 초래할 수 있습니다.

해결 방법은 메모리 시스템의 컴파일 시간 상수 `USE_AWS_MEMORY_MANAGEMENT`를 사용하여 SDK에서 사용하는 STL 유형을 제어하는 것입니다.

컴파일 시간 상수가 활성화(켜짐)된 경우, 해당 유형은 사용자 정의 할당기가 AWS 메모리 시스템에 연결된 STL 유형으로 확인됩니다.

컴파일 시간 상수가 비활성화(꺼짐)된 경우 모든 `Aws::*` 유형은 해당하는 기본 `std::*` 유형으로 확인됩니다.

SDK의 **AWSAllocator.h** 파일에서 가져온 코드 예제

```
#ifdef USE_AWS_MEMORY_MANAGEMENT

template< typename T >
class AwsAllocator : public std::allocator< T >
```

```
{
    ... definition of allocator that uses AWS memory system
};

#else

template< typename T > using Allocator = std::allocator<T>;

#endif
```

예제 코드에서 `AwsAllocator`는 컴파일 시간 상수에 따라 사용자 지정 할당기 또는 기본 할당기일 수 있습니다.

SDK의 **AWSVector.h** 파일에서 가져온 코드 예제

```
template<typename T> using Vector = std::vector<T, Aws::Allocator<T>>;
```

예제 코드에서는 `Aws::*` 유형을 정의합니다.

컴파일 시간 상수가 활성화(켜짐)된 경우 유형은 사용자 지정 메모리 할당 및 AWS 메모리 시스템을 사용하여 벡터에 매핑됩니다.

컴파일 시간 상수가 비활성화(꺼짐)된 경우 유형은 기본 유형 파라미터를 사용하는 일반 `std::vector`에 매핑됩니다.

유형 별칭 지정은 컨테이너, 문자열 스트림, 문자열 버퍼 등 메모리 할당을 수행하는 SDK의 모든 `std::` 유형에 사용됩니다. AWS SDK for C++에서는 이러한 유형을 사용합니다.

## 남아 있는 문제

메모리 할당은 SDK에서 제어할 수 있지만, STL 유형은 여전히 모델 객체 `initialize` 및 `set` 메서드에 대한 문자열 파라미터를 통해 퍼블릭 인터페이스의 주요 비중을 차지합니다. STL을 사용하지 않고 문자열과 컨테이너를 대신 사용한다면 서비스 호출을 수행할 때마다 많은 임시 객체를 생성해야 합니다.

STL이 아닌 방식으로 서비스 호출을 수행할 때 발생하는 임시 객체와 할당을 대부분 제거하기 위해 다음과 같이 구현했습니다.

- 문자열을 사용하는 모든 `Init/Set` 함수에는 `const char*`를 사용하는 오버로드가 있습니다.
- 컨테이너(맵/벡터)를 사용하는 모든 `Init/Set` 함수에는 단일 항목을 사용하는 `add` 변형이 있습니다.

- 바이너리 데이터를 사용하는 모든 Init/Set 함수에는 데이터 포인터와 length 값을 사용하는 오버로드가 있습니다.
- (선택 사항) 문자열을 사용하는 모든 Init/Set 함수에는 0 종단이 아닌 `const char*` 및 length 값을 사용하는 오버로드가 있습니다.

## 네이티브 SDK 개발자 및 메모리 제어

SDK 코드에서는 다음 규칙을 따릅니다.

- `new` 및 `delete`를 대신해서 `Aws::New<>` 및 `Aws::Delete<>`를 사용합니다.
- `new[]` 및 `delete[]`를 대신해서 `Aws::NewArray<>` 및 `Aws::DeleteArray<>`를 사용합니다.
- `std::make_shared`을 대신해서 `Aws::MakeShared`를 사용합니다.
- 단일 객체에 대한 고유 포인터에 `Aws::UniquePtr`을 사용합니다. 고유 포인터를 생성하려면 `Aws::MakeUnique` 함수를 사용합니다.
- 객체 배열에 대한 고유 포인터에 `Aws::UniqueArray`를 사용합니다. 고유 포인터를 생성하려면 `Aws::MakeUniqueArray` 함수를 사용합니다.
- STL 컨테이너를 직접 사용하지 말고, `Aws::typedef` 중 하나를 사용하거나 원하는 컨테이너에 대한 `typedef`를 추가합니다. 예:

```
Aws::Map<Aws::String, Aws::String> m_kvPairs;
```

- SDK로 전달되어 관리되는 모든 외부 포인터에 `shared_ptr`을 사용합니다. 객체 할당 방식과 일치하는 폐기 정책으로 공유 포인터를 초기화해야 합니다. SDK가 포인터를 정리하지 않을 것으로 예상되는 경우 원시 포인터를 사용할 수 있습니다.

## AWS SDK for C++에서 오류 처리

AWS SDK for C++는 예외를 사용하지 않지만 코드에 예외를 사용할 수 있습니다. 모든 서비스 클라이언트는 결과와 오류 코드가 포함된 결과 객체를 반환합니다.

오류 조건 처리 예제

```
bool CreateTableAndWaitForItToBeActive()
{
    CreateTableRequest createTableRequest;
    AttributeDefinition hashKey;
    hashKey.SetAttributeName(HASH_KEY_NAME);
```

```

    hashKey.SetAttributeType(ScalarAttributeType::S);
    createTableRequest.AddAttributeDefinitions(hashKey);
    KeySchemaElement hashKeySchemaElement;
    hashKeySchemaElement.WithAttributeName(HASH_KEY_NAME).WithKeyType(KeyType::HASH);
    createTableRequest.AddKeySchema(hashKeySchemaElement);
    ProvisionedThroughput provisionedThroughput;
    provisionedThroughput.SetReadCapacityUnits(readCap);
    provisionedThroughput.SetWriteCapacityUnits(writeCap);
    createTableRequest.WithProvisionedThroughput(provisionedThroughput);
    createTableRequest.WithTableName(tableName);

    CreateTableOutcome createTableOutcome = dynamoDbClient-
>CreateTable(createTableRequest);
    if (createTableOutcome.IsSuccess())
    {
        DescribeTableRequest describeTableRequest;
        describeTableRequest.SetTableName(tableName);
        bool shouldContinue = true;
        DescribeTableOutcome outcome = dynamoDbClient-
>DescribeTable(describeTableRequest);

        while (shouldContinue)
        {
            if (outcome.GetResult().GetTable().GetTableStatus() == TableStatus::ACTIVE)
            {
                break;
            }
            else
            {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
        }
        return true;
    }
    else if(createTableOutcome.GetError().GetErrorType() ==
DynamoDBErrors::RESOURCE_IN_USE)
    {
        return true;
    }

    return false;
}

```

# AWS SDK for C++에서 AWS 서비스 호출

다음 섹션에는 AWS SDK for C++를 사용하여 AWS 서비스를 사용하는 방법을 보여주는 예제, 자습서, 작업 및 가이드가 포함되어 있습니다.

AWS SDK for C++를 처음 사용하는 경우 [시작하기](#) 주제를 먼저 읽어보는 것이 좋습니다.

더 많은 코드 예제는 GitHub의 [C++ 예제 폴더](#)에서 찾을 수 있습니다.

이 안내서의 [코드 예제](#) 장 또는 GitHub의 [AWS 코드 예제 리포지토리](#)에서도 더 많은 코드 예제를 찾을 수 있습니다.

## 주제

- [AWS SDK for C++ 코드 예제 시작하기](#)
- [AWS SDK for C++에서 런타임 오류 해결 시작하기](#)
- [AWS SDK for C++를 사용하여 AWS 서비스를 호출하는 안내 예제](#)

## AWS SDK for C++ 코드 예제 시작하기

### 코드 예제의 구조

GitHub의 [C++ 예제 폴더](#)는 각 AWS 서비스별 프로젝트 폴더로 구성됩니다. 일반적으로 폴더 내 개별 .cpp 소스 파일은 해당 서비스의 특정 기능이나 동작을 보여줍니다. 예를 들어 Amazon DynamoDB의 경우 데이터베이스에서 항목을 가져오는 작업과 데이터베이스에 항목을 업로드하는 작업은 서로 다른 유형의 동작이므로 DynamoDB 폴더에는 각각 `get_item.cpp` 및 `put_item.cpp`라는 별도의 파일이 있습니다. 각 .cpp 파일에는 독립형 실행 파일의 진입점 역할을 하는 `main()` 함수가 포함되어 있습니다. 프로젝트 실행 파일은 빌드 시스템에서 지정한 폴더에 생성되며 각 예제 소스 파일에 해당하는 실행 파일이 하나씩 있습니다. 실행 파일의 이름은 `{name}.exe` 또는 `{name}`와 같이 플랫폼의 규칙을 따르며, `CMakeLists.txt`에 지정된 `run_`과 같은 사용자 지정 접두사가 적용됩니다.

### 예제 기능을 실행하려면

1. GitHub의 [AWS 코드 예제 리포지토리](#)에서 원하는 코드 예제를 다운로드합니다.
2. .cpp 파일을 열어 `main()` 함수와 호출된 메서드를 살펴봅니다.
3. [AWS SDK for C++를 사용하여 시작하기](#)의 스타터 예제와 같이 진행합니다. 프로젝트를 빌드하면 프로젝트 내 모든 소스 파일별로 실행 파일이 각각 생성됩니다.

#### 4. 선택한 기능의 실행 파일을 실행합니다.

- 명령 프롬프트에서 \*.cpp 파일 이름에 기반한 실행 파일을 사용하여 해당 프로그램을 실행합니다.
- IDE 내에서 작업하는 경우 시연하려는 기능의 .cpp 파일을 선택한 후 스타트업 옵션(또는 스타트업 객체)으로 선택합니다.

## 유닛 테스트

예제에 대한 테스트는 GoogleTest 프레임워크를 사용하여 작성됩니다. 자세한 내용은 GoogleTest 웹 사이트의 [GoogleTest Primer](#)를 참조하세요.

각 예제에 대한 유닛 테스트는 자체 CMakeLists.txt 파일이 포함된 tests 하위 폴더에 있습니다. 각 예제 소스 파일마다 gtest\_<source file>이라는 이름의 대응하는 테스트 파일이 있습니다. 하위 폴더에 대한 테스트 실행 파일의 이름은 <AWS ###>\_gtests입니다.

## CMakeLists.txt 파일

각 서비스의 폴더에는 CMakeLists.txt라는 파일이 포함되어 있습니다. 이러한 파일 중 다수에는 다음과 유사한 구문이 포함되어 있습니다.

```
foreach(EXAMPLE IN LISTS EXAMPLES)
    add_executable(${EXAMPLE} ${EXAMPLE}.cpp)
    target_link_libraries(${EXAMPLE} aws-cpp-sdk-email aws-cpp-sdk-core)
endforeach()
```

폴더 내 각 .cpp 파일에 대해 CMakeLists.txt 파일은 파일 확장자를 제외한 이름을 기반으로 실행 파일(cmake: add\_executable)을 빌드합니다.

## Visual Studio에서 코드 예제 빌드 및 디버깅

### Amazon S3 코드 예제 빌드 및 실행

1. Amazon S3 예제 소스 코드를 가져옵니다. 이 절차에서는 Visual Studio를 사용하여 작업을 시작하기 위해 [AWS SDK for C++를 사용한 Amazon S3용 코드 예제](#) 코드 예제를 사용합니다.
2. Windows 탐색기에서 s3 폴더(예: \aws-doc-sdk-examples\cpp\example\_code\s3)로 이동합니다.
3. s3 예제 폴더를 마우스 오른쪽 버튼으로 클릭하고 Visual Studio로 열기를 선택합니다. CMake 프로젝트용 Visual Studio에는 '프로젝트' 파일이 없고, 전체 폴더가 프로젝트 파일 역할을 합니다.

4. Visual Studio의 상단 메뉴에 있는 구성 선택기 드롭다운에서 선택한 구성이 소스에서 SDK를 빌드할 때 선택한 빌드 유형과 일치하는지 확인합니다. 예를 들어, SDK 설치 지침에 따라 CMake 명령줄에서 디버그 옵션(-DCMAKE\_BUILD\_TYPE=Debug)을 사용하여 소스에서 빌드한 경우 디버그 구성을 선택해야 합니다.
5. CMakeLists.txt 파일을 엽니다.
6. 저장을 클릭합니다. CMakeLists.txt 파일에서 저장을 클릭할 때마다 Visual Studio에서 CMake 생성 파일을 새로 고칩니다. 출력 탭이 표시되어 있다면 이 생성 과정에서 발생한 로그 메시지를 확인할 수 있습니다.
  - 출력 탭에는 "다음에서 출력 표시:"라는 드롭다운 상자가 있는데 CMake가 기본적으로 선택되어 있어야 합니다.
  - 마지막 메시지 출력에는 "CMake 생성 완료"라고 표시되어 있어야 합니다.
  - 마지막 메시지가 이와 다르다면 CMake 파일에 문제가 있는 것입니다. 이 문제가 해결될 때까지 추가 단계를 진행하지 마세요. [AWS SDK for C++ 빌드 문제 해결](#)(를) 참조하세요.
  - CMake에서는 속도 향상을 위해 CMake 캐시를 사용합니다. CMake 관련 문제를 해결하는 경우 표시되는 오류 메시지가 최근 변경 사항을 실제로 반영하도록 "클린 슬레이트" 상태를 유지해야 합니다. 솔루션 탐색기에서 CMakeLists.txt를 마우스 오른쪽 버튼으로 클릭하고 CMake 캐시를 선택한 다음 캐시 삭제를 선택합니다. CMake 문제를 단계적으로 해결할 때 이 작업을 자주 수행합니다.
7. Visual Studio 내에서 예제를 빌드하고 실행하기 위해 Visual Studio는 실행 파일을 명령줄과 다른 폴더 구조에 배치합니다. 코드를 실행하려면 SDK 실행 파일을 올바른 위치에 복사해야 합니다. CMakeLists 파일의 "TODO" 줄(~40번째 줄)을 찾아 Visual Studio에서 사용하기 위해 주석 처리된 줄을 선택합니다. Visual Studio는 빌드 유형 전용 하위 폴더를 사용하지 않으므로 이 줄은 포함되지 않습니다. Visual Studio 사용을 위해 CMakeLists.txt 파일에서 주석 처리된 줄을 전환합니다.
8. 위에서 설명한 대로 CMake 캐시를 삭제하고 CMakeLists.txt 파일을 클릭하여 탭을 선택/활성화한 다음 CMakeLists.txt 파일에서 저장을 다시 선택하여 CMake 빌드 파일 생성을 시작합니다.
9. 실행하려는 '프로그램'의 소스 파일을 엽니다.
  - 예를 들어 list\_buckets.cpp를 엽니다.
  - Amazon S3 예제 폴더는 Amazon S3의 각 '기능'이 해당 기능만을 위한 전용 실행 파일에서 시연되도록 코딩되어 있습니다. 예를 들어 list\_buckets.cpp는 버킷 목록 표시 기능만을 시연하는 실행 파일이 됩니다.
10. 상단 메뉴에서 빌드를 선택한 다음 모두 빌드를 선택합니다.

- 출력 탭의 다음에서 출력 표시에는 빌드 선택 사항이 반영되며 모든 빌드 및 연결 메시지가 표시됩니다.
  - 마지막 출력은 "모두 빌드 성공"이어야 합니다.
  - 이제 개별 소스 파일 각각에 대한 실행 파일이 생성됩니다. 빌드 출력 디렉터리(예: `\aws-doc-sdk-examples\cpp\example_code\s3\out\build\x64-Debug`)를 확인하면 이를 확인할 수 있습니다.
  - CMakeLists.txt 파일에 명시된 대로 실행 파일에는 "run\_" 접두사가 붙습니다.
11. 상단 메뉴에는 디버그 대상에 대한 녹색 화살표와 드롭다운 선택기가 있습니다. `run_list_buckets.exe`를 선택합니다.
  12. 녹색 화살표 실행 버튼을 클릭하여 시작 항목을 선택합니다.
  13. Visual Studio 디버그 콘솔 창이 열리고 코드 출력이 표시됩니다.
  14. 키를 눌러 창을 닫거나 수동으로 창을 닫아 프로그램을 종료합니다. 코드에 중단점을 설정할 수도 있으며, 실행을 다시 클릭하면 중단점에 도달합니다.

## AWS SDK for C++에서 런타임 오류 해결 시작하기

AWS SDK for C++로 애플리케이션을 개발하는 방법을 익히는 동시에 AWS Management 콘솔 및 AWS CLI 사용에도 익숙해지는 것이 중요합니다. 이러한 도구는 런타임 오류 발생 시 다양한 문제 해결 및 진단 작업을 위해 상호 교환적으로 사용할 수 있습니다.

다음 자습서에서는 이러한 문제 해결 및 진단 태스크의 예를 보여줍니다. 여러 가지 이유로 발생할 수 있는 Access denied 오류에 초점을 맞추고 있습니다. 이 자습서에서는 오류의 실제 원인을 파악하는 방법을 예시로 보여줍니다. 특히 두 가지 가능한 원인, 즉 현재 사용자에 대한 잘못된 권한 설정과 현재 사용자가 사용할 수 없는 리소스에 초점을 두고 설명합니다.

프로젝트 소스 및 실행 파일을 가져오려면

1. GitHub의 [AWS 코드 예제 리포지토리](#)에서 Amazon S3 코드 예제 폴더를 다운로드합니다.
2. `delete_bucket.cpp`를 열고 두 가지 메서드인 `main()` 및 `DeleteBucket()`이 있음을 확인합니다. `DeleteBucket()`은 SDK를 사용하여 버킷을 삭제합니다.
3. [AWS SDK for C++를 사용하여 시작하기](#)에 설명된 것과 동일한 빌드 단계를 사용하여 Amazon S3 예제를 빌드합니다. 빌드 프로세스는 각 소스 파일에 대한 실행 파일을 생성합니다.
4. 빌드 시스템이 빌드 실행 파일을 생성한 폴더로 이동하여 명령 프롬프트를 엽니다. 실행 파일 `run_create_bucket`을 실행합니다. 실제 실행 파일 이름은 운영 체제에 따라 다릅니다. 그러면 계정에 버킷이 생성됩니다(삭제할 버킷이 생김).

5. 명령 프롬프트에서 실행 파일 `run_delete_bucket`을 실행합니다. 이 예제에서는 삭제하려는 버킷의 이름을 파라미터로 예상합니다. 잘못된 버킷 이름을 제공합니다. 문제 해결을 탐색할 수 있도록 지금은 의도적으로 이 버킷 이름에 오타를 만듭니다.
6. Access Denied 오류 메시지가 표시되는지 확인합니다. Access Denied 오류 메시지가 표시되면 Amazon S3에 대한 전체 권한을 가진 사용자를 생성했는지 의문이 들게 되는데, 다음 단계에서 이를 확인할 것입니다.

AWS CLI를 설치하고 AWS를 호출하는 사용자 이름을 찾으려면

1. 개발 시스템에 최신 AWS CLI를 설치하려면 AWS Command Line Interface 사용 설명서에서 [AWS CLI 설치](#)를 참조하세요.
2. AWS CLI가 작동하는지 확인하려면 명령 프롬프트를 열고 `aws --version` 명령을 실행합니다.

```
$ aws --version
aws-cli/2.1.29 Python/3.8.8 Windows/10 exe/AMD64 prompt/off
```

3. 실제로 AWS를 호출하는 사용자 이름을 확인하려면 AWS CLI 명령 `aws sts get-caller-identity`를 실행합니다. 다음 예제 출력에서 해당 사용자 이름은 `userX`입니다.

```
$ aws sts get-caller-identity
{
  "UserId": "A12BCD34E5FGHI6JKLM",
  "Account": "1234567890987",
  "Arn": "arn:aws:iam::1234567890987:user/userX"
}
```

자격 증명을 지정하는 방법은 다양하지만 [AWS SDK for C++를 AWS 사용하여 로 인증](#)의 방식에 따라 진행했다면 이 사용자 이름은 AWS 공유 자격 증명 파일에서 가져온 것입니다. 해당 절차 중에 사용자에게 AmazonS3FullAccess 권한을 부여했습니다.

#### Note

일반적으로 대부분의 AWS CLI 명령은 다음과 같은 구문 구조를 따릅니다.

```
$ aws <command> <subcommand> [options and parameters]
```

여기서 *command*는 서비스이고 *subcommand*는 해당 서비스에서 호출되는 메서드입니다. 자세한 내용은 AWS Command Line Interface 사용 설명서에서 [AWS CLI의 명령 구조](#)를 참조하세요.

사용자가 버킷 삭제 권한이 있는지 확인하려면

1. [AWS Management 콘솔](#)을 열고 로그인합니다. 자세한 내용은 [AWS Management 콘솔 시작하기](#)를 참조하세요.
2. 기본 탐색 모음의 서비스 검색...에 **IAM**를 입력하고 결과에서 IAM 서비스를 선택합니다.
3. 대시보드 사이드바 또는 IAM 리소스에서 사용자를 선택합니다.
4. 계정에 사용할 수 있는 사용자 테이블에서 이전 절차에서 확인한 사용자 이름을 선택합니다.
5. 요약 페이지의 권한 탭을 선택하고 정책 이름 테이블에서 AmazonS3FullAccess를 선택합니다.
6. 정책 요약과 JSON 데이터를 살펴봅니다. 이 사용자에게 Amazon S3 서비스에 대한 전체 권한이 있는지 확인합니다.

```
"Effect": "Allow",
"Action": "s3:*",
"Resource": "*"

```

이 제거 프로세스는 문제의 원인을 배제하는 데 흔히 사용됩니다. 이 경우 사용자에게 올바른 권한이 있음을 확인했으므로 문제의 원인은 다른 데 있을 것입니다. 즉, 버킷에 액세스할 수 있는 올바른 권한이 있는데도 Access Denied 오류가 발생한다면 본인이 소유하지 않은 버킷에 액세스하려 했기 때문일 수 있습니다. 문제 해결 시, 프로그램에 제공된 버킷 이름을 검토하면 해당 이름의 버킷이 계정에 존재하지 않으므로 해당 버킷에 '액세스'할 수 없다는 점을 확인하게 됩니다.

코드 예제가 성공적으로 실행되도록 업데이트하려면

1. `delete_bucket.cpp`의 `main()` 함수로 돌아가 `enum`을 사용하여 리전을 계정의 리전으로 변경합니다. 계정의 리전을 찾으려면 AWS Management 콘솔에 로그인한 후 오른쪽 상단 모서리에서 리전을 찾습니다. 또한 `main()`에서 버킷 이름을 계정에 있는 버킷으로 변경합니다. 현재 버킷 이름을 확인하는 방법은 다음과 같습니다.
  - 이 코드 예제의 폴더에도 있는 `run_list_buckets` 실행 파일을 사용하여 버킷 이름을 프로그래밍 방식으로 가져올 수 있습니다.
  - 다음 AWS CLI 명령을 사용하여 Amazon S3 버킷을 나열할 수 있습니다.

```
$ aws s3
ls
2022-01-05 14:27:48 amzn-s3-demo-bucket
```

- [AWS Management 콘솔](#)도 사용할 수 있습니다. 기본 탐색 모음의 서비스 검색...에 **S3**를 입력합니다. 버킷 페이지에 계정의 버킷이 나열됩니다.
2. 코드를 다시 빌드하고 업데이트된 실행 파일 `run_delete_bucket`을 실행합니다.
  3. AWS Management 콘솔 또는 AWS CLI를 사용하여 이전에 생성한 Amazon S3 버킷이 삭제되었는지 확인합니다.

## AWS SDK for C++를 사용하여 AWS 서비스를 호출하는 안내 예제

AWS 또는 AWS 코드 예제를 처음 접하는 경우 [코드 예제 시작하기](#)부터 시작하는 것이 좋습니다.

AWS SDK for C++를 사용하여 AWS 서비스를 사용하는 방법을 보여주는 소스 코드는 이 안내서의 [코드 예제](#) 장에서 확인하거나 GitHub의 [AWS 코드 예제 리포지토리](#)에서 직접 확인할 수 있습니다.

이 섹션에서는 여러 AWS 서비스를 선택하고 이를 활용하는 예제를 안내합니다. 다음 안내 예제는 Github에서 제공되는 예제의 일부입니다.

추가 설명이 포함된 서비스 예제(전체 목록은 [AWS 코드 예제 리포지토리](#) 참조)

| 서비스                                                      | 서비스가 프로그램에 제공하는 기능 요약                            |
|----------------------------------------------------------|--------------------------------------------------|
| <a href="#">Amazon CloudWatch()</a>                      | 사용 중인 AWS 리소스에 대한 지표를 수집하고 모니터링합니다.              |
| <a href="#">- Amazon DynamoDB</a>                        | NoSQL 데이터베이스 서비스                                 |
| <a href="#">Amazon Elastic Compute Cloud(Amazon EC2)</a> | 안전하고 크기 조정 가능한 컴퓨팅 용량                            |
| <a href="#">Amazon Simple Storage Service(Amazon S3)</a> | 데이터 저장 및 검색(객체를 버킷에)                             |
| <a href="#">Amazon Simple Queue Service(Amazon SQS)</a>  | 소프트웨어 구성 요소 간에 메시지를 전송, 저장 및 수신하기 위한 메시지 대기열 서비스 |

[비동기식 메서드](#) 사용법을 보여주는 예시제도 있습니다.

AWS 문서 팀에 새로운 코드 예제를 제안하려면 GitHub의 [Contributing guidelines](#)를 참조하여 새 요청을 생성하세요. 팀에서 선호하는 방식은 개별 API 직접 호출보다는 광범위한 시나리오를 보여주는 코드 예제를 생성하는 것입니다.

Windows에서 코드 예제 사용

SDK 버전 1.9를 사용하여 Windows에서 예제를 빌드하는 경우 [AWS SDK for C++ 빌드 문제 해결](#) 섹션을 참조하세요.

## AWS SDK for C++를 사용한Amazon CloudWatch 예제

Amazon CloudWatch(CloudWatch)는 AWS 클라우드 리소스 및 AWS에서 실행하는 애플리케이션을 모니터링하는 서비스입니다. 다음 예제를 사용하면 AWS SDK for C++를 사용하여 [CloudWatch](#)를 프로그래밍할 수 있습니다.

Amazon CloudWatch는 AWS에서 실행하는 AWS 리소스와 애플리케이션을 실시간으로 모니터링합니다. CloudWatch를 사용하여 리소스 및 애플리케이션에 대해 측정할 수 있는 변수인 지표를 수집하고 추적할 수 있습니다. CloudWatch 경보는 알림을 보내거나 정의한 규칙을 기준으로 모니터링하는 리소스를 자동으로 변경합니다.

CloudWatch에 대한 자세한 내용은 [Amazon CloudWatch 사용 설명서](#)를 참조하세요.

### Note

이 안내서에는 특정 기술을 시연하는 데 필요한 코드만 제공되며 [전체 예제 코드는 GitHub에서 확인할 수 있습니다](#). GitHub에서 단일 소스 파일을 다운로드하거나, 리포지토리를 로컬에 복제하여 모든 예제를 가져오고 빌드하고 실행할 수 있습니다.

### 주제

- [Amazon CloudWatch에서 지표 가져오기](#)
- [사용자 지정 지표 데이터 게시](#)
- [Amazon CloudWatch 경보 작업](#)
- [CloudWatch에서 경보 조치 사용](#)
- [CloudWatch에 이벤트 전송](#)

## Amazon CloudWatch에서 지표 가져오기

### 사전 조건

시작하기 전에 [AWS SDK for C++ 사용 시작하기](#)를 읽어보시기 바랍니다.

예제 코드를 다운로드하고 [코드 예제 시작하기](#)에 설명된 대로 솔루션을 빌드합니다.

예제를 실행하려면 코드가 요청을 수행하는 데 사용하는 사용자 프로필이 AWS에서 적절한 권한(서비스 및 동작에 대한)을 보유하고 있어야 합니다. 자세한 내용은 [AWS 자격 증명 제공](#)을 참조하세요.

### 지표 나열

CloudWatch 지표를 나열하려면 [ListMetricsRequest](#)를 생성하고 CloudWatchClient의 ListMetrics 함수를 호출합니다. ListMetricsRequest를 사용하여 반환된 지표를 네임스페이스, 지표 이름 또는 차원을 기준으로 필터링할 수 있습니다.

#### Note

AWS 서비스에서 게시한 지표 및 측정기준 목록은 Amazon CloudWatch 사용 설명서의 [Amazon CloudWatch 지표 및 차원 참조](#)에서 확인할 수 있습니다.

### 포함 파일

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/ListMetricsRequest.h>
#include <aws/monitoring/model/ListMetricsResult.h>
#include <iomanip>
#include <iostream>
```

### 코드

```
Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::ListMetricsRequest request;

if (argc > 1)
{
    request.SetMetricName(argv[1]);
}
```

```
if (argc > 2)
{
    request.SetNamespace(argv[2]);
}

bool done = false;
bool header = false;
while (!done)
{
    auto outcome = cw.ListMetrics(request);
    if (!outcome.IsSuccess())
    {
        std::cout << "Failed to list CloudWatch metrics:" <<
            outcome.GetError().GetMessage() << std::endl;
        break;
    }

    if (!header)
    {
        std::cout << std::left << std::setw(48) << "MetricName" <<
            std::setw(32) << "Namespace" << "DimensionNameValuePairs" <<
            std::endl;
        header = true;
    }

    const auto &metrics = outcome.GetResult().GetMetrics();
    for (const auto &metric : metrics)
    {
        std::cout << std::left << std::setw(48) <<
            metric.GetMetricName() << std::setw(32) <<
            metric.GetNamespace();
        const auto &dimensions = metric.GetDimensions();
        for (auto iter = dimensions.cbegin();
            iter != dimensions.cend(); ++iter)
        {
            const auto &dimkv = *iter;
            std::cout << dimkv.GetName() << " = " << dimkv.GetValue();
            if (iter + 1 != dimensions.cend())
            {
                std::cout << ", ";
            }
        }
        std::cout << std::endl;
    }
}
```

```

    const auto &next_token = outcome.GetResult().GetNextToken();
    request.SetNextToken(next_token);
    done = next_token.empty();
}

```

해당 `GetMetrics` 함수를 호출하면 지표가 [ListMetricsResult](#)에 반환됩니다. 결과를 페이징할 수 있습니다. 다음 결과 배치를 가져오려면 `ListMetricsResult` 객체의 `GetNextToken` 함수 반환 값과 함께 원래 요청 객체에 대해 `SetNextToken`을 호출한 후 수정된 요청 객체를 `ListMetrics`의 다른 호출에 다시 전달합니다.

[전체 예제](#)를 참조하세요.

## 추가 정보

- Amazon CloudWatch API 참조의 [ListMetrics](#).

## 사용자 지정 지표 데이터 게시

여러 AWS 서비스는 AWS/으로 시작하는 네임스페이스에 [자체 지표](#)를 게시합니다. 자체 네임스페이스 (AWS/로 시작하지 않음)를 사용하여 사용자 지정 지표 데이터를 게시할 수도 있습니다.

## 사전 조건

시작하기 전에 [AWS SDK for C++ 사용 시작하기](#)를 읽어보시기 바랍니다.

예제 코드를 다운로드하고 [코드 예제 시작하기](#)에 설명된 대로 솔루션을 빌드합니다.

예제를 실행하려면 코드가 요청을 수행하는 데 사용하는 사용자 프로필이 AWS에서 적절한 권한(서비스 및 동작에 대한)을 보유하고 있어야 합니다. 자세한 내용은 [AWS 자격 증명 제공](#)을 참조하세요.

## 사용자 지정 지표 데이터 게시

자체 지표 데이터를 게시하려면 `CloudWatchClient`의 `PutMetricData` 함수를 [PutMetricDataRequest](#)와 함께 호출합니다. `PutMetricDataRequest`는 데이터에 사용할 사용자 지정 네임스페이스와, [MetricDatum](#) 객체의 데이터 포인트 자체에 대한 정보를 포함해야 합니다.

### Note

AWS/로 시작하는 네임스페이스는 지정할 수 없습니다. AWS/로 시작하는 네임스페이스는 Amazon Web Services 제품용으로 예약되어 있습니다.

## 포함 파일

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/PutMetricDataRequest.h>
#include <iostream>
```

## 코드

```
Aws::CloudWatch::CloudWatchClient cw;

Aws::CloudWatch::Model::Dimension dimension;
dimension.SetName("UNIQUE_PAGES");
dimension.SetValue("URLS");

Aws::CloudWatch::Model::MetricDatum datum;
datum.SetMetricName("PAGES_VISITED");
datum.SetUnit(Aws::CloudWatch::Model::StandardUnit::None);
datum.SetValue(data_point);
datum.AddDimensions(dimension);

Aws::CloudWatch::Model::PutMetricDataRequest request;
request.SetNamespace("SITE/TRAFFIC");
request.AddMetricData(datum);

auto outcome = cw.PutMetricData(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to put sample metric data:" <<
        outcome.GetError().GetMessage() << std::endl;
}
else
{
    std::cout << "Successfully put sample metric data" << std::endl;
}
```

[전체 예제](#)를 참조하세요.

## 추가 정보

- Amazon CloudWatch 사용 설명서의 [Amazon CloudWatch 지표 사용](#)
- Amazon CloudWatch 사용 설명서의 [AWS 네임스페이스](#)

- Amazon CloudWatch API 참조의 [PutMetricData](#).

## Amazon CloudWatch 경보 작업

### 사전 조건

시작하기 전에 [AWS SDK for C++ 사용 시작하기](#)를 읽어보시기 바랍니다.

예제 코드를 다운로드하고 [코드 예제 시작하기](#)에 설명된 대로 솔루션을 빌드합니다.

예제를 실행하려면 코드가 요청을 수행하는 데 사용하는 사용자 프로필이 AWS에서 적절한 권한(서비스 및 동작에 대한)을 보유하고 있어야 합니다. 자세한 내용은 [AWS 자격 증명 제공](#)을 참조하세요.

### 경보 만들기

CloudWatch 지표를 기반으로 경보를 생성하려면 CloudWatchClient의 PutMetricAlarm 함수를 경보 조건이 입력된 [PutMetricAlarmRequest](#)와 함께 호출합니다.

### 포함 파일

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/PutMetricAlarmRequest.h>
#include <iostream>
```

### 코드

```
Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::PutMetricAlarmRequest request;
request.SetAlarmName(alarm_name);
request.SetComparisonOperator(
    Aws::CloudWatch::Model::ComparisonOperator::GreaterThanThreshold);
request.SetEvaluationPeriods(1);
request.SetMetricName("CPUUtilization");
request.SetNamespace("AWS/EC2");
request.SetPeriod(60);
request.SetStatistic(Aws::CloudWatch::Model::Statistic::Average);
request.SetThreshold(70.0);
request.SetActionsEnabled(false);
request.SetAlarmDescription("Alarm when server CPU exceeds 70%");
request.SetUnit(Aws::CloudWatch::Model::StandardUnit::Seconds);

Aws::CloudWatch::Model::Dimension dimension;
```

```

dimension.SetName("InstanceId");
dimension.SetValue(instanceId);

request.AddDimensions(dimension);

auto outcome = cw.PutMetricAlarm(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to create CloudWatch alarm:" <<
        outcome.GetError().GetMessage() << std::endl;
}
else
{
    std::cout << "Successfully created CloudWatch alarm " << alarm_name
        << std::endl;
}

```

[전체 예제](#)를 참조하세요.

## 경보 나열

생성한 CloudWatch 경보를 나열하려면 CloudWatchClient의 DescribeAlarms 함수를 결과에 대한 옵션을 설정하는 데 사용할 수 있는 [DescribeAlarmsRequest](#)와 함께 호출합니다.

## 포함 파일

```

#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DescribeAlarmsRequest.h>
#include <aws/monitoring/model/DescribeAlarmsResult.h>
#include <iomanip>
#include <iostream>

```

## 코드

```

Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::DescribeAlarmsRequest request;
request.SetMaxRecords(1);

bool done = false;
bool header = false;
while (!done)
{

```

```

    auto outcome = cw.DescribeAlarms(request);
    if (!outcome.IsSuccess())
    {
        std::cout << "Failed to describe CloudWatch alarms:" <<
            outcome.GetError().GetMessage() << std::endl;
        break;
    }

    if (!header)
    {
        std::cout << std::left <<
            std::setw(32) << "Name" <<
            std::setw(64) << "Arn" <<
            std::setw(64) << "Description" <<
            std::setw(20) << "LastUpdated" <<
            std::endl;
        header = true;
    }

    const auto &alarms = outcome.GetResult().GetMetricAlarms();
    for (const auto &alarm : alarms)
    {
        std::cout << std::left <<
            std::setw(32) << alarm.GetAlarmName() <<
            std::setw(64) << alarm.GetAlarmArn() <<
            std::setw(64) << alarm.GetAlarmDescription() <<
            std::setw(20) <<
            alarm.GetAlarmConfigurationUpdatedTimestamp().ToGmtString(
                SIMPLE_DATE_FORMAT_STR) <<
            std::endl;
    }

    const auto &next_token = outcome.GetResult().GetNextToken();
    request.SetNextToken(next_token);
    done = next_token.empty();
}

```

DescribeAlarms에 의해 반환되는 [DescribeAlarmsResult](#)에 대해 getMetricAlarms를 호출하여 경보 목록을 가져올 수 있습니다.

결과를 페이징할 수 있습니다. 다음 결과 배치를 가져오려면 DescribeAlarmsResult 객체의 GetNextToken 함수 반환 값과 함께 원래 요청 객체에 대해 SetNextToken을 호출한 후 수정된 요청 객체를 DescribeAlarms의 다른 호출에 다시 전달합니다.

**Note**

CloudWatchClient의 DescribeAlarmsForMetric 함수를 사용하여 특정 지표의 경보를 검색할 수도 있습니다. 이 메서드의 용도는 DescribeAlarms와 비슷합니다.

[전체 예제](#)를 참조하세요.

## 경보 삭제

CloudWatch 경보를 삭제하려면 삭제하려는 경보 이름이 하나 이상 포함된 [DeleteAlarmsRequest](#)와 함께 CloudWatchClient의 DeleteAlarms 함수를 호출합니다.

## 포함 파일

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DeleteAlarmsRequest.h>
#include <iostream>
```

## 코드

```
Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::DeleteAlarmsRequest request;
request.AddAlarmNames(alarm_name);

auto outcome = cw.DeleteAlarms(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to delete CloudWatch alarm:" <<
        outcome.GetError().GetMessage() << std::endl;
}
else
{
    std::cout << "Successfully deleted CloudWatch alarm " << alarm_name
        << std::endl;
}
```

[전체 예제](#)를 참조하세요.

## 추가 정보

- Amazon CloudWatch 사용 설명서의 [Amazon CloudWatch 경보 생성](#)
- Amazon CloudWatch API 참조의 [PutMetricAlarm](#)
- Amazon CloudWatch API 참조의 [DescribeAlarms](#)
- Amazon CloudWatch API 참조의 [DeleteAlarms](#)

## CloudWatch에서 경보 조치 사용

CloudWatch 경보 작업을 사용하면 Amazon EC2 인스턴스 자동 중지, 종료, 재부팅 또는 복구 등의 작업을 수행하는 경보를 생성할 수 있습니다.

경보 작업은 [경보를 생성](#)할 때 [PutMetricAlarmRequest](#)의 SetAlarmActions 함수를 사용하여 경보에 추가할 수 있습니다.

## 사전 조건

시작하기 전에 [AWS SDK for C++ 사용 시작하기](#)를 읽어보시기 바랍니다.

예제 코드를 다운로드하고 [코드 예제 시작하기](#)에 설명된 대로 솔루션을 빌드합니다.

예제를 실행하려면 코드가 요청을 수행하는 데 사용하는 사용자 프로필이 AWS에서 적절한 권한(서비스 및 동작에 대한)을 보유하고 있어야 합니다. 자세한 내용은 [AWS 자격 증명 제공](#)을 참조하세요.

## 경보 작업 활성화

CloudWatch 경보에 대한 경보 작업을 활성화하려면 활성화하려는 작업을 수행할 경보의 이름이 하나 이상 포함된 [EnableAlarmActionsRequest](#)와 함께 CloudWatchClient의 EnableAlarmActions를 호출합니다.

## 포함 파일

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/EnableAlarmActionsRequest.h>
#include <aws/monitoring/model/PutMetricAlarmRequest.h>
#include <iostream>
```

## 코드

```
Aws::CloudWatch::CloudWatchClient cw;
```

```

Aws::CloudWatch::Model::PutMetricAlarmRequest request;
request.SetAlarmName(alarm_name);
request.SetComparisonOperator(
    Aws::CloudWatch::Model::ComparisonOperator::GreaterThanThreshold);
request.SetEvaluationPeriods(1);
request.SetMetricName("CPUUtilization");
request.SetNamespace("AWS/EC2");
request.SetPeriod(60);
request.SetStatistic(Aws::CloudWatch::Model::Statistic::Average);
request.SetThreshold(70.0);
request.SetActionsEnabled(false);
request.SetAlarmDescription("Alarm when server CPU exceeds 70%");
request.SetUnit(Aws::CloudWatch::Model::StandardUnit::Seconds);
request.AddAlarmActions(actionArn);

Aws::CloudWatch::Model::Dimension dimension;
dimension.SetName("InstanceId");
dimension.SetValue(instanceId);
request.AddDimensions(dimension);

auto outcome = cw.PutMetricAlarm(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to create CloudWatch alarm:" <<
        outcome.GetError().GetMessage() << std::endl;
    return;
}

Aws::CloudWatch::Model::EnableAlarmActionsRequest enable_request;
enable_request.AddAlarmNames(alarm_name);

auto enable_outcome = cw.EnableAlarmActions(enable_request);
if (!enable_outcome.IsSuccess())
{
    std::cout << "Failed to enable alarm actions:" <<
        enable_outcome.GetError().GetMessage() << std::endl;
    return;
}

std::cout << "Successfully created alarm " << alarm_name <<
    " and enabled actions on it." << std::endl;

```

[전체 예제](#)를 참조하세요.

## 경보 작업 비활성화

CloudWatch 경보에 대한 경보 작업을 비활성화하려면 비활성화하려는 작업을 수행할 경보의 이름이 하나 이상 포함된 [DisableAlarmActionsRequest](#)와 함께 CloudWatchClient의 `DisableAlarmActions`를 호출합니다.

### 포함 파일

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DisableAlarmActionsRequest.h>
#include <iostream>
```

### 코드

```
Aws::CloudWatch::CloudWatchClient cw;

Aws::CloudWatch::Model::DisableAlarmActionsRequest disableAlarmActionsRequest;
disableAlarmActionsRequest.AddAlarmNames(alarm_name);

auto disableAlarmActionsOutcome =
cw.DisableAlarmActions(disableAlarmActionsRequest);
if (!disableAlarmActionsOutcome.IsSuccess())
{
    std::cout << "Failed to disable actions for alarm " << alarm_name <<
        ": " << disableAlarmActionsOutcome.GetError().GetMessage() <<
        std::endl;
}
else
{
    std::cout << "Successfully disabled actions for alarm " <<
        alarm_name << std::endl;
}
```

[전체 예제](#)를 참조하세요.

### 추가 정보

- Amazon CloudWatch 사용 설명서의 [인스턴스를 중지, 종료, 재부팅 또는 복구하는 경보 생성](#)
- Amazon CloudWatch API 참조의 [PutMetricAlarm](#)
- Amazon CloudWatch API 참조의 [EnableAlarmActions](#)

- Amazon CloudWatch API 참조의 [DisableAlarmActions](#)

## CloudWatch에 이벤트 전송

CloudWatch Events는 AWS 리소스의 변경 사항을 설명하는 시스템 이벤트의 실시간에 가까운 시스템 이벤트 스트림을 Amazon EC2 인스턴스, Lambda 함수, Kinesis 스트림, Amazon ECS 태스크, Step Functions 상태 머신, Amazon SNS 주제, Amazon SQS 대기열 또는 기본 제공 대상으로 전송합니다. 단순 규칙을 사용하여 일치하는 이벤트를 검색하고 하나 이상의 대상 함수 또는 스트림으로 이를 라우팅할 수 있습니다.

### Note

이 코드 스니펫은 사용자가 [AWS SDK for C++ 사용 시작하기](#)의 내용을 이해하고 있으며, [AWS 자격 증명 개발](#)의 정보를 사용하여 기본 AWS 자격 증명을 구성했다고 가정합니다.

## 이벤트 추가

사용자 지정 CloudWatch 이벤트를 추가하려면 CloudWatchEventsClient의 PutEvents 함수를 호출합니다. 이때, 각 이벤트에 대한 세부 정보를 제공하는 [PutEventsRequestEntry](#) 객체를 하나 이상 포함하는 [PutEventsRequest](#) 객체를 함께 사용합니다. 이벤트 유형 및 소스, 이벤트와 연결된 리소스 등 입력 항목에 대한 여러 파라미터를 지정할 수 있습니다.

### Note

putEvents 호출당 최대 10개 이벤트를 지정할 수 있습니다.

## 포함 파일

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutEventsRequest.h>
#include <aws/events/model/PutEventsResult.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

## 코드

```

Aws::CloudWatchEvents::EventBridgeClient cwe;

Aws::CloudWatchEvents::Model::PutEventsRequestEntry event_entry;
event_entry.SetDetail(MakeDetails(event_key, event_value));
event_entry.SetDetailType("sampleSubmitted");
event_entry.AddResources(resource_arn);
event_entry.SetSource("aws-sdk-cpp-cloudwatch-example");

Aws::CloudWatchEvents::Model::PutEventsRequest request;
request.AddEntries(event_entry);

auto outcome = cwe.PutEvents(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to post CloudWatch event: " <<
        outcome.GetError().GetMessage() << std::endl;
}
else
{
    std::cout << "Successfully posted CloudWatch event" << std::endl;
}

```

## 규칙 추가

규칙을 생성하거나 업데이트하려면 CloudWatchEventsClient의 PutRule 함수를 호출합니다. 이때 규칙 이름과 선택적 파라미터([이벤트 패턴](#), 규칙과 연결할 IAM 역할, 규칙의 실행 빈도를 설명하는 [예약 표현식](#))를 [PutRuleRequest](#)와 함께 사용합니다.

## 포함 파일

```

#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutRuleRequest.h>
#include <aws/events/model/PutRuleResult.h>
#include <aws/core/Utils/Outcome.h>
#include <iostream>

```

## 코드

```

Aws::CloudWatchEvents::EventBridgeClient cwe;
Aws::CloudWatchEvents::Model::PutRuleRequest request;

```

```

request.SetName(rule_name);
request.SetRoleArn(role_arn);
request.SetScheduleExpression("rate(5 minutes)");
request.SetState(Aws::CloudWatchEvents::Model::RuleState::ENABLED);

auto outcome = cwe.PutRule(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to create CloudWatch events rule " <<
        rule_name << ": " << outcome.GetError().GetMessage() <<
        std::endl;
}
else
{
    std::cout << "Successfully created CloudWatch events rule " <<
        rule_name << " with resulting Arn " <<
        outcome.GetResult().GetRuleArn() << std::endl;
}

```

## 대상 추가

대상은 규칙이 트리거될 때 호출되는 리소스입니다. 대상의 예시로는 Amazon EC2 인스턴스, Lambda 함수, Kinesis 스트림, Amazon ECS 태스크, Step Functions 상태 머신, 기본 제공 대상 등이 있습니다.

규칙에 대상을 추가하려면 CloudWatchEventsClient의 PutTargets 함수를 호출합니다. 이때 업데이트할 규칙과 규칙에 추가할 대상 목록을 포함하여 [PutTargetsRequest](#)를 함께 사용합니다.

## 포함 파일

```

#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutTargetsRequest.h>
#include <aws/events/model/PutTargetsResult.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>

```

## 코드

```

Aws::CloudWatchEvents::EventBridgeClient cwe;

Aws::CloudWatchEvents::Model::Target target;
target.SetArn(lambda_arn);

```

```

target.SetId(target_id);

Aws::CloudWatchEvents::Model::PutTargetsRequest request;
request.SetRule(rule_name);
request.AddTargets(target);

auto putTargetsOutcome = cwe.PutTargets(request);
if (!putTargetsOutcome.IsSuccess())
{
    std::cout << "Failed to create CloudWatch events target for rule "
                << rule_name << ": " <<
                putTargetsOutcome.GetError().GetMessage() << std::endl;
}
else
{
    std::cout <<
        "Successfully created CloudWatch events target for rule "
        << rule_name << std::endl;
}

```

[전체 예제](#)를 참조하세요.

## 추가 정보

- Amazon CloudWatch Events 사용 설명서의 [PutEvents를 통한 이벤트 추가](#)
- Amazon CloudWatch Events 사용 설명서의 [규칙에 대한 예약 표현식](#)
- Amazon CloudWatch Events 사용 설명서의 [CloudWatch Events의 이벤트 유형](#)
- Amazon CloudWatch Events 사용 설명서의 [이벤트 및 이벤트 패턴](#)
- Amazon CloudWatch Events API 참조의 [PutEvents](#)
- Amazon CloudWatch Events API 참조의 [PutTargets](#)
- Amazon CloudWatch Events API 참조의 [PutRule](#)

## AWS SDK for C++를 사용한 Amazon DynamoDB 예제

Amazon DynamoDB는 완전관리형 NoSQL 데이터베이스 서비스로서 원활한 확장성과 함께 빠르고 예측 가능한 성능을 제공합니다. 다음 예제에서는 AWS SDK for C++를 사용하여 [Amazon DynamoDB](#)를 프로그래밍하는 방법을 보여줍니다.

**Note**

이 안내서에는 특정 기술을 시연하는 데 필요한 코드만 제공되며 [전체 예제 코드는 GitHub에서 확인할 수 있습니다](#). GitHub에서 단일 소스 파일을 다운로드하거나, 리포지토리를 로컬에 복제하여 모든 예제를 가져오고 빌드하고 실행할 수 있습니다.

## 주제

- [DynamoDB에서 테이블 작업](#)
- [DynamoDB에서 항목 작업](#)

## DynamoDB에서 테이블 작업

테이블은 DynamoDB 데이터베이스 내 모든 항목을 위한 컨테이너입니다. DynamoDB에 데이터를 추가하거나 제거하기 전에 테이블을 생성해야 합니다.

각 테이블마다 다음을 정의해야 합니다.

- AWS 계정 및 AWS 리전에 고유한 테이블 이름
- 프라이머리 키(모든 값이 고유해야 함). 테이블 내 두 항목은 동일한 프라이머리 키 값을 사용할 수 없습니다.

기본 키는 단일 파티션(HASH) 키로 이루어진 단순형이거나, 파티션과 정렬(RANGE) 키로 이루어진 복합형일 수 있습니다.

각 키 값에는 [ScalarAttributeType](#) 클래스에 의해 열거되는 관련 데이터 유형이 있습니다. 키 값은 이진(B), 숫자(N) 또는 문자열(S)일 수 있습니다. 자세한 내용은 Amazon DynamoDB 개발자 안내서에서 [이름 지정 규칙 및 데이터 형식](#)을 참조하세요.

- 테이블에 대해 예약된 읽기/쓰기 용량 단위를 정의하는 프로비저닝된 처리량

**Note**

[Amazon DynamoDB 요금](#)은 테이블에 대해 설정하는 프로비저닝된 처리량 값을 기준으로 하므로 테이블에 대해 필요한 만큼의 용량만 예약하십시오. 언제라도 테이블의 프로비저닝된 처리량을 수정할 수 있으므로 변경이 필요할 경우 용량을 조정할 수 있습니다.

## 테이블 생성

새 DynamoDB 테이블을 생성하려면 [DynamoDB 클라이언트](#)의 CreateTable 메서드를 사용합니다. 테이블 속성과 테이블 스키마를 구성해야 하며, 이 두 가지 요소 모두 테이블의 기본 키를 식별하는 데 사용됩니다. 초기 프로비저닝 처리량 값과 테이블 이름을 제공해야 합니다. CreateTable은 비동기식 작업입니다. 테이블이 ACTIVE 상태가 되어 사용 준비가 될 때까지 GetTableStatus는 CREATING 상태를 반환합니다.

### 단순형 기본 키를 사용하여 테이블 생성

이 코드는 단순형 기본 키("Name")를 사용하여 테이블을 만듭니다.

#### 포함 파일

```
#include <aws/core/Aws.h>
#include <aws/dynamodb/DynamoDBClient.h>
#include <aws/dynamodb/model/AttributeDefinition.h>
#include <aws/dynamodb/model/CreateTableRequest.h>
#include <aws/dynamodb/model/KeySchemaElement.h>
#include <aws/dynamodb/model/ProvisionedThroughput.h>
#include <aws/dynamodb/model/ScalarAttributeType.h>
#include <iostream>
```

#### 코드

```
//! Create an Amazon DynamoDB table.
/*!
    \sa createTable()
    \param tableName: Name for the DynamoDB table.
    \param primaryKey: Primary key for the DynamoDB table.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::createTable(const Aws::String &tableName,
                                   const Aws::String &primaryKey,
                                   const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    std::cout << "Creating table " << tableName <<
        " with a simple primary key: \"" << primaryKey << "\"." << std::endl;
```

```

    Aws::DynamoDB::Model::CreateTableRequest request;

    Aws::DynamoDB::Model::AttributeDefinition hashKey;
    hashKey.SetAttributeName(primaryKey);
    hashKey.SetAttributeType(Aws::DynamoDB::Model::ScalarAttributeType::S);
    request.AddAttributeDefinitions(hashKey);

    Aws::DynamoDB::Model::KeySchemaElement keySchemaElement;
    keySchemaElement.WithAttributeName(primaryKey).WithKeyType(
        Aws::DynamoDB::Model::KeyType::HASH);
    request.AddKeySchema(keySchemaElement);

    Aws::DynamoDB::Model::ProvisionedThroughput throughput;
    throughput.WithReadCapacityUnits(5).WithWriteCapacityUnits(5);
    request.SetProvisionedThroughput(throughput);
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::CreateTableOutcome &outcome = dynamoClient.CreateTable(
        request);
    if (outcome.IsSuccess()) {
        std::cout << "Table \""
                    << outcome.GetResult().GetTableDescription().GetTableName() <<
                    " created!" << std::endl;
    }
    else {
        std::cerr << "Failed to create table: " << outcome.GetError().GetMessage()
                  << std::endl;
        return false;
    }

    return waitTableActive(tableName, dynamoClient);
}

```

[전체 예제](#)를 참조하세요.

복합형 기본 키를 사용하여 테이블 생성

다른 [AttributeDefinition](#) 및 [KeySchemaElement](#)를 [CreateTableRequest](#)에 추가합니다.

포함 파일

```

#include <aws/core/Aws.h>
#include <aws/dynamodb/DynamoDBClient.h>
#include <aws/dynamodb/model/AttributeDefinition.h>

```

```
#include <aws/dynamodb/model/CreateTableRequest.h>
#include <aws/dynamodb/model/KeySchemaElement.h>
#include <aws/dynamodb/model/ProvisionedThroughput.h>
#include <aws/dynamodb/model/ScalarAttributeType.h>
#include <iostream>
```

## 코드

```
//! Create an Amazon DynamoDB table with a composite key.
/*!
    \sa createTableWithCompositeKey()
    \param tableName: Name for the DynamoDB table.
    \param partitionKey: Name for the partition (hash) key.
    \param sortKey: Name for the sort (range) key.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::createTableWithCompositeKey(const Aws::String &tableName,
                                                    const Aws::String &partitionKey,
                                                    const Aws::String &sortKey,
                                                    const
                                                    Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    std::cout << "Creating table " << tableName <<
        " with a composite primary key:\n" \
        "*" << partitionKey << " - partition key\n" \
        "*" << sortKey << " - sort key\n";

    Aws::DynamoDB::Model::CreateTableRequest request;

    Aws::DynamoDB::Model::AttributeDefinition hashKey1, hashKey2;
    hashKey1.WithAttributeName(partitionKey).WithAttributeType(
        Aws::DynamoDB::Model::ScalarAttributeType::S);
    request.AddAttributeDefinitions(hashKey1);
    hashKey2.WithAttributeName(sortKey).WithAttributeType(
        Aws::DynamoDB::Model::ScalarAttributeType::S);
    request.AddAttributeDefinitions(hashKey2);

    Aws::DynamoDB::Model::KeySchemaElement keySchemaElement1, keySchemaElement2;
    keySchemaElement1.WithAttributeName(partitionKey).WithKeyType(
        Aws::DynamoDB::Model::KeyType::HASH);
    request.AddKeySchema(keySchemaElement1);
```

```

keySchemaElement2.WithAttributeName(sortKey).WithKeyType(
    Aws::DynamoDB::Model::KeyType::RANGE);
request.AddKeySchema(keySchemaElement2);

Aws::DynamoDB::Model::ProvisionedThroughput throughput;
throughput.WithReadCapacityUnits(5).WithWriteCapacityUnits(5);
request.SetProvisionedThroughput(throughput);

request.SetTableName(tableName);

const Aws::DynamoDB::Model::CreateTableOutcome &outcome = dynamoClient.CreateTable(
    request);
if (outcome.IsSuccess()) {
    std::cout << "Table \""
        << outcome.GetResult().GetTableDescription().GetTableName() <<
        "\" was created!" << std::endl;
}
else {
    std::cerr << "Failed to create table:" << outcome.GetError().GetMessage()
        << std::endl;
    return false;
}

return waitTableActive(tableName, dynamoClient);
}

```

[전체 예제](#)는 GitHub에서 확인하세요.

## 테이블 나열

[DynamoDB 클라이언트](#)의 ListTables 메서드를 호출하면 특정 리전의 테이블을 나열할 수 있습니다.

## 포함 파일

```

#include <aws/core/Aws.h>
#include <aws/core/utils/Outcome.h>
#include <aws/dynamodb/DynamoDBClient.h>
#include <aws/dynamodb/model/ListTablesRequest.h>
#include <aws/dynamodb/model/ListTablesResult.h>
#include <iostream>

```

## 코드

```

//! List the Amazon DynamoDB tables for the current AWS account.
/*!
  \sa listTables()
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */

bool AwsDoc::DynamoDB::listTables(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::ListTablesRequest listTablesRequest;
    listTablesRequest.SetLimit(50);
    do {
        const Aws::DynamoDB::Model::ListTablesOutcome &outcome =
dynamoClient.ListTables(
            listTablesRequest);
        if (!outcome.IsSuccess()) {
            std::cout << "Error: " << outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        for (const auto &tableName: outcome.GetResult().GetTableNames())
            std::cout << tableName << std::endl;
        listTablesRequest.SetExclusiveStartTableName(
            outcome.GetResult().GetLastEvaluatedTableName());

    } while (!listTablesRequest.GetExclusiveStartTableName().empty());

    return true;
}

```

기본적으로 호출당 최대 100개의 테이블이 반환됩니다. 반환된 [ListTablesOutcome](#) 객체에 `GetExclusiveStartTableName`을 사용하면 마지막으로 평가된 테이블을 가져올 수 있습니다. 이 값을 사용하여 이전 목록의 마지막으로 반환된 값 다음에 이어지는 목록을 시작할 수 있습니다.

[전체 예제](#)를 참조하세요.

## 테이블 정보 검색

테이블에 대한 자세한 정보는 [DynamoDB 클라이언트](#)의 `DescribeTable` 메서드를 호출하여 확인할 수 있습니다.

## 포함 파일

```
#include <aws/core/Aws.h>
#include <aws/dynamodb/DynamoDBClient.h>
#include <aws/dynamodb/model/DescribeTableRequest.h>
#include <iostream>
```

## 코드

```
//! Describe an Amazon DynamoDB table.
/*!
 \sa describeTable()
 \param tableName: The DynamoDB table name.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::describeTable(const Aws::String &tableName,
                                     const Aws::Client::ClientConfiguration
                                     &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::DescribeTableOutcome &outcome =
    dynamoClient.DescribeTable(
        request);

    if (outcome.IsSuccess()) {
        const Aws::DynamoDB::Model::TableDescription &td =
        outcome.GetResult().GetTable();
        std::cout << "Table name   : " << td.GetTableName() << std::endl;
        std::cout << "Table ARN    : " << td.GetTableArn() << std::endl;
        std::cout << "Status      : "
            << Aws::DynamoDB::Model::TableStatusMapper::GetNameForTableStatus(
                td.GetTableStatus()) << std::endl;
        std::cout << "Item count   : " << td.GetItemCount() << std::endl;
        std::cout << "Size (bytes): " << td.GetTableSizeBytes() << std::endl;

        const Aws::DynamoDB::Model::ProvisionedThroughputDescription &ptd =
        td.GetProvisionedThroughput();
        std::cout << "Throughput" << std::endl;
        std::cout << "  Read Capacity : " << ptd.GetReadCapacityUnits() << std::endl;
```

```

        std::cout << "  Write Capacity: " << ptd.GetWriteCapacityUnits() << std::endl;

        const Aws::Vector<Aws::DynamoDB::Model::AttributeDefinition> &ad =
td.GetAttributeDefinitions();
        std::cout << "Attributes" << std::endl;
        for (const auto &a: ad)
            std::cout << "  " << a.GetAttributeName() << " (" <<

Aws::DynamoDB::Model::ScalarAttributeTypeMapper::GetNameForScalarAttributeType(
            a.GetAttributeType()) <<
            ")" << std::endl;
    }
    else {
        std::cerr << "Failed to describe table: " << outcome.GetError().GetMessage();
    }

    return outcome.IsSuccess();
}

```

[전체 예제](#)는 GitHub에서 확인하세요.

## 테이블 수정

[DynamoDB 클라이언트](#)의 UpdateTable 메서드를 호출하면 테이블의 프로비저닝 처리량 값을 언제 든지 수정할 수 있습니다.

## 포함 파일

```

#include <aws/core/Aws.h>
#include <aws/core/utils/Outcome.h>
#include <aws/dynamodb/DynamoDBClient.h>
#include <aws/dynamodb/model/ProvisionedThroughput.h>
#include <aws/dynamodb/model/UpdateTableRequest.h>
#include <iostream>

```

## 코드

```

//! Update a DynamoDB table.
/*!
    \sa updateTable()
    \param tableName: Name for the DynamoDB table.
    \param readCapacity: Provisioned read capacity.

```

```

\param writeCapacity: Provisioned write capacity.
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::updateTable(const Aws::String &tableName,
                                   long long readCapacity, long long writeCapacity,
                                   const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    std::cout << "Updating " << tableName << " with new provisioned throughput values"
                << std::endl;
    std::cout << "Read capacity : " << readCapacity << std::endl;
    std::cout << "Write capacity: " << writeCapacity << std::endl;

    Aws::DynamoDB::Model::UpdateTableRequest request;
    Aws::DynamoDB::Model::ProvisionedThroughput provisionedThroughput;
    provisionedThroughput.WithReadCapacityUnits(readCapacity).WithWriteCapacityUnits(
        writeCapacity);
    request.WithProvisionedThroughput(provisionedThroughput).WithTableName(tableName);

    const Aws::DynamoDB::Model::UpdateTableOutcome &outcome = dynamoClient.UpdateTable(
        request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated the table." << std::endl;
    } else {
        const Aws::DynamoDB::DynamoDBError &error = outcome.GetError();
        if (error.GetErrorType() == Aws::DynamoDB::DynamoDBErrors::VALIDATION &&
            error.GetMessage().find("The provisioned throughput for the table will not
change") != std::string::npos) {
            std::cout << "The provisioned throughput for the table will not change." <<
std::endl;
        } else {
            std::cerr << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }

    return waitTableActive(tableName, dynamoClient);
}

```

[전체 예제](#)를 참조하세요.

## 테이블 삭제

[DynamoDB 클라이언트](#)의 DeleteTable 메서드를 호출하고 테이블 이름을 이 메서드에 전달합니다.

### 포함 파일

```
#include <aws/core/Aws.h>
#include <aws/dynamodb/DynamoDBClient.h>
#include <aws/dynamodb/model/DeleteTableRequest.h>
#include <iostream>
```

### 코드

```
//! Delete an Amazon DynamoDB table.
/*!
 \sa deleteTable()
 \param tableName: The DynamoDB table name.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::deleteTable(const Aws::String &tableName,
                                    const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result = dynamoClient.DeleteTable(
        request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
                    << result.GetResult().GetTableDescription().GetTableName()
                    << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
                  << std::endl;
    }

    return result.IsSuccess();
}
```

[전체 예제](#)는 GitHub에서 확인하세요.

## 추가 정보

- Amazon DynamoDB 개발자 안내서의 [테이블 작업 가이드라인](#)
- Amazon DynamoDB 개발자 안내서의 [DynamoDB에서 테이블 작업](#)

## DynamoDB에서 항목 작업

DynamoDB에서 항목은 각각 이름과 값을 갖는 속성 모음입니다. 속성 값은 스칼라, 세트 또는 문서 유형일 수 있습니다. 자세한 내용은 Amazon DynamoDB 개발자 안내서에서 [이름 지정 규칙 및 데이터 형식](#)을 참조하세요.

### 테이블에서 항목 가져오기

[DynamoDB 클라이언트](#)의 `GetItem` 메서드를 호출합니다. 테이블 이름과 원하는 항목의 프라이머리 키 값이 포함된 [GetItemRequest](#) 객체를 전달합니다. 이 메서드는 [GetItemResult](#) 객체를 반환합니다.

반환된 `GetItemResult` 객체의 `GetItem()` 메서드를 사용하여 해당 항목과 연관된 키 `Aws::String`과 값 [AttributeValue](#) 페어로 구성된 `Aws::Map`을 가져올 수 있습니다.

### 포함 파일

```
#include <aws/core/Aws.h>
#include <aws/dynamodb/DynamoDBClient.h>
#include <aws/dynamodb/model/AttributeDefinition.h>
#include <aws/dynamodb/model/GetItemRequest.h>
#include <iostream>
```

### 코드

```
//! Get an item from an Amazon DynamoDB table.
/*!
    \sa getItem()
    \param tableName: The table name.
    \param partitionKey: The partition key.
    \param partitionValue: The value for the partition key.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/

bool AwsDoc::DynamoDB::getItem(const Aws::String &tableName,
```

```

        const Aws::String &partitionKey,
        const Aws::String &partitionValue,
        const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
    Aws::DynamoDB::Model::GetItemRequest request;

    // Set up the request.
    request.SetTableName(tableName);
    request.AddKey(partitionKey,
        Aws::DynamoDB::Model::AttributeValue().SetS(partitionValue));

    // Retrieve the item's fields and values.
    const Aws::DynamoDB::Model::GetItemOutcome &outcome =
dynamoClient.GetItem(request);
    if (outcome.IsSuccess()) {
        // Reference the retrieved fields/values.
        const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> &item =
outcome.GetResult().GetItem();
        if (!item.empty()) {
            // Output each retrieved field and its value.
            for (const auto &i: item)
                std::cout << "Values: " << i.first << ": " << i.second.GetS()
                    << std::endl;
        }
        else {
            std::cout << "No item found with the key " << partitionKey << std::endl;
        }
    }
    else {
        std::cerr << "Failed to get item: " << outcome.GetError().GetMessage();
    }

    return outcome.IsSuccess();
}

```

[전체 예제](#)는 GitHub에서 확인하세요.

## 테이블에 항목 추가

각 항목을 나타내는 키 `Aws::String` 및 값 [AttributeValue](#) 페어를 생성합니다. 여기에는 테이블의 기본 키 필드 값이 포함되어야 합니다. 기본 키로 식별되는 항목이 이미 존재하면 필드가 요청에 따라 업데이트됩니다. `AddItem` 메서드를 사용하여 [PutItemRequest](#)에 추가합니다.

## 포함 파일

```
#include <aws/core/Aws.h>
#include <aws/dynamodb/DynamoDBClient.h>
#include <aws/dynamodb/model/AttributeDefinition.h>
#include <aws/dynamodb/model/PutItemRequest.h>
#include <aws/dynamodb/model/PutItemResult.h>
#include <iostream>
```

## 코드

```
//! Put an item in an Amazon DynamoDB table.
/*!
    \sa putItem()
    \param tableName: The table name.
    \param artistKey: The artist key. This is the partition key for the table.
    \param artistValue: The artist value.
    \param albumTitleKey: The album title key.
    \param albumTitleValue: The album title value.
    \param awardsKey: The awards key.
    \param awardsValue: The awards value.
    \param songTitleKey: The song title key.
    \param songTitleValue: The song title value.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::putItem(const Aws::String &tableName,
                                const Aws::String &artistKey,
                                const Aws::String &artistValue,
                                const Aws::String &albumTitleKey,
                                const Aws::String &albumTitleValue,
                                const Aws::String &awardsKey,
                                const Aws::String &awardsValue,
                                const Aws::String &songTitleKey,
                                const Aws::String &songTitleValue,
                                const Aws::Client::ClientConfiguration
                                &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::PutItemRequest putItemRequest;
    putItemRequest.SetTableName(tableName);

    putItemRequest.AddItem(artistKey, Aws::DynamoDB::Model::AttributeValue().SetS(
```

```

        artistValue)); // This is the hash key.
putItemRequest.AddItem(albumTitleKey, Aws::DynamoDB::Model::AttributeValue().SetS(
    albumTitleValue));
putItemRequest.AddItem(awardsKey,
    Aws::DynamoDB::Model::AttributeValue().SetS(awardsValue));
putItemRequest.AddItem(songTitleKey,
    Aws::DynamoDB::Model::AttributeValue().SetS(songTitleValue));

const Aws::DynamoDB::Model::PutItemOutcome outcome = dynamoClient.PutItem(
    putItemRequest);
if (outcome.IsSuccess()) {
    std::cout << "Successfully added Item!" << std::endl;
}
else {
    std::cerr << outcome.GetError().GetMessage() << std::endl;
    return false;
}

return waitTableActive(tableName, dynamoClient);
}

```

[전체 예제](#)는 GitHub에서 확인하세요.

## 테이블의 기존 항목 업데이트

테이블에 이미 존재하는 항목의 속성을 업데이트하려면 DynamoDBClient의 UpdateItem 메서드를 사용합니다. 이때 테이블 이름, 프라이머리 키 값, 업데이트할 필드 및 해당 값을 제공해야 합니다.

가져옵니다.

```

#include <aws/core/Aws.h>
#include <aws/dynamodb/DynamoDBClient.h>
#include <aws/dynamodb/model/UpdateItemRequest.h>
#include <aws/dynamodb/model/UpdateItemResult.h>
#include <iostream>

```

## 코드

```

//! Update an Amazon DynamoDB table item.
/*!
    \sa updateItem()

```

```

\param tableName: The table name.
\param partitionKey: The partition key.
\param partitionValue: The value for the partition key.
\param attributeKey: The key for the attribute to be updated.
\param attributeValue: The value for the attribute to be updated.
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/

/*
 * The example code only sets/updates an attribute value. It processes
 * the attribute value as a string, even if the value could be interpreted
 * as a number. Also, the example code does not remove an existing attribute
 * from the key value.
 */

bool AwsDoc::DynamoDB::updateItem(const Aws::String &tableName,
                                   const Aws::String &partitionKey,
                                   const Aws::String &partitionValue,
                                   const Aws::String &attributeKey,
                                   const Aws::String &attributeValue,
                                   const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    // *** Define UpdateItem request arguments.
    // Define TableName argument.
    Aws::DynamoDB::Model::UpdateItemRequest request;
    request.SetTableName(tableName);

    // Define KeyName argument.
    Aws::DynamoDB::Model::AttributeValue attribValue;
    attribValue.SetS(partitionValue);
    request.AddKey(partitionKey, attribValue);

    // Construct the SET update expression argument.
    Aws::String update_expression("SET #a = :valueA");
    request.SetUpdateExpression(update_expression);

    // Construct attribute name argument.
    Aws::Map<Aws::String, Aws::String> expressionAttributeNames;
    expressionAttributeNames["#a"] = attributeKey;
    request.SetExpressionAttributeNames(expressionAttributeNames);

```

```
// Construct attribute value argument.
Aws::DynamoDB::Model::AttributeValue attributeUpdatedValue;
attributeUpdatedValue.SetS(attributeValue);
Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
expressionAttributeValues;
expressionAttributeValues[":valueA"] = attributeUpdatedValue;
request.SetExpressionAttributeValues(expressionAttributeValues);

// Update the item.
const Aws::DynamoDB::Model::UpdateItemOutcome &outcome = dynamoClient.UpdateItem(
    request);
if (outcome.IsSuccess()) {
    std::cout << "Item was updated" << std::endl;
} else {
    std::cerr << outcome.GetError().GetMessage() << std::endl;
    return false;
}

return waitTableActive(tableName, dynamoClient);
}
```

[전체 예제](#)를 참조하세요.

## 추가 정보

- Amazon DynamoDB 개발자 안내서의 [항목 작업 지침](#)
- Amazon DynamoDB 개발자 안내서의 [DynamoDB에서 항목 작업](#)

## AWS SDK for C++를 사용한 Amazon EC2 예제

Amazon Elastic Compute Cloud(Amazon EC2)는 확장 가능한 컴퓨팅 용량 즉, Amazon 데이터 센터 내 서버를 제공하는 웹 서비스로, 사용자가 소프트웨어 시스템을 빌드하고 호스팅하는 데 활용할 수 있습니다. 다음 예제를 사용하면 AWS SDK for C++로 [Amazon EC2](#)를 프로그래밍할 수 있습니다.

### Note

이 안내서에는 특정 기술을 시연하는 데 필요한 코드만 제공되며 [전체 예제 코드는 GitHub에서 확인할 수 있습니다](#). GitHub에서 단일 소스 파일을 다운로드하거나, 리포지토리를 로컬에 복제하여 모든 예제를 가져오고 빌드하고 실행할 수 있습니다.

## 주제

- [Amazon EC2 인스턴스 관리](#)
- [Amazon EC2에서 탄력적 IP 주소 사용](#)
- [Amazon EC2에 리전 및 가용 영역 사용](#)
- [Amazon EC2 키 페어로 작업](#)
- [Amazon EC2의 보안 그룹 작업](#)

## Amazon EC2 인스턴스 관리

### 사전 조건

시작하기 전에 [AWS SDK for C++ 사용 시작하기](#)를 읽어보시기 바랍니다.

예제 코드를 다운로드하고 [코드 예제 시작하기](#)에 설명된 대로 솔루션을 빌드합니다.

예제를 실행하려면 코드가 요청을 수행하는 데 사용하는 사용자 프로필이 AWS에서 적절한 권한(서비스 및 동작에 대한)을 보유하고 있어야 합니다. 자세한 내용은 [AWS 자격 증명 제공](#)을 참조하세요.

### 인스턴스 생성

EC2Client의 RunInstances 함수를 호출하여 새 Amazon EC2 인스턴스를 생성하고, 사용할 [Amazon Machine Image\(AMI\)](#)와 [인스턴스 유형](#)을 포함한 [RunInstancesRequest](#)를 해당 함수에 제공합니다.

### 포함 파일

```
#include <aws/core/Aws.h>
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/RunInstancesRequest.h>
#include <iostream>
```

### 코드

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);

Aws::EC2::Model::RunInstancesRequest runRequest;
runRequest.SetImageId(amiId);
runRequest.SetInstanceType(Aws::EC2::Model::InstanceType::t1_micro);
runRequest.SetMinCount(1);
```

```

runRequest.SetMaxCount(1);

Aws::EC2::Model::RunInstancesOutcome runOutcome = ec2Client.RunInstances(
    runRequest);
if (!runOutcome.IsSuccess()) {
    std::cerr << "Failed to launch EC2 instance " << instanceName <<
        " based on ami " << amiId << ":" <<
        runOutcome.GetError().GetMessage() << std::endl;
    return false;
}

const Aws::Vector<Aws::EC2::Model::Instance> &instances =
runOutcome.GetResult().GetInstances();
if (instances.empty()) {
    std::cerr << "Failed to launch EC2 instance " << instanceName <<
        " based on ami " << amiId << ":" <<
        runOutcome.GetError().GetMessage() << std::endl;
    return false;
}

```

[전체 예제](#)를 참조하세요.

## 인스턴스 시작

Amazon EC2 인스턴스를 시작하려면 EC2Client의 StartInstances 함수를 호출하고, 시작할 인스턴스의 ID가 포함된 [StartInstancesRequest](#)를 해당 함수에 제공합니다.

## 포함 파일

```

#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/StartInstancesRequest.h>

```

## 코드

```

Aws::EC2::EC2Client ec2Client(clientConfiguration);

Aws::EC2::Model::StartInstancesRequest startRequest;
startRequest.AddInstanceIds(instanceId);
startRequest.SetDryRun(true);

Aws::EC2::Model::StartInstancesOutcome dryRunOutcome =
ec2Client.StartInstances(startRequest);

```

```

    if (dryRunOutcome.IsSuccess()) {
        std::cerr
            << "Failed dry run to start instance. A dry run should trigger an
error."
            << std::endl;
        return false;
    } else if (dryRunOutcome.GetError().GetErrorType() !=
        Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to start instance " << instanceId << ": "
            << dryRunOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    startRequest.SetDryRun(false);
    Aws::EC2::Model::StartInstancesOutcome startInstancesOutcome =
ec2Client.StartInstances(startRequest);

    if (!startInstancesOutcome.IsSuccess()) {
        std::cout << "Failed to start instance " << instanceId << ": " <<
            startInstancesOutcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully started instance " << instanceId <<
            std::endl;
    }
}

```

[전체 예제](#)를 참조하세요.

## 인스턴스 중지

Amazon EC2 인스턴스를 중지하려면 EC2Client의 StopInstances 함수를 호출하고, 중지할 인스턴스의 ID가 포함된 [StopInstancesRequest](#)를 해당 함수에 제공합니다.

## 포함 파일

```
#include <aws/ec2/model/StopInstancesRequest.h>
```

## 코드

```

Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::StopInstancesRequest request;
request.AddInstanceIds(instanceId);
request.SetDryRun(true);

```

```

    Aws::EC2::Model::StopInstancesOutcome dryRunOutcome =
ec2Client.StopInstances(request);
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
            << "Failed dry run to stop instance. A dry run should trigger an
error."
            << std::endl;
        return false;
    } else if (dryRunOutcome.GetError().GetErrorType() !=
        Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to stop instance " << instanceId << ": "
            << dryRunOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    request.SetDryRun(false);
    Aws::EC2::Model::StopInstancesOutcome outcome = ec2Client.StopInstances(request);
    if (!outcome.IsSuccess()) {
        std::cout << "Failed to stop instance " << instanceId << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully stopped instance " << instanceId <<
            std::endl;
    }
}

```

[전체 예제](#)를 참조하세요.

## 인스턴스 재부팅

Amazon EC2 인스턴스를 재부팅하려면 EC2Client의 RebootInstances 함수를 호출하고, 재부팅할 인스턴스의 ID가 포함된 [RebootInstancesRequest](#)를 해당 함수에 제공합니다.

## 포함 파일

```

#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/RebootInstancesRequest.h>
#include <iostream>

```

## 코드

```

Aws::EC2::EC2Client ec2Client(clientConfiguration);

Aws::EC2::Model::RebootInstancesRequest request;

```

```

request.AddInstanceIds(instanceId);
request.SetDryRun(true);

Aws::EC2::Model::RebootInstancesOutcome dry_run_outcome =
ec2Client.RebootInstances(request);
if (dry_run_outcome.IsSuccess()) {
    std::cerr
        << "Failed dry run to reboot on instance. A dry run should trigger an
error."
        <<
        std::endl;
    return false;
} else if (dry_run_outcome.GetError().GetErrorType()
    != Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
    std::cout << "Failed dry run to reboot instance " << instanceId << ": "
        << dry_run_outcome.GetError().GetMessage() << std::endl;
    return false;
}

request.SetDryRun(false);
Aws::EC2::Model::RebootInstancesOutcome outcome =
ec2Client.RebootInstances(request);
if (!outcome.IsSuccess()) {
    std::cout << "Failed to reboot instance " << instanceId << ": " <<
        outcome.GetError().GetMessage() << std::endl;
} else {
    std::cout << "Successfully rebooted instance " << instanceId <<
        std::endl;
}

```

[전체 예제](#)를 참조하세요.

## 인스턴스 설명

인스턴스를 나열하려면 [DescribeInstancesRequest](#)를 생성하고 EC2Client의 DescribeInstances 함수를 호출합니다. 그러면 AWS 계정과 AWS 리전에 대한 Amazon EC2 인스턴스를 나열하는 데 사용할 수 있는 [DescribeInstancesResponse](#) 객체가 반환됩니다.

인스턴스는 예약별로 그룹화됩니다. 각 예약은 인스턴스를 시작하는 StartInstances 호출에 해당합니다. 인스턴스를 나열하려면 먼저 DescribeInstancesResponse 클래스의 GetReservations 함수를 호출한 다음 반환된 각 Reservation 객체에서 getInstances를 호출해야 합니다.

## 포함 파일

```
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/DescribeInstancesRequest.h>
#include <aws/ec2/model/DescribeInstancesResponse.h>
#include <iomanip>
#include <iostream>
```

## 코드

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::DescribeInstancesRequest request;
bool header = false;
bool done = false;
while (!done) {
    Aws::EC2::Model::DescribeInstancesOutcome outcome =
ec2Client.DescribeInstances(request);
    if (outcome.IsSuccess()) {
        if (!header) {
            std::cout << std::left <<
                std::setw(48) << "Name" <<
                std::setw(20) << "ID" <<
                std::setw(25) << "Ami" <<
                std::setw(15) << "Type" <<
                std::setw(15) << "State" <<
                std::setw(15) << "Monitoring" << std::endl;
            header = true;
        }

        const std::vector<Aws::EC2::Model::Reservation> &reservations =
            outcome.GetResult().GetReservations();

        for (const auto &reservation: reservations) {
            const std::vector<Aws::EC2::Model::Instance> &instances =
                reservation.GetInstances();
            for (const auto &instance: instances) {
                Aws::String instanceStateString =

                Aws::EC2::Model::InstanceStateNameMapper::GetNameForInstanceStateName(
                    instance.GetState().GetName());

                Aws::String typeString =

                Aws::EC2::Model::InstanceTypeMapper::GetNameForInstanceType(
                    instance.GetInstanceType());
```

```

        Aws::String monitorString =

    Aws::EC2::Model::MonitoringStateMapper::GetNameForMonitoringState(
        instance.GetMonitoring().GetState());
    Aws::String name = "Unknown";

    const std::vector<Aws::EC2::Model::Tag> &tags = instance.GetTags();
    auto nameIter = std::find_if(tags.cbegin(), tags.cend(),
        [](const Aws::EC2::Model::Tag &tag) {
            return tag.GetKey() == "Name";
        });
    if (nameIter != tags.cend()) {
        name = nameIter->GetValue();
    }
    std::cout <<
        std::setw(48) << name <<
        std::setw(20) << instance.GetInstanceId() <<
        std::setw(25) << instance.GetImageId() <<
        std::setw(15) << typeString <<
        std::setw(15) << instanceStateString <<
        std::setw(15) << monitorString << std::endl;
    }
}

if (!outcome.GetResult().GetNextToken().empty()) {
    request.SetNextToken(outcome.GetResult().GetNextToken());
} else {
    done = true;
}
} else {
    std::cerr << "Failed to describe EC2 instances:" <<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}
}
}

```

결과는 페이지 단위로 제공됩니다. 결과 객체의 `GetNextToken` 함수에서 반환된 값을 원래 요청 객체의 `SetNextToken` 함수에 전달한 후 동일한 요청 객체를 다음 `DescribeInstances` 호출에 사용하면 추가 결과를 얻을 수 있습니다.

[전체 예제](#)를 참조하세요.

## 인스턴스 모니터링 활성화

CPU와 네트워크 사용률, 사용 가능한 메모리, 남은 디스크 공간 등 Amazon EC2 인스턴스의 다양한 측면을 모니터링할 수 있습니다. 인스턴스 모니터링에 대한 자세한 내용은 Amazon EC2 사용 설명서에서 [Amazon EC2 모니터링](#)을 참조하세요.

인스턴스 모니터링을 시작하려면 모니터링할 인스턴스의 ID로 [MonitorInstancesRequest](#)를 생성한 후 EC2Client의 MonitorInstances 함수에 전달합니다.

### 포함 파일

```
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/MonitorInstancesRequest.h>
#include <aws/ec2/model/UnmonitorInstancesRequest.h>
#include <iostream>
```

### 코드

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::MonitorInstancesRequest request;
request.AddInstanceIds(instanceId);
request.SetDryRun(true);

Aws::EC2::Model::MonitorInstancesOutcome dryRunOutcome =
ec2Client.MonitorInstances(request);
if (dryRunOutcome.IsSuccess()) {
    std::cerr
        << "Failed dry run to enable monitoring on instance. A dry run should
trigger an error."
        <<
        std::endl;
    return false;
} else if (dryRunOutcome.GetError().GetErrorType()
    != Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
    std::cerr << "Failed dry run to enable monitoring on instance " <<
        instanceId << ": " << dryRunOutcome.GetError().GetMessage() <<
        std::endl;
    return false;
}

request.SetDryRun(false);
Aws::EC2::Model::MonitorInstancesOutcome monitorInstancesOutcome =
ec2Client.MonitorInstances(request);
```

```

if (!monitorInstancesOutcome.IsSuccess()) {
    std::cerr << "Failed to enable monitoring on instance " <<
        instanceId << ": " <<
        monitorInstancesOutcome.GetError().GetMessage() << std::endl;
} else {
    std::cout << "Successfully enabled monitoring on instance " <<
        instanceId << std::endl;
}

```

[전체 예제](#)를 참조하세요.

## 인스턴스 모니터링 비활성화

인스턴스 모니터링을 중지하려면 모니터링을 중지할 인스턴스의 ID로 [UnmonitorInstancesRequest](#)를 생성한 후 EC2Client의 UnmonitorInstances 함수에 전달합니다.

## 포함 파일

```

#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/MonitorInstancesRequest.h>
#include <aws/ec2/model/UnmonitorInstancesRequest.h>
#include <iostream>

```

## 코드

```

Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::UnmonitorInstancesRequest unrequest;
unrequest.AddInstanceIds(instanceId);
unrequest.SetDryRun(true);

Aws::EC2::Model::UnmonitorInstancesOutcome dryRunOutcome =
ec2Client.UnmonitorInstances(unrequest);
if (dryRunOutcome.IsSuccess()) {
    std::cerr
        << "Failed dry run to disable monitoring on instance. A dry run should
trigger an error."
        <<
        std::endl;
    return false;
} else if (dryRunOutcome.GetError().GetErrorType() !=
    Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
    std::cout << "Failed dry run to disable monitoring on instance " <<
        instanceId << ": " << dryRunOutcome.GetError().GetMessage() <<

```

```

        std::endl;
        return false;
    }

    unrequest.SetDryRun(false);
    Aws::EC2::Model::UnmonitorInstancesOutcome unmonitorInstancesOutcome =
    ec2Client.UnmonitorInstances(unrequest);
    if (!unmonitorInstancesOutcome.IsSuccess()) {
        std::cout << "Failed to disable monitoring on instance " << instanceId
        << ": " << unmonitorInstancesOutcome.GetError().GetMessage() <<
        std::endl;
    } else {
        std::cout << "Successfully disable monitoring on instance " <<
        instanceId << std::endl;
    }
}

```

[전체 예제](#)를 참조하세요.

## 추가 정보

- Amazon EC2 API 참조의 [RunInstances](#)
- Amazon EC2 API 참조의 [DescribeInstances](#)
- Amazon EC2 API 참조의 [StartInstances](#)
- Amazon EC2 API 참조의 [StopInstances](#)
- Amazon EC2 API 참조의 [RebootInstances](#)
- Amazon EC2 API 참조의 [DescribeInstances](#)
- Amazon EC2 API 참조의 [MonitorInstances](#)
- Amazon EC2 API 참조의 [UnmonitorInstances](#)

## Amazon EC2에서 탄력적 IP 주소 사용

### 사전 조건

시작하기 전에 [AWS SDK for C++ 사용 시작하기](#)를 읽어보시기 바랍니다.

예제 코드를 다운로드하고 [코드 예제 시작하기](#)에 설명된 대로 솔루션을 빌드합니다.

예제를 실행하려면 코드가 요청을 수행하는 데 사용하는 사용자 프로필이 AWS에서 적절한 권한(서비스 및 동작에 대한)을 보유하고 있어야 합니다. 자세한 내용은 [AWS 자격 증명 제공](#)을 참조하세요.

## 탄력적 IP 주소 할당

탄력적 IP 주소를 사용하려면 먼저 계정에 주소를 할당한 후 인스턴스 또는 네트워크 인터페이스와 연결합니다.

탄력적 IP 주소를 할당하려면 EC2Client의 `AllocateAddress` 함수를 네트워크 유형(클래식 EC2 또는 VPC)이 포함된 [AllocateAddressRequest](#) 객체와 함께 호출합니다.

### Warning

EC2-Classic은 2022년 8월 15일에 사용 중지될 예정입니다. EC2-Classic에서 VPC로 마이그레이션하는 것이 좋습니다. 자세한 내용은 [Linux 인스턴스용 Amazon EC2 사용 설명서](#) 또는 [Windows 인스턴스용 Amazon EC2 사용 설명서](#)의 EC2-Classic에서 VPC로 마이그레이션을 참조하세요. 또는 블로그 게시물 [EC2-Classic 네트워킹은 사용 중지 중입니다 - 준비 방법은 다음과 같습니다](#)를 참조하세요.

응답 객체의 [AllocateAddressResponse](#) 클래스에는 할당 ID가 포함되어 있는데, 이 할당 ID를 사용하여 주소를 인스턴스와 연결할 수 있습니다. 할당 ID와 인스턴스 ID를 EC2Client의 `AssociateAddress` 함수에 [AssociateAddressRequest](#)로 전달하면 됩니다.

## 포함 파일

```
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/AllocateAddressRequest.h>
#include <aws/ec2/model/AssociateAddressRequest.h>
#include <iostream>
```

## 코드

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);

Aws::EC2::Model::AllocateAddressRequest request;
request.SetDomain(Aws::EC2::Model::DomainType::vpc);

const Aws::EC2::Model::AllocateAddressOutcome outcome =
    ec2Client.AllocateAddress(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Failed to allocate Elastic IP address:" <<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}
```

```

}
const Aws::EC2::Model::AllocateAddressResponse &response = outcome.GetResult();
allocationID = response.GetAllocationId();
publicIPAddress = response.GetPublicIp();

Aws::EC2::Model::AssociateAddressRequest associate_request;
associate_request.SetInstanceId(instanceId);
associate_request.SetAllocationId(allocationID);

const Aws::EC2::Model::AssociateAddressOutcome associate_outcome =
    ec2Client.AssociateAddress(associate_request);
if (!associate_outcome.IsSuccess()) {
    std::cerr << "Failed to associate Elastic IP address " << allocationID
                << " with instance " << instanceId << ":" <<
                associate_outcome.GetError().GetMessage() << std::endl;
    return false;
}

std::cout << "Successfully associated Elastic IP address " << allocationID
            << " with instance " << instanceId << std::endl;

```

[전체 예제](#)를 참조하세요.

## 탄력적 IP 주소 설명

계정에 지정된 탄력적 IP 주소를 나열하려면 EC2Client의 DescribeAddresses 함수를 호출합니다. 반환되는 결과 객체에는 [DescribeAddressesResponse](#)가 포함되어 있는데, 이를 사용하여 계정의 탄력적 IP 주소를 나타내는 [Address](#) 객체 목록을 얻을 수 있습니다.

## 포함 파일

```

#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/DescribeAddressesRequest.h>
#include <aws/ec2/model/DescribeAddressesResponse.h>
#include <iomanip>
#include <iostream>

```

## 코드

```

Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::DescribeAddressesRequest request;

```

```

    Aws::EC2::Model::DescribeAddressesOutcome outcome =
    ec2Client.DescribeAddresses(request);
    if (outcome.IsSuccess()) {
        std::cout << std::left << std::setw(20) << "InstanceId" <<
            std::setw(15) << "Public IP" << std::setw(10) << "Domain" <<
            std::setw(30) << "Allocation ID" << std::setw(25) <<
            "NIC ID" << std::endl;

        const Aws::Vector<Aws::EC2::Model::Address> &addresses =
        outcome.GetResult().GetAddresses();
        for (const auto &address: addresses) {
            Aws::String domainString =
                Aws::EC2::Model::DomainTypeMapper::GetNameForDomainType(
                    address.GetDomain());

            std::cout << std::left << std::setw(20) <<
                address.GetInstanceId() << std::setw(15) <<
                address.GetPublicIp() << std::setw(10) << domainString <<
                std::setw(30) << address.GetAllocationId() << std::setw(25)
                << address.GetNetworkInterfaceId() << std::endl;
        }
    } else {
        std::cerr << "Failed to describe Elastic IP addresses:" <<
            outcome.GetError().GetMessage() << std::endl;
    }
}

```

[전체 예제](#)를 참조하세요.

## 탄력적 IP 주소 릴리스

탄력적 IP 주소를 릴리스하려면 EC2Client의 ReleaseAddress 함수를 호출하고 릴리스할 탄력적 IP 주소의 할당 ID가 포함된 [ReleaseAddressRequest](#)를 해당 함수에 전달합니다.

## 포함 파일

```

#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/ReleaseAddressRequest.h>
#include <iostream>

```

## 코드

```

Aws::EC2::EC2Client ec2(clientConfiguration);

```

```

Aws::EC2::Model::ReleaseAddressRequest request;
request.SetAllocationId(allocationID);

Aws::EC2::Model::ReleaseAddressOutcome outcome = ec2.ReleaseAddress(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Failed to release Elastic IP address " <<
        allocationID << ":" << outcome.GetError().GetMessage() <<
        std::endl;
} else {
    std::cout << "Successfully released Elastic IP address " <<
        allocationID << std::endl;
}

```

탄력적 IP 주소를 릴리스하고 나면 이 주소가 AWS IP 주소 풀로 릴리스되어 이후부터는 사용할 수 없게 됩니다. 해당 주소와 통신하는 모든 서버 또는 장치와 DNS 레코드를 업데이트해야 합니다. 이미 릴리스한 탄력적 IP 주소를 릴리스하려고 할 때 주소가 이미 다른 AWS 계정에 할당된 경우 AuthFailure 오류가 발생합니다.

기본 VPC를 사용하는 경우 탄력적 IP 주소를 릴리스하면 해당 주소가 연결된 모든 인스턴스와의 연결이 자동으로 해제됩니다. 탄력적 IP 주소를 릴리스하지 않고 연결을 해제하려면 EC2Client의 DisassociateAddress 함수를 사용합니다.

기본이 아닌 VPC를 사용하는 경우, 릴리스하기 전에 반드시 DisassociateAddress를 사용해 탄력적 IP 주소를 연결 해제합니다. 그렇지 않으면 Amazon EC2에서 오류(InvalidIPAddress.InUse)를 반환합니다.

[전체 예제](#)를 참조하세요.

## 추가 정보

- Amazon EC2 사용 설명서의 [탄력적 IP 주소](#)
- Amazon EC2 API 참조의 [AllocateAddress](#)
- Amazon EC2 API 참조의 [DescribeAddresses](#)
- Amazon EC2 API 참조의 [ReleaseAddress](#)

## Amazon EC2에 리전 및 가용 영역 사용

### 사전 조건

시작하기 전에 [AWS SDK for C++ 사용 시작하기](#)를 읽어보시기 바랍니다.

예제 코드를 다운로드하고 [코드 예제 시작하기](#)에 설명된 대로 솔루션을 빌드합니다.

예제를 실행하려면 코드가 요청을 수행하는 데 사용하는 사용자 프로필이 AWS에서 적절한 권한(서비스 및 동작에 대한)을 보유하고 있어야 합니다. 자세한 내용은 [AWS 자격 증명 제공](#)을 참조하세요.

## 리전 설명

AWS 계정에서 사용 가능한 AWS 리전을 나열하려면 EC2Client의 DescribeRegions 함수를 [DescribeRegionsRequest](#)와 함께 호출합니다.

결과 객체에 [DescribeRegionsResponse](#)가 표시됩니다. GetRegions 함수를 호출하면 각 리전을 나타내는 [Region](#) 객체 목록을 확인할 수 있습니다.

## 포함 파일

```
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/DescribeRegionsRequest.h>
```

## 코드

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);

Aws::EC2::Model::DescribeRegionsRequest request;
Aws::EC2::Model::DescribeRegionsOutcome outcome =
ec2Client.DescribeRegions(request);
if (outcome.IsSuccess()) {
    std::cout << std::left <<
        std::setw(32) << "RegionName" <<
        std::setw(64) << "Endpoint" << std::endl;

    const auto &regions = outcome.GetResult().GetRegions();
    for (const auto &region: regions) {
        std::cout << std::left <<
            std::setw(32) << region.GetRegionName() <<
            std::setw(64) << region.GetEndpoint() << std::endl;
    }
} else {
    std::cerr << "Failed to describe regions:" <<
        outcome.GetError().GetMessage() << std::endl;
}
```

[전체 예제](#)를 참조하세요.

## 가용 영역 설명

계정에서 사용 가능한 가용 영역을 나열하려면 EC2Client의 DescribeAvailabilityZones 함수를 [DescribeAvailabilityZonesRequest](#)와 함께 호출합니다.

결과 객체에 [DescribeAvailabilityZonesResponse](#)가 표시됩니다. GetAvailabilityZones 함수를 호출하면 각 가용 영역을 나타내는 [AvailabilityZone](#) 객체 목록을 확인할 수 있습니다.

## 포함 파일

```
#include <aws/ec2/model/DescribeAvailabilityZonesRequest.h>
```

## 코드

```
Aws::EC2::Model::DescribeAvailabilityZonesRequest request;
Aws::EC2::Model::DescribeAvailabilityZonesOutcome outcome =
ec2Client.DescribeAvailabilityZones(request);

if (outcome.IsSuccess()) {
    std::cout << std::left <<
        std::setw(32) << "ZoneName" <<
        std::setw(20) << "State" <<
        std::setw(32) << "Region" << std::endl;

    const auto &zones =
        outcome.GetResult().GetAvailabilityZones();

    for (const auto &zone: zones) {
        Aws::String stateString =
            Aws::EC2::Model::AvailabilityZoneStateMapper::GetNameForAvailabilityZoneState(
                zone.GetState());
        std::cout << std::left <<
            std::setw(32) << zone.GetZoneName() <<
            std::setw(20) << stateString <<
            std::setw(32) << zone.GetRegionName() << std::endl;
    }
} else {
    std::cerr << "Failed to describe availability zones:" <<
        outcome.GetError().GetMessage() << std::endl;
}
```

[전체 예제](#)를 참조하세요.

## 추가 정보

- Amazon EC2 사용 설명서의 [리전 및 가용 영역](#)
- Amazon EC2 API 참조의 [DescribeRegions](#)
- Amazon EC2 API 참조의 [DescribeAvailabilityZones](#)

## Amazon EC2 키 페어로 작업

### 사전 조건

시작하기 전에 [AWS SDK for C++ 사용 시작하기](#)를 읽어보시기 바랍니다.

예제 코드를 다운로드하고 [코드 예제 시작하기](#)에 설명된 대로 솔루션을 빌드합니다.

예제를 실행하려면 코드가 요청을 수행하는 데 사용하는 사용자 프로필이 AWS에서 적절한 권한(서비스 및 동작에 대한)을 보유하고 있어야 합니다. 자세한 내용은 [AWS 자격 증명 제공](#)을 참조하세요.

### 키 페어 생성

키 페어를 생성하려면 EC2Client의 CreateKeyPair 함수를 키 이름이 포함된 [CreateKeyPairRequest](#)와 함께 호출합니다.

### 포함 파일

```
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/CreateKeyPairRequest.h>
#include <iostream>
#include <fstream>
```

### 코드

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::CreateKeyPairRequest request;
request.SetKeyName(keyPairName);

Aws::EC2::Model::CreateKeyPairOutcome outcome = ec2Client.CreateKeyPair(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Failed to create key pair - " << keyPairName << ". " <<
        outcome.GetError().GetMessage() << std::endl;
```

```

    } else {
        std::cout << "Successfully created key pair named " <<
            keyPairName << std::endl;
        if (!keyFilePath.empty()) {
            std::ofstream keyFile(keyFilePath.c_str());
            keyFile << outcome.GetResult().GetKeyMaterial();
            keyFile.close();
            std::cout << "Keys written to the file " <<
                keyFilePath << std::endl;
        }
    }
}

```

[전체 예제](#)를 참조하세요.

## 키 페어 설명

키 페어를 나열하거나 키 페어에 대한 정보를 가져오려면 EC2Client의 DescribeKeyPairs 함수를 [DescribeKeyPairsRequest](#)와 함께 호출합니다.

[DescribeKeyPairsResponse](#)를 수신하게 되며 이 객체의 GetKeyPairs 함수를 호출하여 키 페어 목록에 액세스할 수 있습니다. 그러면 [KeyPairInfo](#) 객체 목록이 반환됩니다.

## 포함 파일

```

#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/DescribeKeyPairsRequest.h>
#include <aws/ec2/model/DescribeKeyPairsResponse.h>
#include <iomanip>
#include <iostream>

```

## 코드

```

Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::DescribeKeyPairsRequest request;

Aws::EC2::Model::DescribeKeyPairsOutcome outcome =
ec2Client.DescribeKeyPairs(request);
if (outcome.IsSuccess()) {
    std::cout << std::left <<
        std::setw(32) << "Name" <<
        std::setw(64) << "Fingerprint" << std::endl;
}

```

```

const std::vector<Aws::EC2::Model::KeyPairInfo> &key_pairs =
    outcome.GetResult().GetKeyPairs();
for (const auto &key_pair: key_pairs) {
    std::cout << std::left <<
        std::setw(32) << key_pair.GetKeyName() <<
        std::setw(64) << key_pair.GetKeyFingerprint() << std::endl;
}
} else {
    std::cerr << "Failed to describe key pairs:" <<
        outcome.GetError().GetMessage() << std::endl;
}

```

[전체 예제](#)를 참조하세요.

## 키 페어 삭제

키 페어를 삭제하려면 EC2Client의 DeleteKeyPair 함수를 호출하고 삭제할 키 페어 이름이 포함된 [DeleteKeyPairRequest](#)를 해당 함수에 전달합니다.

## 포함 파일

```

#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/DeleteKeyPairRequest.h>
#include <iostream>

```

## 코드

```

Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::DeleteKeyPairRequest request;

request.SetKeyName(keyPairName);
const Aws::EC2::Model::DeleteKeyPairOutcome outcome = ec2Client.DeleteKeyPair(
    request);

if (!outcome.IsSuccess()) {
    std::cerr << "Failed to delete key pair " << keyPairName <<
        ":" << outcome.GetError().GetMessage() << std::endl;
} else {
    std::cout << "Successfully deleted key pair named " << keyPairName <<
        std::endl;
}

```

[전체 예제](#)를 참조하세요.

## 추가 정보

- Amazon EC2 사용 설명서의 [Amazon EC2 키 페어](#)
- Amazon EC2 API 참조의 [CreateKeyPair](#)
- Amazon EC2 API 참조의 [DescribeKeyPairs](#)
- Amazon EC2 API 참조의 [DeleteKeyPair](#)

## Amazon EC2의 보안 그룹 작업

### 사전 조건

시작하기 전에 [AWS SDK for C++ 사용 시작하기](#)를 읽어보시기 바랍니다.

예제 코드를 다운로드하고 [코드 예제 시작하기](#)에 설명된 대로 솔루션을 빌드합니다.

예제를 실행하려면 코드가 요청을 수행하는 데 사용하는 사용자 프로필이 AWS에서 적절한 권한(서비스 및 동작에 대한)을 보유하고 있어야 합니다. 자세한 내용은 [AWS 자격 증명 제공](#)을 참조하세요.

### 보안 그룹 생성

보안 그룹을 생성하려면 EC2Client의 CreateSecurityGroup 함수를 키 이름이 포함된 [CreateSecurityGroupRequest](#)와 함께 호출합니다.

### 포함 파일

```
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/CreateSecurityGroupRequest.h>
```

### 코드

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);

Aws::EC2::Model::CreateSecurityGroupRequest request;

request.SetGroupName(groupName);
request.SetDescription(description);
request.SetVpcId(vpcID);
```

```
const Aws::EC2::Model::CreateSecurityGroupOutcome outcome =
    ec2Client.CreateSecurityGroup(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Failed to create security group:" <<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}

std::cout << "Successfully created security group named " << groupName <<
    std::endl;
```

[전체 예제](#)를 참조하세요.

## 보안 그룹 구성

보안 그룹은 Amazon EC2 인스턴스에 대한 인바운드(수신) 및 아웃바운드(송신) 트래픽을 모두 제어할 수 있습니다.

보안 그룹에 수신 규칙을 추가하려면 EC2Client의 `AuthorizeSecurityGroupIngress` 함수를 사용합니다. 이때 [AuthorizeSecurityGroupIngressRequest](#) 객체 내에서 보안 그룹의 이름과 보안 그룹에 할당하려는 액세스 규칙([IpPermission](#))을 제공해야 합니다. 다음 예제에서는 IP 권한을 보안 그룹에 추가하는 방법을 보여줍니다.

## 포함 파일

```
#include <aws/ec2/model/AuthorizeSecurityGroupIngressRequest.h>
```

## 코드

```
Aws::EC2::Model::AuthorizeSecurityGroupIngressRequest
authorizeSecurityGroupIngressRequest;
authorizeSecurityGroupIngressRequest.SetGroupId(groupId);
```

```
Aws::String ingressIPRange = "203.0.113.0/24"; // Configure this for your allowed
IP range.
Aws::EC2::Model::IpRange ip_range;
ip_range.SetCidrIp(ingressIPRange);

Aws::EC2::Model::IpPermission permission1;
permission1.SetIpProtocol("tcp");
```

```

permission1.SetToPort(80);
permission1.SetFromPort(80);
permission1.AddIpRanges(ip_range);

authorize_request.AddIpPermissions(permission1);

Aws::EC2::Model::IpPermission permission2;
permission2.SetIpProtocol("tcp");
permission2.SetToPort(22);
permission2.SetFromPort(22);
permission2.AddIpRanges(ip_range);

authorize_request.AddIpPermissions(permission2);

```

```

Aws::EC2::Model::AuthorizeSecurityGroupIngressOutcome
authorizeSecurityGroupIngressOutcome =

ec2Client.AuthorizeSecurityGroupIngress(authorizeSecurityGroupIngressRequest);

if (authorizeSecurityGroupIngressOutcome.IsSuccess()) {
    std::cout << "Successfully authorized security group ingress." << std::endl;
} else {
    std::cerr << "Error authorizing security group ingress: "
               << authorizeSecurityGroupIngressOutcome.GetError().GetMessage() <<
std::endl;
}

```

보안 그룹에 송신 규칙을 추가하려면 [AuthorizeSecurityGroupEgressRequest](#)의 유사한 데이터를 EC2Client의 AuthorizeSecurityGroupEgress 함수에 제공합니다.

[전체 예제](#)를 참조하세요.

## 보안 그룹 설명

보안 그룹을 설명하거나 보안 그룹에 대한 정보를 가져오려면 EC2Client의 DescribeSecurityGroups 함수를 [DescribeSecurityGroupsRequest](#)와 함께 호출합니다.

결과 객체에서 [DescribeSecurityGroupsResponse](#)를 수신하게 되며, 이 객체의 GetSecurityGroups 함수를 호출하여 보안 그룹 목록에 액세스할 수 있습니다. 그러면 [SecurityGroup](#) 객체 목록이 반환됩니다.

## 포함 파일

```
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/DescribeSecurityGroupsRequest.h>
#include <aws/ec2/model/DescribeSecurityGroupsResponse.h>
#include <iomanip>
#include <iostream>
```

## 코드

```
Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::DescribeSecurityGroupsRequest request;

if (!groupID.empty()) {
    request.AddGroupIds(groupID);
}

Aws::String nextToken;
do {
    if (!nextToken.empty()) {
        request.SetNextToken(nextToken);
    }

    Aws::EC2::Model::DescribeSecurityGroupsOutcome outcome =
ec2Client.DescribeSecurityGroups(request);
    if (outcome.IsSuccess()) {
        std::cout << std::left <<
            std::setw(32) << "Name" <<
            std::setw(30) << "GroupId" <<
            std::setw(30) << "VpcId" <<
            std::setw(64) << "Description" << std::endl;

        const std::vector<Aws::EC2::Model::SecurityGroup> &securityGroups =
            outcome.GetResult().GetSecurityGroups();

        for (const auto &securityGroup: securityGroups) {
            std::cout << std::left <<
                std::setw(32) << securityGroup.GetGroupName() <<
                std::setw(30) << securityGroup.GetGroupId() <<
                std::setw(30) << securityGroup.GetVpcId() <<
                std::setw(64) << securityGroup.GetDescription() <<
                std::endl;
        }
    } else {
        std::cerr << "Failed to describe security groups:" <<
```

```

        outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    nextToken = outcome.GetResult().GetNextToken();
} while (!nextToken.empty());

```

[전체 예제](#)를 참조하세요.

## 보안 그룹 삭제

보안 그룹을 삭제하려면 EC2Client의 DeleteSecurityGroup 함수를 호출합니다. 이때 삭제할 보안 그룹의 ID가 포함된 [DeleteSecurityGroupRequest](#)를 전달합니다.

## 포함 파일

```

#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/DeleteSecurityGroupRequest.h>
#include <iostream>

```

## 코드

```

Aws::EC2::EC2Client ec2Client(clientConfiguration);
Aws::EC2::Model::DeleteSecurityGroupRequest request;

request.SetGroupId(securityGroupID);
Aws::EC2::Model::DeleteSecurityGroupOutcome outcome =
ec2Client.DeleteSecurityGroup(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Failed to delete security group " << securityGroupID <<
        ":" << outcome.GetError().GetMessage() << std::endl;
} else {
    std::cout << "Successfully deleted security group " << securityGroupID <<
        std::endl;
}

```

[전체 예제](#)를 참조하세요.

## 추가 정보

- Amazon EC2 사용 설명서의 [Amazon EC2 보안 그룹](#)

- Amazon EC2 사용 설명서의 [Linux 인스턴스의 인바운드 트래픽 권한 부여](#)
- Amazon EC2 API 참조의 [CreateSecurityGroup](#)
- Amazon EC2 API 참조의 [DescribeSecurityGroups](#)
- Amazon EC2 API 참조의 [DeleteSecurityGroup](#)
- Amazon EC2 API 참조의 [AuthorizeSecurityGroupIngress](#)

## AWS SDK for C++를 사용한 Amazon S3용 코드 예제

[Amazon S3](#)는 어디서든 원하는 양의 데이터를 저장하고 검색할 수 있도록 빌드된 객체 스토리지입니다. AWS SDK for C++는 Amazon S3와 연동하기 위해 여러 클래스를 제공합니다.

### Note

이 안내서에는 특정 기술을 시연하는 데 필요한 코드만 제공되며 [전체 예제 코드는 GitHub에서 확인할 수 있습니다](#). GitHub에서 단일 소스 파일을 다운로드하거나, 리포지토리를 로컬에 복제하여 모든 예제를 가져오고 빌드하고 실행할 수 있습니다.

- [S3Client](#) 클래스

S3Client 라이브러리는 완전한 기능을 갖춘 Amazon S3 인터페이스입니다.

이 세트의 `list_buckets_disabling_dns_cache.cpp` 예제는 Linux/Mac 환경에서 CURL과 함께 작동하도록 특별히 설계되었습니다(Windows에서도 작동하도록 수정 가능). Windows에서 작업하는 경우, 프로젝트를 빌드하기 전에 `list_buckets_disabling_dns_cache.cpp` 파일을 삭제합니다. 이 파일은 Linux의 `curl` HttpClient에 의존하기 때문입니다.

S3Client를 활용하는 예제 코드는 Github의 [s3 폴더](#)에 있습니다. 이 예제 세트로 시연된 기능의 전체 목록은 Github의 [Readme](#)를 참조하세요.

s3 예제 세트의 일부는 본 안내서에서 추가적으로 상세히 다루고 있습니다.

- [버킷 생성, 나열, 삭제](#)
- [객체 관련 작업](#) – 데이터 객체 업로드 및 다운로드
- [Amazon S3 액세스 권한 관리](#)
- [버킷 정책을 사용하여 Amazon S3 버킷에 대한 액세스 관리](#)
- [Amazon S3 버킷을 웹 사이트로 구성](#)

- [S3CrtClient](#) 클래스

S3CrtClient는 SDK 버전 1.9에서 추가되었습니다. S3CrtClient는 Amazon S3 GET(다운로드) 및 PUT(업로드) 작업에 높은 처리량을 제공합니다. S3CrtClient는 AWS 공통 런타임(CRT) 라이브러리 위에 구현됩니다.

S3CrtClient를 활용하는 예제 코드는 Github의 [s3-crt 폴더](#)에 있습니다. 이 예제 세트로 시연된 기능의 전체 목록은 Github의 [Readme](#)를 참조하세요.

- [Amazon S3 작업에 S3CrtClient 사용](#)

- [TransferManager](#) 클래스

TransferManager는 File Transfer Protocol(FTP), File Transfer Protocol over SSL(FTPS) 또는 Secure Shell(SSH) File Transfer Protocol(SFTP)을 통해 파일을 Amazon S3와 직접 주고받도록 지원하는 완전 관리형 서비스입니다.

TransferManager를 활용하는 예제 코드는 Github의 [transfer-manager 폴더](#)에 있습니다. 이 예제 세트로 시연된 기능의 전체 목록은 Github의 [Readme](#)를 참조하세요.

- [Amazon S3 작업에 TransferManager 사용](#)

## 버킷 생성, 나열, 삭제

Amazon Simple Storage Service(Amazon S3)의 모든 객체 또는 파일은 객체 폴더를 나타내는 버킷에 포함됩니다. 각 버킷에는 AWS 내에서 전역적으로 고유한 이름이 있습니다. 자세한 내용은 Amazon Simple Storage Service 사용 설명서에서 [Amazon S3 버킷을 사용한 작업](#)을 참조하세요.

### 사전 조건

시작하기 전에 [AWS SDK for C++ 사용 시작하기](#)를 읽어보시기 바랍니다.

예제 코드를 다운로드하고 [코드 예제 시작하기](#)에 설명된 대로 솔루션을 빌드합니다.

예제를 실행하려면 코드가 요청을 수행하는 데 사용하는 사용자 프로필이 AWS에서 적절한 권한(서비스 및 동작에 대한)을 보유하고 있어야 합니다. 자세한 내용은 [AWS 자격 증명 제공](#)을 참조하세요.

### 버킷 나열

list\_buckets 예제를 실행하려면 명령 프롬프트에서 빌드 시스템이 빌드 실행 파일을 생성하는 폴더로 이동합니다. run\_list\_buckets과 같은 실행 파일을 실행합니다. 전체 실행 파일 이름은 운영 체제에 따라 다릅니다. 출력에는 계정 버킷이 있는 경우 해당 버킷 목록이 표시되며, 버킷이 없는 경우에는 빈 목록이 표시됩니다.

`list_buckets.cpp`에는 두 가지 메서드가 있습니다.

- `main()` - `ListBuckets()`를 호출합니다.
- `ListBuckets()` - SDK를 사용하여 버킷을 쿼리합니다.

`S3Client` 객체는 SDK의 `ListBuckets()` 메서드를 호출합니다. 성공하면 메서드는 `ListBucketResult` 객체가 포함된 `ListBucketOutcome` 객체를 반환합니다.

`ListBucketResult` 객체는 `GetBuckets()` 메서드를 호출하여 계정의 각 Amazon S3 버킷에 대한 정보가 포함된 `Bucket` 객체 목록을 가져옵니다.

## 코드

```
bool AwsDoc::S3::listBuckets(const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    auto outcome = client.ListBuckets();

    bool result = true;
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed with error: " << outcome.GetError() << std::endl;
        result = false;
    } else {
        std::cout << "Found " << outcome.GetResult().GetBuckets().size() << " buckets
\n";
        for (auto &b: outcome.GetResult().GetBuckets()) {
            std::cout << b.GetName() << std::endl;
        }
    }

    return result;
}
```

전체 [list\\_buckets 예제](#)는 Github에서 확인하세요.

## 버킷 생성

`create_bucket` 예제를 실행하려면 명령 프롬프트에서 빌드 시스템이 빌드 실행 파일을 생성하는 폴더로 이동합니다. `run_create_bucket`과 같은 실행 파일을 실행합니다. 전체 실행 파일 이름은 운영 체제에 따라 다릅니다. 이 코드는 계정 아래에 빈 버킷을 생성한 다음 요청의 성공 또는 실패를 표시합니다.

create\_bucket.cpp에는 두 가지 메서드가 있습니다.

- main() - CreateBucket()를 호출합니다. main()에서는 enum를 사용하여 AWS 리전을 계정의 리전으로 변경해야 합니다. [AWS Management 콘솔](#)에 로그인한 후 오른쪽 상단 모서리에서 리전을 찾으면 본인 계정의 리전을 확인할 수 있습니다.
- CreateBucket() - SDK를 사용하여 버킷을 생성합니다.

S3Client 객체는 SDK의 CreateBucket() 메서드를 호출하여 버킷 이름과 함께 CreateBucketRequest를 전달합니다. 기본적으로 버킷은 us-east-1(북부 버지니아) 리전에 생성됩니다. 리전이 us-east-1이 아닌 경우 코드는 버킷이 해당 리전에 생성되도록 보장하기 위해 버킷 제약 조건을 설정합니다.

## 코드

```
bool AwsDoc::S3::createBucket(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CreateBucketRequest request;
    request.SetBucket(bucketName);

    if (clientConfig.region != "us-east-1") {
        Aws::S3::Model::CreateBucketConfiguration createBucketConfig;
        createBucketConfig.SetLocationConstraint(
            Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                clientConfig.region));
        request.SetCreateBucketConfiguration(createBucketConfig);
    }

    Aws::S3::Model::CreateBucketOutcome outcome = client.CreateBucket(request);
    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: createBucket: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
        std::cout << "Created bucket " << bucketName <<
            " in the specified AWS Region." << std::endl;
    }

    return outcome.IsSuccess();
}
```

```
}
```

전체 [create\\_buckets 예제](#)는 Github에서 확인하세요.

## 버킷 삭제

delete\_bucket 예제를 실행하려면 명령 프롬프트에서 빌드 시스템이 빌드 실행 파일을 생성하는 폴더로 이동합니다. run\_delete\_bucket과 같은 실행 파일을 실행합니다. 전체 실행 파일 이름은 운영 체제에 따라 다릅니다. 코드는 계정에 지정된 버킷을 삭제한 다음 요청의 성공 또는 실패를 표시합니다.

delete\_bucket.cpp에는 두 가지 메서드가 있습니다.

- main() - DeleteBucket()를 호출합니다. main()에서는 enum를 사용하여 AWS 리전을 계정의 리전으로 변경해야 합니다. 또한 bucket\_name을 삭제할 버킷의 이름으로 변경해야 합니다.
- DeleteBucket() - SDK를 사용하여 버킷을 삭제합니다.

S3Client 객체는 SDK의 DeleteBucket() 메서드를 사용하여 삭제할 버킷 이름과 함께 DeleteBucketRequest 객체를 전달합니다. 성공하려면 버킷이 비어 있어야 합니다.

## 코드

```
bool AwsDoc::S3::deleteBucket(const Aws::String &bucketName,
                               const Aws::S3::S3ClientConfiguration &clientConfig) {

    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
        client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucket: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
        std::cout << "The bucket was deleted" << std::endl;
    }
}
```

```
    return outcome.IsSuccess();
}
```

전체 [delete\\_bucket 예제](#)는 Github에서 확인하세요.

## 객체 관련 작업

Amazon S3 객체는 데이터 모음인 파일을 나타냅니다. 각 객체는 [버킷](#) 안에 상주해야 합니다.

### 사전 조건

시작하기 전에 [AWS SDK for C++ 사용 시작하기](#)를 읽어보시기 바랍니다.

예제 코드를 다운로드하고 [코드 예제 시작하기](#)에 설명된 대로 솔루션을 빌드합니다.

예제를 실행하려면 코드가 요청을 수행하는 데 사용하는 사용자 프로필이 AWS에서 적절한 권한(서비스 및 동작에 대한)을 보유하고 있어야 합니다. 자세한 내용은 [AWS 자격 증명 제공](#)을 참조하세요.

### 버킷에 파일 업로드

S3Client 객체의 PutObject 함수를 사용하여 버킷 이름, 키 이름 및 업로드할 파일을 제공합니다. Aws::FStream은 로컬 파일의 콘텐츠를 버킷에 업로드하는 데 사용됩니다. 버킷이 반드시 있어야 하며, 그렇지 않으면 오류가 발생합니다.

객체를 비동기적으로 업로드하는 예제는 [AWS SDK for C++를 사용한 비동기 프로그래밍](#) 섹션을 참조하세요.

### 코드

```
bool AwsDoc::S3::putObject(const Aws::String &bucketName,
                           const Aws::String &fileName,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    //We are using the name of the file as the key for the object in the bucket.
    //However, this is just a string and can be set according to your retrieval needs.
    request.SetKey(fileName);

    std::shared_ptr<Aws::IOStream> inputData =
        Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                       fileName.c_str(),
```

```

std::ios_base::in | std::ios_base::binary);

if (!*inputData) {
    std::cerr << "Error unable to read file " << fileName << std::endl;
    return false;
}

request.SetBody(inputData);

Aws::S3::Model::PutObjectOutcome outcome =
    s3Client.PutObject(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Error: putObject: " <<
        outcome.GetError().GetMessage() << std::endl;
} else {
    std::cout << "Added object '" << fileName << "' to bucket '"
        << bucketName << "'.";
}

return outcome.IsSuccess();
}

```

[전체 예제](#)는 Github에서 확인하세요.

## 버킷에 문자열 업로드

S3Client 객체의 PutObject 함수를 사용하여 버킷 이름, 키 이름 및 업로드할 파일을 제공합니다. 버킷이 반드시 있어야 하며, 그렇지 않으면 오류가 발생합니다. 이 예제는 Aws::StringStream을 사용하여 인 메모리 문자열 데이터 객체를 버킷에 직접 업로드한다는 점에서 이전 예제와 다릅니다.

객체를 비동기적으로 업로드하는 예제는 [AWS SDK for C++를 사용한 비동기 프로그래밍](#) 섹션을 참조하세요.

## 코드

```

bool AwsDoc::S3::putObjectBuffer(const Aws::String &bucketName,
                                const Aws::String &objectName,
                                const std::string &objectContent,
                                const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::PutObjectRequest request;

```

```

request.SetBucket(bucketName);
request.SetKey(objectName);

const std::shared_ptr<Aws::IOStream> inputData =
    Aws::MakeShared<Aws::StringStream>("");
*inputData << objectContent.c_str();

request.SetBody(inputData);

Aws::S3::Model::PutObjectOutcome outcome = s3Client.PutObject(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Error: putObjectBuffer: " <<
        outcome.GetError().GetMessage() << std::endl;
} else {
    std::cout << "Success: Object '" << objectName << "' with content '"
        << objectContent << "' uploaded to bucket '" << bucketName << "'.";
}

return outcome.IsSuccess();
}

```

[전체 예제](#)는 Github에서 확인하세요.

## 객체 나열

버킷 내의 객체 목록을 가져오려면 S3Client 객체의 ListObjects 함수를 사용합니다. 콘텐츠를 나열할 버킷의 이름이 설정된 ListObjectsRequest를 이 함수에 제공합니다.

ListObjects 함수는 객체 목록을 Object 인스턴스 형태로 가져오는 데 사용할 수 있는 ListObjectsOutcome 객체를 반환합니다.

## 코드

```

bool AwsDoc::S3::listObjects(const Aws::String &bucketName,
                             Aws::Vector<Aws::String> &keysResult,
                             const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::ListObjectsV2Request request;
    request.WithBucket(bucketName);

    Aws::String continuationToken; // Used for pagination.

```

```

    Aws::Vector<Aws::S3::Model::Object> allObjects;

    do {
        if (!continuationToken.empty()) {
            request.SetContinuationToken(continuationToken);
        }

        auto outcome = s3Client.ListObjectsV2(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error: listObjects: " <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        } else {
            Aws::Vector<Aws::S3::Model::Object> objects =
                outcome.GetResult().GetContents();

            allObjects.insert(allObjects.end(), objects.begin(), objects.end());
            continuationToken = outcome.GetResult().GetNextContinuationToken();
        }
    } while (!continuationToken.empty());

    std::cout << allObjects.size() << " object(s) found:" << std::endl;

    for (const auto &object: allObjects) {
        std::cout << " " << object.GetKey() << std::endl;
        keysResult.push_back(object.GetKey());
    }

    return true;
}

```

[전체 예제](#)는 Github에서 확인하세요.

## 객체 다운로드

S3Client 객체의 GetObject 함수를 사용하여 버킷 이름과 다운로드할 객체 키가 설정된 GetObjectRequest를 이 함수에 전달합니다. GetObject는 [GetObjectResult](#) 및 [S3Error](#)로 구성된 [GetObjectOutcome](#) 객체를 반환합니다. GetObjectResult는 S3 객체의 데이터에 액세스하는 데 사용할 수 있습니다.

다음 예제는 Amazon S3에서 객체를 다운로드합니다. 객체 콘텐츠는 로컬 변수에 저장되고 콘텐츠의 첫 번째 줄이 콘솔에 출력됩니다.

## 코드

```
bool AwsDoc::S3::getObject(const Aws::String &objectKey,
                           const Aws::String &fromBucket,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(fromBucket);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectOutcome outcome =
        client.GetObject(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObject: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully retrieved '" << objectKey << "' from '"
            << fromBucket << "'." << std::endl;
    }

    return outcome.IsSuccess();
}
```

[전체 예제](#)는 Github에서 확인하세요.

## 객체 삭제

S3Client 객체의 DeleteObject 함수를 사용하여 다운로드할 버킷과 객체의 이름이 설정된 DeleteObjectRequest를 이 함수에 전달합니다. 지정된 버킷과 객체 키가 반드시 있어야 하며, 그렇지 않으면 오류가 발생합니다.

## 코드

```
bool AwsDoc::S3::deleteObject(const Aws::String &objectKey,
                               const Aws::String &fromBucket,
                               const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectRequest request;

    request.WithKey(objectKey)
```

```

        .WithBucket(fromBucket);

    Aws::S3::Model::DeleteObjectOutcome outcome =
        client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: deleteObject: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully deleted the object." << std::endl;
    }

    return outcome.IsSuccess();
}

```

[전체 예제](#)는 Github에서 확인하세요.

## Amazon S3 액세스 권한 관리

Amazon S3 버킷 또는 객체에 대한 액세스 권한은 액세스 제어 목록(ACL)에 정의되어 있습니다. ACL은 버킷/객체의 소유자와 권한 부여 목록을 지정합니다. 각 권한 부여에는 사용자(또는 수혜자)와 버킷/객체에 대한 사용자의 액세스 권한(예: READ 또는 WRITE 권한)이 지정됩니다.

### 사전 조건

시작하기 전에 [AWS SDK for C++ 사용 시작하기](#)를 읽어보시기 바랍니다.

예제 코드를 다운로드하고 [코드 예제 시작하기](#)에 설명된 대로 솔루션을 빌드합니다.

예제를 실행하려면 코드가 요청을 수행하는 데 사용하는 사용자 프로필이 AWS에서 적절한 권한(서비스 및 동작에 대한)을 보유하고 있어야 합니다. 자세한 내용은 [AWS 자격 증명 제공](#)을 참조하세요.

### 객체의 액세스 제어 목록 관리

S3Client 메서드 GetObjectAcl를 호출하여 객체에 대한 액세스 제어 목록을 검색할 수 있습니다. 이 메서드는 객체와 버킷의 이름을 허용합니다. 반환 값에는 ACL의 Owner 및 Grants 목록이 포함됩니다.

```

bool AwsDoc::S3::getObjectAcl(const Aws::String &bucketName,
                              const Aws::String &objectKey,
                              const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

```

```

Aws::S3::Model::GetObjectAclRequest request;
request.SetBucket(bucketName);
request.SetKey(objectKey);

Aws::S3::Model::GetObjectAclOutcome outcome =
    s3Client.GetObjectAcl(request);

if (!outcome.IsSuccess()) {
    const Aws::S3::S3Error &err = outcome.GetError();
    std::cerr << "Error: getObjectAcl: "
                << err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
} else {
    Aws::Vector<Aws::S3::Model::Grant> grants =
        outcome.GetResult().GetGrants();

    for (auto it = grants.begin(); it != grants.end(); it++) {
        std::cout << "For object " << objectKey << ": "
                    << std::endl << std::endl;

        Aws::S3::Model::Grant grant = *it;
        Aws::S3::Model::Grantee grantee = grant.GetGrantee();

        if (grantee.TypeHasBeenSet()) {
            std::cout << "Type: "
                        << getGranteeTypeString(grantee.GetType()) << std::endl;
        }

        if (grantee.DisplayNameHasBeenSet()) {
            std::cout << "Display name: "
                        << grantee.GetDisplayName() << std::endl;
        }

        if (grantee.EmailAddressHasBeenSet()) {
            std::cout << "Email address: "
                        << grantee.GetEmailAddress() << std::endl;
        }

        if (grantee.IDHasBeenSet()) {
            std::cout << "ID: "
                        << grantee.GetID() << std::endl;
        }

        if (grantee.URIHasBeenSet()) {

```

```

        std::cout << "URI:          "
                    << grantee.GetURI() << std::endl;
    }

    std::cout << "Permission:    " <<
                getPermissionString(grant.GetPermission()) <<
                std::endl << std::endl;
    }
}

return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable string.
/*!
 \param type: Type enumeration.
 \return String: Human-readable string
 */
Aws::String getGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:
            return "Predefined Amazon S3 group";
        case Aws::S3::Model::Type::NOT_SET:
            return "Not set";
        default:
            return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable string.
/*!
 \param permission: Permission enumeration.
 \return String: Human-readable string
 */
Aws::String getPermissionString(const Aws::S3::Model::Permission &permission) {
    switch (permission) {
        case Aws::S3::Model::Permission::FULL_CONTROL:
            return "Can read this object's data and its metadata, "
                  "and read/write this object's permissions";
        case Aws::S3::Model::Permission::NOT_SET:

```

```

        return "Permission not set";
    case Aws::S3::Model::Permission::READ:
        return "Can read this object's data and its metadata";
    case Aws::S3::Model::Permission::READ_ACP:
        return "Can read this object's permissions";
        // case Aws::S3::Model::Permission::WRITE // Not applicable.
    case Aws::S3::Model::Permission::WRITE_ACP:
        return "Can write this object's permissions";
    default:
        return "Permission unknown";
    }
}

```

ACL은 새 ACL을 생성하거나 현재 ACL에 지정된 권한 부여를 변경하여 수정할 수 있습니다. 업데이트된 ACL은 PutObjectAcl 메서드에 전달하여 새로운 현재 ACL이 됩니다.

다음 코드는 GetObjectAcl로 검색된 ACL을 사용하고 여기에 새 권한 부여를 추가합니다. 사용자 또는 피부여자에게 객체에 대한 READ 권한이 부여됩니다. 수정된 ACL은 PutObjectAcl로 전달되어 새로운 현재 ACL이 됩니다.

```

bool AwsDoc::S3::putObjectAcl(const Aws::String &bucketName, const Aws::String
    &objectKey, const Aws::String &ownerID,
                                const Aws::String &granteePermission, const Aws::String
    &granteeType,
                                const Aws::String &granteeID, const Aws::String
    &granteeEmailAddress,
                                const Aws::String &granteeURI, const
    Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::Owner owner;
    owner.SetID(ownerID);

    Aws::S3::Model::Grantee grantee;
    grantee.SetType(setGranteeType(granteeType));

    if (!granteeEmailAddress.empty()) {
        grantee.SetEmailAddress(granteeEmailAddress);
    }

    if (!granteeID.empty()) {
        grantee.SetID(granteeID);
    }
}

```

```

    }

    if (!granteeURI.empty()) {
        grantee.SetURI(granteeURI);
    }

    Aws::S3::Model::Grant grant;
    grant.SetGrantee(grantee);
    grant.SetPermission(setGranteePermission(granteePermission));

    Aws::Vector<Aws::S3::Model::Grant> grants;
    grants.push_back(grant);

    Aws::S3::Model::AccessControlPolicy acp;
    acp.SetOwner(owner);
    acp.SetGrants(grants);

    Aws::S3::Model::PutObjectAclRequest request;
    request.SetAccessControlPolicy(acp);
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::PutObjectAclOutcome outcome =
        s3Client.PutObjectAcl(request);

    if (!outcome.IsSuccess()) {
        auto error = outcome.GetError();
        std::cerr << "Error: putObjectAcl: " << error.GetExceptionName()
            << " - " << error.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully added an ACL to the object '" << objectKey
            << "' in the bucket '" << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which converts a human-readable string to a built-in type enumeration.
/*!
    \param access: Human readable string.
    \return Permission: Permission enumeration.
*/
Aws::S3::Model::Permission setGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")

```

```

        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a human-readable string to a built-in type enumeration.
/*!
 \param type: Human readable string.
 \return Type: Type enumeration.
 */
Aws::S3::Model::Type setGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")
        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")
        return Aws::S3::Model::Type::CanonicalUser;
    if (type == "Group")
        return Aws::S3::Model::Type::Group;
    return Aws::S3::Model::Type::NOT_SET;
}

```

[전체 예제](#)는 Github에서 확인하세요.

## 버킷의 액세스 제어 목록 관리

대부분의 경우 버킷의 액세스 권한을 설정할 때 선호하는 방법은 버킷 정책을 정의하는 것입니다. 그러나 버킷은 액세스 제어 목록을 사용하고자 하는 사용자를 위해 이를 지원하기도 합니다.

버킷에 대한 액세스 제어 목록 관리는 객체에 사용되는 것과 동일합니다. GetBucketAcl 메서드는 버킷의 현재 ACL을 검색하고 PutBucketAcl 메서드는 버킷에 새 ACL을 적용합니다.

다음 코드는 버킷 ACL을 가져오고 설정하는 방법을 보여줍니다.

```

//! Routine which demonstrates setting the ACL for an S3 bucket.
/*!
 \param bucketName: Name of a bucket.

```

```

\param ownerID: The canonical ID of the bucket owner.
See https://docs.aws.amazon.com/AmazonS3/latest/userguide/finding-canonical-user-id.html for more information.
\param granteePermission: The access level to enable for the grantee.
\param granteeType: The type of grantee.
\param granteeID: The canonical ID of the grantee.
\param granteeEmailAddress: The email address associated with the grantee's AWS account.
\param granteeURI: The URI of a built-in access group.
\param clientConfig: Aws client configuration.
\return bool: Function succeeded.
*/

bool AwsDoc::S3::getPutBucketAcl(const Aws::String &bucketName,
                                const Aws::String &ownerID,
                                const Aws::String &granteePermission,
                                const Aws::String &granteeType,
                                const Aws::String &granteeID,
                                const Aws::String &granteeEmailAddress,
                                const Aws::String &granteeURI,
                                const Aws::S3::S3ClientConfiguration &clientConfig) {
    bool result = ::putBucketAcl(bucketName, ownerID, granteePermission, granteeType,
                                granteeID,
                                granteeEmailAddress,
                                granteeURI,
                                clientConfig);

    if (result) {
        result = ::getBucketAcl(bucketName, clientConfig);
    }

    return result;
}

//! Routine which demonstrates setting the ACL for an S3 bucket.
/*!
\param bucketName: Name of from bucket.
\param ownerID: The canonical ID of the bucket owner.
See https://docs.aws.amazon.com/AmazonS3/latest/userguide/finding-canonical-user-id.html for more information.
\param granteePermission: The access level to enable for the grantee.
\param granteeType: The type of grantee.
\param granteeID: The canonical ID of the grantee.
\param granteeEmailAddress: The email address associated with the grantee's AWS account.

```

```

\param granteeURI: The URI of a built-in access group.
\param clientConfig: Aws client configuration.
\return bool: Function succeeded.
*/

bool putBucketAcl(const Aws::String &bucketName,
                 const Aws::String &ownerID,
                 const Aws::String &granteePermission,
                 const Aws::String &granteeType,
                 const Aws::String &granteeID,
                 const Aws::String &granteeEmailAddress,
                 const Aws::String &granteeURI,
                 const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::Owner owner;
    owner.SetID(ownerID);

    Aws::S3::Model::Grantee grantee;
    grantee.SetType(setGranteeType(granteeType));

    if (!granteeEmailAddress.empty()) {
        grantee.SetEmailAddress(granteeEmailAddress);
    }

    if (!granteeID.empty()) {
        grantee.SetID(granteeID);
    }

    if (!granteeURI.empty()) {
        grantee.SetURI(granteeURI);
    }

    Aws::S3::Model::Grant grant;
    grant.SetGrantee(grantee);
    grant.SetPermission(setGranteePermission(granteePermission));

    Aws::Vector<Aws::S3::Model::Grant> grants;
    grants.push_back(grant);

    Aws::S3::Model::AccessControlPolicy acp;
    acp.SetOwner(owner);
    acp.SetGrants(grants);

```

```

    Aws::S3::Model::PutBucketAclRequest request;
    request.SetAccessControlPolicy(acp);
    request.SetBucket(bucketName);

    Aws::S3::Model::PutBucketAclOutcome outcome =
        s3Client.PutBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &error = outcome.GetError();

        std::cerr << "Error: putBucketAcl: " << error.GetExceptionName()
            << " - " << error.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully added an ACL to the bucket '" << bucketName
            << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which demonstrates getting the ACL for an S3 bucket.
/*!
    \param bucketName: Name of the s3 bucket.
    \param clientConfig: Aws client configuration.
    \return bool: Function succeeded.
*/
bool getBucketAcl(const Aws::String &bucketName,
                  const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketAclRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketAclOutcome outcome =
        s3Client.GetBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getBucketAcl: "
            << err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
        const Aws::Vector<Aws::S3::Model::Grant> &grants =
            outcome.GetResult().GetGrants();
    }
}

```

```

    for (const Aws::S3::Model::Grant &grant: grants) {
        const Aws::S3::Model::Grantee &grantee = grant.GetGrantee();

        std::cout << "For bucket " << bucketName << ": "
                    << std::endl << std::endl;

        if (grantee.TypeHasBeenSet()) {
            std::cout << "Type:          "
                        << getGranteeTypeString(grantee.GetType()) << std::endl;
        }

        if (grantee.DisplayNameHasBeenSet()) {
            std::cout << "Display name:  "
                        << grantee.GetDisplayName() << std::endl;
        }

        if (grantee.EmailAddressHasBeenSet()) {
            std::cout << "Email address: "
                        << grantee.GetEmailAddress() << std::endl;
        }

        if (grantee.IDHasBeenSet()) {
            std::cout << "ID:           "
                        << grantee.GetID() << std::endl;
        }

        if (grantee.URIHasBeenSet()) {
            std::cout << "URI:          "
                        << grantee.GetURI() << std::endl;
        }

        std::cout << "Permission:    " <<
                    getPermissionString(grant.GetPermission()) <<
                    std::endl << std::endl;
    }
}

return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable string.
/*!
\param permission: Permission enumeration.
\return String: Human-readable string.

```

```

*/

Aws::String getPermissionString(const Aws::S3::Model::Permission &permission) {
    switch (permission) {
        case Aws::S3::Model::Permission::FULL_CONTROL:
            return "Can list objects in this bucket, create/overwrite/delete "
                "objects in this bucket, and read/write this "
                "bucket's permissions";
        case Aws::S3::Model::Permission::NOT_SET:
            return "Permission not set";
        case Aws::S3::Model::Permission::READ:
            return "Can list objects in this bucket";
        case Aws::S3::Model::Permission::READ_ACP:
            return "Can read this bucket's permissions";
        case Aws::S3::Model::Permission::WRITE:
            return "Can create, overwrite, and delete objects in this bucket";
        case Aws::S3::Model::Permission::WRITE_ACP:
            return "Can write this bucket's permissions";
        default:
            return "Permission unknown";
    }
}

//! Routine which converts a human-readable string to a built-in type enumeration
/*!
\param access: Human readable string.
\return Permission: Permission enumeration.
*/
Aws::S3::Model::Permission setGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a built-in type enumeration to a human-readable string.
/*!

```

```

\param type: Type enumeration.
\return bool: Human-readable string.
*/
Aws::String getGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:
            return "Predefined Amazon S3 group";
        case Aws::S3::Model::Type::NOT_SET:
            return "Not set";
        default:
            return "Type unknown";
    }
}

Aws::S3::Model::Type setGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")
        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")
        return Aws::S3::Model::Type::CanonicalUser;
    if (type == "Group")
        return Aws::S3::Model::Type::Group;
    return Aws::S3::Model::Type::NOT_SET;
}

```

[전체 예제](#)는 Github에서 확인하세요.

## 버킷 정책을 사용하여 Amazon S3 버킷에 대한 액세스 관리

버킷 정책을 설정하거나 가져오거나 삭제하여 Amazon S3 버킷에 대한 액세스를 관리할 수 있습니다.

### 사전 조건

시작하기 전에 [AWS SDK for C++ 사용 시작하기](#)를 읽어보시기 바랍니다.

예제 코드를 다운로드하고 [코드 예제 시작하기](#)에 설명된 대로 솔루션을 빌드합니다.

예제를 실행하려면 코드가 요청을 수행하는 데 사용하는 사용자 프로필이 AWS에서 적절한 권한(서비스 및 동작에 대한)을 보유하고 있어야 합니다. 자세한 내용은 [AWS 자격 증명 제공](#)을 참조하세요.

## 버킷 정책 설정

S3Client의 PutBucketPolicy 함수를 호출하고 [PutBucketPolicyRequest](#)에 버킷 이름과 정책의 JSON 표현을 제공하여 특정 S3 버킷에 대한 버킷 정책을 설정할 수 있습니다.

### 코드

```

//! Build a policy JSON string.
/*!
    \param userArn: Aws user Amazon Resource Name (ARN).
        For more information, see https://docs.aws.amazon.com/IAM/latest/UserGuide/
reference_identifiers.html#identifiers-arns.
    \param bucketName: Name of a bucket.
    \return String: Policy as JSON string.
*/

Aws::String getPolicyString(const Aws::String &userArn,
                           const Aws::String &bucketName) {
    return
        "{\n"
        "  \"Version\": \"2012-10-17\", \n"
        "  \"Statement\": [\n"
        "    {\n"
        "      \"Sid\": \"1\", \n"
        "      \"Effect\": \"Allow\", \n"
        "      \"Principal\": {\n"
        "        \"AWS\": \"\"
        + userArn +
        "\"\"\"\"      }, \n"
        "      \"Action\": [ \"s3:getObject\" ], \n"
        "      \"Resource\": [ \"arn:aws:s3::\"
        + bucketName +
        \"/*\" ] \n"
        "    } \n"
        "  ] \n"
        "}";
}

```

```

bool AwsDoc::S3::putBucketPolicy(const Aws::String &bucketName,
                                  const Aws::String &policyBody,
                                  const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);
}

```

```

std::shared_ptr<Aws::StringStream> request_body =
    Aws::MakeShared<Aws::StringStream>("");
*request_body << policyBody;

Aws::S3::Model::PutBucketPolicyRequest request;
request.SetBucket(bucketName);
request.SetBody(request_body);

Aws::S3::Model::PutBucketPolicyOutcome outcome =
    s3Client.PutBucketPolicy(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Error: putBucketPolicy: "
                << outcome.GetError().GetMessage() << std::endl;
} else {
    std::cout << "Set the following policy body for the bucket '" <<
                bucketName << "':" << std::endl << std::endl;
    std::cout << policyBody << std::endl;
}

return outcome.IsSuccess();
}

```

### Note

[Aws::Utils::Json::JsonValue](#) 유틸리티 클래스는 PutBucketPolicy에 전달할 유효한 JSON 객체를 구성하는 데 사용할 수 있습니다.

[전체 예제](#)는 Github에서 확인하세요.

## 버킷 정책 가져오기

Amazon S3 버킷에 대한 정책을 가져오려면 S3Client의 GetBucketPolicy 함수를 호출하고 버킷 이름을 [GetBucketPolicyRequest](#)에 전달합니다.

### 코드

```

bool AwsDoc::S3::getBucketPolicy(const Aws::String &bucketName,
                                const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

```

```

Aws::S3::Model::GetBucketPolicyRequest request;
request.SetBucket(bucketName);

Aws::S3::Model::GetBucketPolicyOutcome outcome =
    s3Client.GetBucketPolicy(request);

if (!outcome.IsSuccess()) {
    const Aws::S3::S3Error &err = outcome.GetError();
    std::cerr << "Error: getBucketPolicy: "
                << err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
} else {
    Aws::StringStream policy_stream;
    Aws::String line;

    outcome.GetResult().GetPolicy() >> line;
    policy_stream << line;

    std::cout << "Retrieve the policy for bucket '" << bucketName << "':\n\n" <<
                policy_stream.str() << std::endl;
}

return outcome.IsSuccess();
}

```

[전체 예제](#)는 Github에서 확인하세요.

## 버킷 정책 삭제

버킷 정책을 삭제하려면 S3Client의 DeleteBucketPolicy 함수를 호출하고 버킷 이름을 [DeleteBucketPolicyRequest](#)에 제공합니다.

## 코드

```

bool AwsDoc::S3::deleteBucketPolicy(const Aws::String &bucketName,
                                     const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::DeleteBucketPolicyRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketPolicyOutcome outcome =
        client.DeleteBucketPolicy(request);
}

```

```

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucketPolicy: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
        std::cout << "Policy was deleted from the bucket." << std::endl;
    }

    return outcome.IsSuccess();
}

```

이 함수는 버킷에 정책이 아직 존재하지 않더라도 성공합니다. 존재하지 않는 버킷 이름을 지정하거나 해당 버킷에 대한 액세스 권한이 없는 경우 `AmazonServiceException`이 발생합니다.

[전체 예제](#)는 Github에서 확인하세요.

### 추가 정보

- Amazon Simple Storage Service API 참조의 [PutBucketPolicy](#)
- Amazon Simple Storage Service API 참조의 [GetBucketPolicy](#)
- Amazon Simple Storage Service API 참조의 [DeleteBucketPolicy](#)
- Amazon Simple Storage Service 사용 설명서의 [액세스 정책 언어 개요](#)
- Amazon Simple Storage Service 사용 설명서의 [버킷 정책 예](#)

## Amazon S3 버킷을 웹 사이트로 구성

Amazon S3 버킷이 웹 사이트로 작동하도록 구성할 수 있습니다. 이렇게 하려면 웹 사이트 구성을 설정해야 합니다.

### 사전 조건

시작하기 전에 [AWS SDK for C++ 사용 시작하기](#)를 읽어보시기 바랍니다.

예제 코드를 다운로드하고 [코드 예제 시작하기](#)에 설명된 대로 솔루션을 빌드합니다.

예제를 실행하려면 코드가 요청을 수행하는 데 사용하는 사용자 프로필이 AWS에서 적절한 권한(서비스 및 동작에 대한)을 보유하고 있어야 합니다. 자세한 내용은 [AWS 자격 증명 제공](#)을 참조하세요.

## 버킷의 웹 사이트 구성 설정

Amazon S3 버킷의 웹 사이트 구성을 설정하려면 `S3Client`의 `PutBucketWebsite` 함수를 호출합니다. 이때 버킷 이름과 [WebsiteConfiguration](#) 객체에 제공되는 웹 사이트 구성이 포함된 [PutBucketWebsiteRequest](#) 객체를 함께 사용합니다.

인덱스 문서 설정은 필수이며, 그 밖의 다른 파라미터는 선택적 파라미터입니다.

### 코드

```
bool AwsDoc::S3::putWebsiteConfig(const Aws::String &bucketName,
                                   const Aws::String &indexPath, const Aws::String
                                   &errorPage,
                                   const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::IndexDocument indexDocument;
    indexDocument.SetSuffix(indexPath);

    Aws::S3::Model::ErrorDocument errorDocument;
    errorDocument.SetKey(errorPage);

    Aws::S3::Model::WebsiteConfiguration websiteConfiguration;
    websiteConfiguration.SetIndexDocument(indexDocument);
    websiteConfiguration.SetErrorDocument(errorDocument);

    Aws::S3::Model::PutBucketWebsiteRequest request;
    request.SetBucket(bucketName);
    request.SetWebsiteConfiguration(websiteConfiguration);

    Aws::S3::Model::PutBucketWebsiteOutcome outcome =
        client.PutBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutBucketWebsite: "
                  << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Success: Set website configuration for bucket '"
                  << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}
```

**Note**

웹 사이트 구성을 설정해도 버킷의 액세스 권한을 수정되지 않습니다. 또한 파일이 웹에 표시되도록 하려면 버킷 내 파일에 대한 퍼블릭 읽기 액세스 권한을 허용하는 버킷 정책을 설정해야 합니다. 자세한 내용은 [Managing Access to Amazon S3 Buckets Using Bucket Policies](#) 단원을 참조하십시오.

[전체 예제](#)는 Github에서 확인하세요.

**버킷의 웹 사이트 구성 가져오기**

Amazon S3 버킷의 웹 사이트 구성을 가져오려면 S3Client의 GetBucketWebsite 함수를 호출합니다. 이때 구성을 가져올 버킷의 이름이 포함된 [GetBucketWebsiteRequest](#)를 와 함께 사용합니다.

구성은 결과 객체 내에서 [GetBucketWebsiteResult](#) 객체로 반환됩니다. 버킷에 대한 웹 사이트 구성이 없으면 null이 반환됩니다.

**코드**

```
bool AwsDoc::S3::getWebsiteConfig(const Aws::String &bucketName,
                                   const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketWebsiteOutcome outcome =
        s3Client.GetBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();

        std::cerr << "Error: GetBucketWebsite: "
                    << err.GetMessage() << std::endl;
    } else {
        Aws::S3::Model::GetBucketWebsiteResult websiteResult = outcome.GetResult();

        std::cout << "Success: GetBucketWebsite: "
                    << std::endl << std::endl
                    << "For bucket '" << bucketName << "':"
                    << std::endl;
```

```

        << "Index page : "
        << websiteResult.GetIndexDocument().GetSuffix()
        << std::endl
        << "Error page: "
        << websiteResult.GetErrorDocument().GetKey()
        << std::endl;
    }

    return outcome.IsSuccess();
}

```

[전체 예제](#)는 Github에서 확인하세요.

## 버킷의 웹 사이트 구성 삭제

Amazon S3 버킷의 웹 사이트 구성을 삭제하려면 S3Client의 DeleteBucketWebsite 함수를 호출합니다. 이때 구성을 삭제할 버킷의 이름이 포함된 [DeleteBucketWebsiteRequest](#)를 함께 사용합니다.

## 코드

```

bool AwsDoc::S3::deleteBucketWebsite(const Aws::String &bucketName,
                                       const Aws::S3::S3ClientConfiguration
                                       &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketWebsiteOutcome outcome =
        client.DeleteBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: deleteBucketWebsite: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
        std::cout << "Website configuration was removed." << std::endl;
    }

    return outcome.IsSuccess();
}

```

[전체 예제](#)는 Github에서 확인하세요.

## 추가 정보

- Amazon Simple Storage Service API 참조의 [PUT 버킷 웹 사이트](#)
- Amazon Simple Storage Service API 참조의 [GET 버킷 웹 사이트](#)
- Amazon Simple Storage Service API 참조의 [DELETE 버킷 웹 사이트](#)

## Amazon S3 작업에 TransferManager 사용

AWS SDK for C++의 `TransferManager` 클래스를 사용하면 로컬 환경에서 Amazon S3로 파일을 안정적으로 전송하고 Amazon S3의 한 위치에서 다른 위치로 객체를 복사할 수 있습니다. `TransferManager`는 전송 진행 상황을 확인하고 업로드 및 다운로드를 일시 중지 또는 재개할 수 있습니다.

### Note

불완전하거나 부분적인 업로드에 대한 요금이 부과되지 않도록 Amazon S3 버킷에서 [AbortIncompleteMultipartUpload](#) 수명 주기 규칙을 활성화하는 것이 좋습니다. 이 규칙은 시작된 후 지정된 일수 내에 완료되지 않은 멀티파트 업로드를 중단하도록 Amazon S3에 지시합니다. 설정된 시간 제한을 초과하면 Amazon S3에서 업로드를 중단한 후 완료되지 않은 업로드 데이터를 삭제합니다. 자세한 내용은 Amazon S3 사용 설명서의 [버킷에서 수명 주기 구성 설정](#)을 참조하세요.

## 사전 조건

시작하기 전에 [AWS SDK for C++ 사용 시작하기](#)를 읽어보시기 바랍니다.

예제 코드를 다운로드하고 [코드 예제 시작하기](#)에 설명된 대로 솔루션을 빌드합니다.

예제를 실행하려면 코드가 요청을 수행하는 데 사용하는 사용자 프로필이 AWS에서 적절한 권한(서비스 및 동작에 대한)을 보유하고 있어야 합니다. 자세한 내용은 [AWS 자격 증명 제공](#)을 참조하세요.

## `TransferManager`를 사용한 객체 업로드 및 다운로드

이 예제는 `TransferManager`가 메모리 내 대형 객체를 전송하는 방식을 보여줍니다. `UploadFile` 및 `DownloadFile` 메서드는 모두 비동기적으로 호출되며 요청 상태를 관리하기 위해 `TransferHandle`을 반환합니다. 업로드된 객체가 `bufferSize`보다 클 경우 멀티파트 업로드가 수행됩니다. `bufferSize`는 기본적으로 5MB로 설정되지만 [TransferManagerConfiguration](#)을 통해 구성할 수 있습니다.

```

    auto s3_client = Aws::MakeShared<Aws::S3::S3Client>("S3Client");
    auto executor =
    Aws::MakeShared<Aws::Utils::Threading::PooledThreadExecutor>("executor", 25);
    Aws::Transfer::TransferManagerConfiguration transfer_config(executor.get());
    transfer_config.s3Client = s3_client;

    // Create buffer to hold data received by the data stream.
    Aws::Utils::Array<unsigned char> buffer(BUFFER_SIZE);

    // The local variable 'streamBuffer' is captured by reference in a lambda.
    // It must persist until all downloading by the 'transfer_manager' is complete.
    Stream::PreallocatedStreamBuf streamBuffer(buffer.GetUnderlyingData(),
    buffer.GetLength());

    auto transfer_manager =
    Aws::Transfer::TransferManager::Create(transfer_config);

    auto uploadHandle = transfer_manager->UploadFile(LOCAL_FILE, BUCKET, KEY,
    "text/plain", Aws::Map<Aws::String, Aws::String>());
    uploadHandle->WaitUntilFinished();
    bool success = uploadHandle->GetStatus() ==
    Transfer::TransferStatus::COMPLETED;

    if (!success)
    {
        auto err = uploadHandle->GetLastError();
        std::cout << "File upload failed: " << err.GetMessage() << std::endl;
    }
    else
    {
        std::cout << "File upload finished." << std::endl;

        auto downloadHandle = transfer_manager->DownloadFile(BUCKET,
            KEY,
            [&]() { //Define a lambda expression for the callback method parameter
            to stream back the data.
                return Aws::New<MyUnderlyingStream>("TestTag", &streamBuffer);
            });
        downloadHandle->WaitUntilFinished();// Block calling thread until download
        is complete.
        auto downStat = downloadHandle->GetStatus();
        if (downStat != Transfer::TransferStatus::COMPLETED)
        {

```

```

        auto err = downloadHandle->GetLastError();
        std::cout << "File download failed: " << err.GetMessage() <<
std::endl;
    }
    std::cout << "File download to memory finished." << std::endl;

```

[전체 예제](#)는 Github에서 확인하세요.

## Amazon S3 작업에 **S3CrtClient** 사용

S3CrtClient 클래스는 AWS SDK for C++ 버전 1.9에서 사용할 수 있으며, Amazon S3와의 대용량 데이터 파일 업로드 및 다운로드 처리량을 개선합니다. 이 릴리스의 개선 사항에 대한 자세한 내용은 [Improving Amazon S3 Throughput with AWS SDK for C++ v1.9](#)를 참조하세요.

S3CrtClient는 [AWS 공통 런타임\(CRT\) 라이브러리](#) 위에 구현됩니다.

### Note

불완전하거나 부분적인 업로드에 대한 요금이 부과되지 않도록 Amazon S3 버킷에서 [AbortIncompleteMultipartUpload](#) 수명 주기 규칙을 활성화하는 것이 좋습니다. 이 규칙은 시작된 후 지정된 일수 내에 완료되지 않은 멀티파트 업로드를 중단하도록 Amazon S3에 지시합니다. 설정된 시간 제한을 초과하면 Amazon S3에서 업로드를 중단한 후 완료되지 않은 업로드 데이터를 삭제합니다. 자세한 내용은 Amazon S3 사용 설명서의 [버킷에서 수명 주기 구성 설정](#)을 참조하세요.

## 사전 조건

시작하기 전에 [AWS SDK for C++ 사용 시작하기](#)를 읽어보시기 바랍니다.

예제 코드를 다운로드하고 [코드 예제 시작하기](#)에 설명된 대로 솔루션을 빌드합니다.

예제를 실행하려면 코드가 요청을 수행하는 데 사용하는 사용자 프로필이 AWS에서 적절한 권한(서비스 및 동작에 대한)을 보유하고 있어야 합니다. 자세한 내용은 [AWS 자격 증명 제공](#)을 참조하세요.

## **S3CrtClient**를 사용한 객체 업로드 및 다운로드

이 예제는 [S3CrtClient](#) 사용 방법을 보여줍니다. 이 예제에서는 버킷을 생성하고, 객체를 업로드하고, 객체를 다운로드한 다음 파일과 버킷을 삭제합니다. PUT 작업은 멀티파트 업로드로 변환됩니다. GET 작업은 여러 개의 “범위 지정” GET 요청으로 변환됩니다. 멀티파트 업로드에 대한 자세한 내용은 Amazon S3 사용 설명서에서 [멀티파트 업로드를 사용한 객체 업로드 및 복사](#)를 참조하세요.

제공된 데이터 파일 `ny.json`은 이 예제에서 멀티파트 업로드로 업로드됩니다. 프로그램을 성공적으로 실행한 후 디버그 로그를 검토하면 이를 확인할 수 있습니다.

업로드에 실패하면 기본 CRT 라이브러리에서 `AbortMultipartUpload`가 발행되어 이미 업로드된 부분을 정리합니다. 그러나 네트워크 케이블이 뽑힌 경우처럼 모든 오류를 내부적으로 처리할 수 있는 것은 아닙니다. 부분적으로 업로드된 데이터가 계정에 남아 있지 않도록(부분적으로 업로드된 데이터도 과금 대상임) Amazon S3 버킷에 수명 주기 규칙을 생성하는 것이 좋습니다. 수명 주기 규칙을 설정하는 방법을 알아보려면 [Amazon S3 비용을 줄이기 위해 불완전 멀티파트 업로드 검색 및 삭제](#)를 참조하세요.

디버그 로그를 사용하여 멀티파트 업로드 세부 정보 탐색

1. `main()`에는 코드 업데이트 지침이 포함된 "TODO" 주석이 있습니다.
  - a. `file_name`: 코드 주석에 제공된 링크에서 샘플 데이터 파일 `ny.json`를 다운로드하거나 자체 대용량 데이터 파일을 사용합니다.
  - b. `region`: enum을 사용하여 `region` 변수를 계정의 AWS 리전으로 업데이트합니다. 계정의 리전을 찾으려면 AWS Management 콘솔에 로그인한 후 오른쪽 상단 모서리에서 리전을 찾습니다.
2. 예제를 빌드합니다.
3. `file_name` 변수로 지정된 파일을 실행 파일 폴더로 복사하고 `s3-crt-demo` 실행 파일을 실행합니다.
4. 실행 파일 폴더에서 최신 `.log` 파일을 찾습니다.
5. 로그 파일을 열고 검색을 선택한 다음 **partNumber**를 입력합니다.
6. 로그에는 다음과 유사한 항목이 포함되어 있습니다. 여기서 `partNumber`와 `uploadId`는 업로드된 파일의 각 부분마다 지정됩니다.

```
PUT /my-object
partNumber=1&uploadId=gsk8vDbmn1A5EseDo._LDEgq22Qmt0SeuszYxMsZ9ABt503VqDIFOP8
content-length:8388608 host:my-bucketasdfasdf.s3.us-east-2.amazonaws.com x-amz-content-sha256:UNSIGNED-PAYLOAD
```

및

```
PUT /my-object
partNumber=2&uploadId=gsk8vDbmn1A5EseDo._LDEgq22Qmt0SeuszYxMsZ9ABt503VqDIFOP8
content-length:8388608 host:my-bucketasdfasdf.s3.us-east-2.amazonaws.com x-amz-content-sha256:UNSIGNED-PAYLOAD
```

[전체 예제](#)는 Github에서 확인하세요.

## AWS SDK for C++를 사용한 Amazon SQS 코드 예제

Amazon Simple Queue Service(Amazon SQS)는 마이크로서비스, 분산 시스템 및 서버리스 애플리케이션을 분리하고 규모 조정하는 완전 관리형 메시지 대기열 서비스입니다. 다음 예제를 사용하면 AWS SDK for C++로 [Amazon SQS](#)를 프로그래밍할 수 있습니다.

### Note

이 안내서에는 특정 기술을 시연하는 데 필요한 코드만 제공되며 [전체 예제 코드는 GitHub에서 확인할 수 있습니다](#). GitHub에서 단일 소스 파일을 다운로드하거나, 리포지토리를 로컬에 복제하여 모든 예제를 가져오고 빌드하고 실행할 수 있습니다.

### 주제

- [Amazon SQS 메시지 대기열 작업](#)
- [Amazon SQS 메시지 전송, 수신 및 삭제](#)
- [Amazon SQS 메시지 대기열에 대한 긴 폴링 활성화](#)
- [Amazon SQS에서 표시 제한 시간 설정](#)
- [Amazon SQS에서 DLQ\(Dead Letter Queue\) 사용](#)

## Amazon SQS 메시지 대기열 작업

메시지 대기열은 Amazon SQS에서 메시지를 안정적으로 전송하기 위해 사용하는 논리적 컨테이너입니다. 표준과 선입선출(FIFO), 이렇게 두 가지 유형의 대기열이 있습니다. 이들 대기열과 이러한 유형 간의 차이점에 대해 자세히 알아보려면 [Amazon Simple Queue Service 개발자 안내서](#)를 참조하세요.

이 C++ 예제에서는 AWS SDK for C++를 사용하여 Amazon SQS 대기열의 URL을 생성, 나열, 삭제 및 가져오는 방법을 보여줍니다.

### 사전 조건

시작하기 전에 [AWS SDK for C++ 사용 시작하기](#)를 읽어보시기 바랍니다.

예제 코드를 다운로드하고 [코드 예제 시작하기](#)에 설명된 대로 솔루션을 빌드합니다.

예제를 실행하려면 코드가 요청을 수행하는 데 사용하는 사용자 프로필이 AWS에서 적절한 권한(서비스 및 동작에 대한)을 보유하고 있어야 합니다. 자세한 내용은 [AWS 자격 증명 제공](#)을 참조하세요.

## 대기열 생성

SQSClient 클래스의 CreateQueue 멤버 함수를 사용하고, 대기열 파라미터를 설명하는 [CreateQueueRequest](#) 객체를 제공합니다.

### 포함 파일

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/CreateQueueRequest.h>
#include <iostream>
```

### 코드

```
Aws::SQS::SQSClient sqsClient(clientConfiguration);

Aws::SQS::Model::CreateQueueRequest request;
request.SetQueueName(queueName);

const Aws::SQS::Model::CreateQueueOutcome outcome = sqsClient.CreateQueue(request);
if (outcome.IsSuccess()) {
    std::cout << "Successfully created queue " << queueName << " with a queue URL "
               << outcome.GetResult().GetQueueUrl() << "." << std::endl;
}
else {
    std::cerr << "Error creating queue " << queueName << ": " <<
              outcome.GetError().GetMessage() << std::endl;
}
```

[전체 예제](#)를 참조하세요.

## 대기열 목록 나열

계정의 Amazon SQS 대기열을 나열하려면 SQSClient 클래스의 ListQueues 멤버 함수를 호출하고 [ListQueuesRequest](#) 객체를 전달합니다.

### 포함 파일

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/ListQueuesRequest.h>
#include <iostream>
```

## 코드

```

Aws::SQS::SQSClient sqsClient(clientConfiguration);

Aws::SQS::Model::ListQueuesRequest listQueuesRequest;

Aws::String nextToken; // Used for pagination.
Aws::Vector<Aws::String> allQueueUrls;

do {
    if (!nextToken.empty()) {
        listQueuesRequest.SetNextToken(nextToken);
    }
    const Aws::SQS::Model::ListQueuesOutcome outcome = sqsClient.ListQueues(
        listQueuesRequest);
    if (outcome.IsSuccess()) {
        const Aws::Vector<Aws::String> &queueUrls =
outcome.GetResult().GetQueueUrls();
        allQueueUrls.insert(allQueueUrls.end(),
                            queueUrls.begin(),
                            queueUrls.end());

        nextToken = outcome.GetResult().GetNextToken();
    }
    else {
        std::cerr << "Error listing queues: " <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }
} while (!nextToken.empty());

std::cout << allQueueUrls.size() << " Amazon SQS queue(s) found." << std::endl;
for (const auto &iter: allQueueUrls) {
    std::cout << " " << iter << std::endl;
}

```

[전체 예제](#)를 참조하세요.

## 대기열 URL 가져오기

기존 Amazon SQS 대기열의 URL을 가져오려면 SQSClient 클래스의 GetQueueUrl 멤버 함수를 호출합니다.

## 포함 파일

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/GetQueueUrlRequest.h>
#include <iostream>
```

## 코드

```
Aws::SQS::SQSClient sqsClient(clientConfiguration);

Aws::SQS::Model::GetQueueUrlRequest request;
request.SetQueueName(queueName);

const Aws::SQS::Model::GetQueueUrlOutcome outcome = sqsClient.GetQueueUrl(request);
if (outcome.IsSuccess()) {
    std::cout << "Queue " << queueName << " has url " <<
        outcome.GetResult().GetQueueUrl() << std::endl;
}
else {
    std::cerr << "Error getting url for queue " << queueName << ": " <<
        outcome.GetError().GetMessage() << std::endl;
}
```

[전체 예제](#)를 참조하세요.

## 대기열 삭제

SQSClient 클래스의 DeleteQueue 멤버 함수에 대한 [URL](#)을 제공합니다.

## 포함 파일

```
#include <aws/core/Aws.h>
#include <aws/core/client/DefaultRetryStrategy.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/DeleteQueueRequest.h>
#include <iostream>
```

## 코드

```
Aws::SQS::Model::DeleteQueueRequest request;
request.SetQueueUrl(queueURL);
```

```
const Aws::SQS::Model::DeleteQueueOutcome outcome = sqsClient.DeleteQueue(request);
if (outcome.IsSuccess()) {
    std::cout << "Successfully deleted queue with url " << queueURL <<
        std::endl;
}
else {
    std::cerr << "Error deleting queue " << queueURL << ": " <<
        outcome.GetError().GetMessage() << std::endl;
}
```

[전체 예제](#)를 참조하세요.

## 추가 정보

- Amazon Simple Queue Service 개발자 안내서의 [Amazon SQS 대기열의 작동 방식](#)
- Amazon Simple Queue Service API 참조의 [CreateQueue](#)
- Amazon Simple Queue Service API 참조의 [GetQueueUrl](#)
- Amazon Simple Queue Service API 참조의 [ListQueues](#)
- Amazon Simple Queue Service API 참조의 [DeleteQueues](#)

## Amazon SQS 메시지 전송, 수신 및 삭제

메시지는 항상 [SQS 대기열](#)을 사용하여 전달됩니다. 이 C++ 예제에서는 AWS SDK for C++를 사용하여 SQS 대기열에서 Amazon SQS 메시지를 전송, 수신 및 삭제하는 방법을 보여줍니다.

### 사전 조건

시작하기 전에 [AWS SDK for C++ 사용 시작하기](#)를 읽어보시기 바랍니다.

예제 코드를 다운로드하고 [코드 예제 시작하기](#)에 설명된 대로 솔루션을 빌드합니다.

예제를 실행하려면 코드가 요청을 수행하는 데 사용하는 사용자 프로필이 AWS에서 적절한 권한(서비스 및 동작에 대한)을 보유하고 있어야 합니다. 자세한 내용은 [AWS 자격 증명 제공](#)을 참조하세요.

### 메시지 보내기

Amazon SQS 대기열에 단일 메시지를 추가하려면 SQSClient 클래스의 SendMessage 멤버 함수를 호출합니다. SendMessage에는 대기열 [URL](#), 메시지 본문 및 선택적 지연 값(초)이 포함된 [SendMessageRequest](#) 객체를 제공합니다.

## 포함 파일

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/SendMessageRequest.h>
#include <iostream>
```

## 코드

```
Aws::SQS::SQSClient sqsClient(clientConfiguration);

Aws::SQS::Model::SendMessageRequest request;
request.SetQueueUrl(queueUrl);
request.SetMessageBody(messageBody);

const Aws::SQS::Model::SendMessageOutcome outcome = sqsClient.SendMessage(request);
if (outcome.IsSuccess()) {
    std::cout << "Successfully sent message to " << queueUrl <<
        std::endl;
}
else {
    std::cerr << "Error sending message to " << queueUrl << ": " <<
        outcome.GetError().GetMessage() << std::endl;
}
```

[전체 예제](#)를 참조하세요.

## 메시지 수신

SQSClient 클래스의 ReceiveMessage 멤버 함수를 호출하고 대기열의 URL을 이 함수에 전달하여 현재 대기열에 있는 모든 메시지를 가져옵니다. 메시지는 [Message](#) 객체의 목록으로 반환됩니다.

## 포함 파일

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/ReceiveMessageRequest.h>
#include <iostream>
```

## 코드

```
Aws::SQS::SQSClient sqsClient(clientConfiguration);
```

```

Aws::SQS::Model::ReceiveMessageRequest request;
request.SetQueueUrl(queueUrl);
request.SetMaxNumberOfMessages(1);

const Aws::SQS::Model::ReceiveMessageOutcome outcome = sqsClient.ReceiveMessage(
    request);
if (outcome.IsSuccess()) {

    const Aws::Vector<Aws::SQS::Model::Message> &messages =
        outcome.GetResult().GetMessages();
    if (!messages.empty()) {
        const Aws::SQS::Model::Message &message = messages[0];
        std::cout << "Received message:" << std::endl;
        std::cout << "  MessageId: " << message.GetMessageId() << std::endl;
        std::cout << "  ReceiptHandle: " << message.GetReceiptHandle() <<
std::endl;
        std::cout << "  Body: " << message.GetBody() << std::endl << std::endl;
    }
    else {
        std::cout << "No messages received from queue " << queueUrl <<
std::endl;
    }
}
else {
    std::cerr << "Error receiving message from queue " << queueUrl << ": "
        << outcome.GetError().GetMessage() << std::endl;
}
}

```

[전체 예제](#)를 참조하세요.

## 수신 후 메시지 삭제

메시지를 수신하고 내용을 처리한 후에는 메시지의 수신 핸들과 대기열 URL을 SQSClient 클래스의 DeleteMessage 멤버 함수로 전송하여 대기열에서 메시지를 삭제합니다.

## 포함 파일

```

#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/DeleteMessageRequest.h>
#include <iostream>

```

## 코드

```
Aws::SQS::Model::DeleteMessageRequest request;
request.SetQueueUrl(queueUrl);
request.SetReceiptHandle(messageReceiptHandle);

const Aws::SQS::Model::DeleteMessageOutcome outcome = sqsClient.DeleteMessage(
    request);
if (outcome.IsSuccess()) {
    std::cout << "Successfully deleted message from queue " << queueUrl
        << std::endl;
}
else {
    std::cerr << "Error deleting message from queue " << queueUrl << ": " <<
        outcome.GetError().GetMessage() << std::endl;
}
```

[전체 예제](#)를 참조하세요.

## 추가 정보

- Amazon Simple Queue Service 개발자 안내서의 [Amazon SQS 대기열의 작동 방식](#)
- Amazon Simple Queue Service API 참조의 [SendMessage](#)
- Amazon Simple Queue Service API 참조의 [SendMessageBatch](#)
- Amazon Simple Queue Service API 참조의 [ReceiveMessage](#)
- Amazon Simple Queue Service API 참조의 [DeleteMessage](#)

## Amazon SQS 메시지 대기열에 대한 긴 폴링 활성화

Amazon SQS는 기본적으로 짧은 폴링 방식을 사용합니다. 즉, 응답에 포함할 메시지가 있는지 확인하기 위해 가중 무작위 분포에 기반하여 서버의 하위 세트만 쿼리합니다.

긴 폴링은 Amazon SQS 대기열에 전송된 ReceiveMessage 요청에 대한 응답으로 반환할 메시지가 없을 때 빈 응답의 수를 줄이고, 거짓 빈 응답을 제거하여 Amazon SQS 사용 비용을 줄여줍니다. 긴 폴링 빈도는 1-20초로 설정할 수 있습니다.

## 사전 조건

시작하기 전에 [AWS SDK for C++ 사용 시작하기](#)를 읽어보시기 바랍니다.

예제 코드를 다운로드하고 [코드 예제 시작하기](#)에 설명된 대로 솔루션을 빌드합니다.

예제를 실행하려면 코드가 요청을 수행하는 데 사용하는 사용자 프로필이 AWS에서 적절한 권한(서비스 및 동작에 대한)을 보유하고 있어야 합니다. 자세한 내용은 [AWS 자격 증명 제공](#)을 참조하세요.

### 대기열 생성 시 긴 폴링 활성화

Amazon SQS 대기열을 생성할 때 긴 폴링을 활성화하려면 SQSClient 클래스의 CreateQueue 멤버 함수를 호출하기 전에 [CreateQueueRequest](#) 객체에 ReceiveMessageWaitTimeSeconds 속성을 설정합니다.

### 포함 파일

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/CreateQueueRequest.h>
#include <iostream>
```

### 코드

```
Aws::SQS::SQSClient sqsClient(clientConfiguration);

Aws::SQS::Model::CreateQueueRequest request;
request.SetQueueName(queueName);
request.AddAttributes(
    Aws::SQS::Model::QueueAttributeName::ReceiveMessageWaitTimeSeconds,
    pollTimeSeconds);

const Aws::SQS::Model::CreateQueueOutcome outcome = sqsClient.CreateQueue(request);
if (outcome.IsSuccess()) {
    std::cout << "Successfully created queue " << queueName <<
        std::endl;
}
else {
    std::cout << "Error creating queue " << queueName << ": " <<
        outcome.GetError().GetMessage() << std::endl;
}
```

[전체 예제](#)를 참조하세요.

## 기존 대기열에 대한 긴 폴링 활성화

대기열을 생성할 때 긴 폴링을 활성화하는 것 외에도, SQSClient 클래스의 SetQueueAttributes 멤버 함수를 호출하기 전에 [SetQueueAttributesRequest](#)에 ReceiveMessageWaitTimeSeconds를 설정하여 기존 대기열에서도 이를 활성화할 수 있습니다.

### 포함 파일

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/SetQueueAttributesRequest.h>
#include <iostream>
```

### 코드

```
Aws::SQS::SQSClient sqsClient(clientConfiguration);

Aws::SQS::Model::SetQueueAttributesRequest request;
request.SetQueueUrl(queueURL);
request.AddAttributes(
    Aws::SQS::Model::QueueAttributeName::ReceiveMessageWaitTimeSeconds,
    pollTimeSeconds);

const Aws::SQS::Model::SetQueueAttributesOutcome outcome =
sqsClient.SetQueueAttributes(
    request);
if (outcome.IsSuccess()) {
    std::cout << "Successfully updated long polling time for queue " <<
        queueURL << " to " << pollTimeSeconds << std::endl;
}
else {
    std::cout << "Error updating long polling time for queue " <<
        queueURL << ": " << outcome.GetError().GetMessage() <<
        std::endl;
}
```

[전체 예제](#)를 참조하세요.

## 메시지 수신 시 긴 폴링 활성화

SQSClient 클래스의 ReceiveMessage 멤버 함수에 전달하는 [ReceiveMessageRequest](#)에서 대기 시간을 초 단위로 설정하여 메시지 수신 시 긴 폴링을 활성화할 수 있습니다.

**Note**

AWS 클라이언트의 요청 제한 시간이 최대 긴 폴링 시간(20초)보다 길어야만 다음 폴링 이벤트 까지 기다리는 동안 ReceiveMessage 요청의 제한 시간이 초과되지 않습니다.

**포함 파일**

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/ReceiveMessageRequest.h>
```

**코드**

```
Aws::Sqs::SQSClient sqsClient(customConfiguration);

Aws::Sqs::Model::ReceiveMessageRequest request;
request.SetQueueUrl(queueUrl);
request.SetMaxNumberOfMessages(1);
request.SetWaitTimeSeconds(waitTimeSeconds);

auto outcome = sqsClient.ReceiveMessage(request);
if (outcome.IsSuccess()) {
    const auto &messages = outcome.GetResult().GetMessages();
    if (messages.empty()) {
        std::cout << "No messages received from queue " << queueUrl <<
            std::endl;
    }
    else {
        const auto &message = messages[0];
        std::cout << "Received message:" << std::endl;
        std::cout << "  MessageId: " << message.GetMessageId() << std::endl;
        std::cout << "  ReceiptHandle: " << message.GetReceiptHandle() <<
std::endl;
        std::cout << "  Body: " << message.GetBody() << std::endl << std::endl;
    }
}
else {
    std::cout << "Error receiving message from queue " << queueUrl << ": "
        << outcome.GetError().GetMessage() << std::endl;
}
```

[전체 예제](#)를 참조하세요.

## 추가 정보

- Amazon Simple Queue Service 개발자 안내서의 [Amazon SQS 긴 폴링](#)
- Amazon Simple Queue Service API 참조의 [CreateQueue](#)
- Amazon Simple Queue Service API 참조의 [ReceiveMessage](#)
- Amazon Simple Queue Service API 참조의 [SetQueueAttributes](#)

## Amazon SQS에서 표시 제한 시간 설정

Amazon SQS에서 메시지가 수신되면 수신 여부를 확인하기 위해 삭제될 때까지 대기열에 남아 있습니다. 수신되었지만 삭제되지 않은 메시지는 메시지가 처리 및 삭제되기 전에 두 번 이상 수신되지 않도록 하기 위해 지정된 제한 시간 초과 이후에는 후속 요청에서 제공됩니다.

[표준 대기열](#)을 사용 중인 경우 제한 시간 초과를 설정해도 메시지가 두 번 이상 수신되지 않는다고 장담할 수 없습니다. 표준 대기열을 사용 중인 경우 동일 메시지가 두 번 이상 전달된 경우를 코드에서 처리할 수 있도록 해야 합니다.

## 사전 조건

시작하기 전에 [AWS SDK for C++ 사용 시작하기](#)를 읽어보시기 바랍니다.

예제 코드를 다운로드하고 [코드 예제 시작하기](#)에 설명된 대로 솔루션을 빌드합니다.

예제를 실행하려면 코드가 요청을 수행하는 데 사용하는 사용자 프로필이 AWS에서 적절한 권한(서비스 및 동작에 대한)을 보유하고 있어야 합니다. 자세한 내용은 [AWS 자격 증명 제공](#)을 참조하세요.

## 메시지 수신 시 메시지 표시 제한 시간 설정

메시지를 수신한 후에는 SQSClient 클래스의 ChangeMessageVisibility 멤버 함수에 전달되는 [ChangeMessageVisibilityRequest](#)에 해당 수신 핸들을 전달하여 표시 제한 시간을 수정할 수 있습니다.

## 포함 파일

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/ChangeMessageVisibilityRequest.h>
#include <aws/sqs/model/ReceiveMessageRequest.h>
```

```
#include <iostream>
```

## 코드

```
Aws::SQS::Model::ChangeMessageVisibilityRequest request;
request.SetQueueUrl(queue_url);
request.SetReceiptHandle(messageReceiptHandle);
request.SetVisibilityTimeout(visibilityTimeoutSeconds);

auto outcome = sqsClient.ChangeMessageVisibility(request);
if (outcome.IsSuccess()) {
    std::cout << "Successfully changed visibility of message " <<
        messageReceiptHandle << " from queue " << queue_url << std::endl;
}
else {
    std::cout << "Error changing visibility of message from queue "
        << queue_url << ": " <<
        outcome.GetError().GetMessage() << std::endl;
}
```

[전체 예제](#)를 참조하세요.

## 추가 정보

- Amazon Simple Queue Service 개발자 안내서의 [표시 제한 시간](#)
- Amazon Simple Queue Service API 참조의 [SetQueueAttributes](#)
- Amazon Simple Queue Service API 참조의 [GetQueueAttributes](#)
- Amazon Simple Queue Service API 참조의 [ReceiveMessage](#)
- Amazon Simple Queue Service API 참조의 [ChangeMessageVisibility](#)
- Amazon Simple Queue Service API 참조의 [ChangeMessageVisibilityBatch](#)

## Amazon SQS에서 DLQ(Dead Letter Queue) 사용

Amazon SQS는 Dead Letter Queue(DLQ)를 지원합니다. Dead Letter Queue(DLQ)는 성공적으로 처리할 수 없는 메시지에 대해 다른 대기열이 대상으로 지정할 수 있는 대기열입니다. 배달 못한 편지 대기열에서 이 메시지를 구분하고 격리하여 처리에 실패한 이유를 확인할 수 있습니다.

Dead Letter Queue(DLQ)를 생성하려면 먼저 리드라이브 정책을 생성한 후 대기열의 속성에 해당 정책을 설정해야 합니다.

**⚠ Important**

Dead Letter Queue(DLQ)는 소스 대기열과 동일한 유형의 대기열(FIFO 또는 표준)이어야 합니다. 또한 소스 대기열과 동일한 AWS 계정 및 AWS 리전을 사용하여 생성해야 합니다.

**사전 조건**

시작하기 전에 [AWS SDK for C++ 사용 시작하기](#)를 읽어보시기 바랍니다.

예제 코드를 다운로드하고 [코드 예제 시작하기](#)에 설명된 대로 솔루션을 빌드합니다.

예제를 실행하려면 코드가 요청을 수행하는 데 사용하는 사용자 프로필이 AWS에서 적절한 권한(서비스 및 동작에 대한)을 보유하고 있어야 합니다. 자세한 내용은 [AWS 자격 증명 제공](#)을 참조하세요.

**리드라이브 정책 생성**

리드라이브 정책은 JSON으로 지정됩니다. 이 정책을 생성하려면 AWS SDK for C++와 함께 제공된 JSON 유틸리티 클래스를 사용할 수 있습니다.

다음은 Dead Letter Queue(DLQ)의 ARN과 메시지가 DLQ로 전송되기 전에 수신되었으나 처리되지 않은 최대 횟수를 제공하여 리드라이브 정책을 생성하는 예시 함수입니다.

**포함 파일**

```
#include <aws/core/Aws.h>
#include <aws/core/utils/json/JsonSerializer.h>
```

**코드**

```
Aws::String MakeRedrivePolicy(const Aws::String &queueArn, int maxReceiveCount) {
    Aws::Utils::Json::JsonValue redrive_arn_entry;
    redrive_arn_entry.AsString(queueArn);

    Aws::Utils::Json::JsonValue max_msg_entry;
    max_msg_entry.AsInteger(maxReceiveCount);

    Aws::Utils::Json::JsonValue policy_map;
    policy_map.WithObject("deadLetterTargetArn", redrive_arn_entry);
    policy_map.WithObject("maxReceiveCount", max_msg_entry);
}
```

```
return policy_map.View().WriteReadable();
}
```

[전체 예제](#)를 참조하세요.

## 소스 대기열에서 리드라이브 정책 설정

Dead Letter Queue(DLQ) 설정을 완료하려면 SQSClient 클래스의 SetQueueAttributes 멤버 함수를 호출합니다. 이때 JSON 재전송 정책으로 RedrivePolicy 속성을 설정한 [SetQueueAttributesRequest](#) 객체를 함께 사용합니다.

## 포함 파일

```
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/SetQueueAttributesRequest.h>
#include <iostream>
```

## 코드

```
Aws::SQS::Model::SetQueueAttributesRequest request;
request.SetQueueUrl(srcQueueUrl);
request.AddAttributes(
    Aws::SQS::Model::QueueAttributeName::RedrivePolicy,
    redrivePolicy);

const Aws::SQS::Model::SetQueueAttributesOutcome outcome =
    sqsClient.SetQueueAttributes(request);
if (outcome.IsSuccess()) {
    std::cout << "Successfully set dead letter queue for queue " <<
        srcQueueUrl << " to " << deadLetterQueueARN << std::endl;
}
else {
    std::cerr << "Error setting dead letter queue for queue " <<
        srcQueueUrl << ": " << outcome.GetError().GetMessage() <<
        std::endl;
}
```

[전체 예제](#)를 참조하세요.

## 추가 정보

- Amazon Simple Queue Service 개발자 안내서의 [Dead Letter Queue\(DLQ\) 사용](#)

- Amazon Simple Queue Service API 참조의 [SetQueueAttributes](#)

## SDK for C++ 코드 예제

이 주제의 코드 예제에서는 AWS SDK for C++ 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 C++를 사용한 ACM 예제](#)
- [SDK for C++를 사용한 API Gateway 예제](#)
- [SDK for C++를 사용한 Aurora 예제](#)
- [SDK for C++를 사용한 Auto Scaling 예제](#)
- [SDK for C++를 사용한 CloudTrail 예제](#)
- [SDK for C++를 사용한 CloudWatch 예제](#)
- [SDK for C++를 사용한 CloudWatch Logs 예제](#)
- [SDK for C++를 사용한 CodeBuild 예제](#)
- [SDK for C++를 사용한 Amazon Cognito 자격 증명 공급자 예제](#)
- [SDK for C++를 사용한 DynamoDB 예제](#)
- [SDK for C++를 사용한 Amazon EC2 예제](#)
- [SDK for C++를 사용한 EventBridge 예제](#)
- [AWS Glue SDK for C++를 사용한 예제](#)
- [SDK for C++를 사용한 HealthImaging 예제](#)
- [SDK for C++를 사용한 IAM 예제](#)
- [AWS IoT SDK for C++를 사용한 예제](#)

- [AWS IoT data SDK for C++를 사용한 예제](#)
- [SDK for C++를 사용한 Lambda 예제](#)
- [SDK for C++를 사용한 MediaConvert 예제](#)
- [SDK for C++를 사용한 Amazon RDS 예제](#)
- [SDK for C++를 사용한 Amazon RDS Data Service 예제](#)
- [SDK for C++를 사용한 Amazon Rekognition 예제](#)
- [SDK for C++를 사용한 Amazon S3용 예제](#)
- [SDK for C++를 사용한 Secrets Manager 예제](#)
- [SDK for C++를 사용한 Amazon SES 예제](#)
- [SDK for C++를 사용한 Amazon SNS 예제](#)
- [SDK for C++를 사용한 Amazon SQS 예제](#)
- [AWS STS SDK for C++를 사용한 예제](#)
- [SDK for C++를 사용한 Amazon Transcribe 스트리밍 예제](#)

## SDK for C++를 사용한 ACM 예제

다음 코드 예제에서는 ACM과 AWS SDK for C++ 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

#### AddTagsToCertificate

다음 코드 예시는 AddTagsToCertificate의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Add tags to an AWS Certificate Manager (ACM) certificate.
/*!
    \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
    \param tagKey: The key for the tag.
    \param tagValue: The value for the tag.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::ACM::addTagsToCertificate(const Aws::String &certificateArn,
                                       const Aws::String &tagKey,
                                       const Aws::String &tagValue,
                                       const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::AddTagsToCertificateRequest request;
    Aws::Vector<Aws::ACM::Model::Tag> tags;
    Aws::ACM::Model::Tag tag;

    tag.WithKey(tagKey).WithValue(tagValue);
    tags.push_back(tag);

    request.WithCertificateArn(certificateArn).WithTags(tags);

    Aws::ACM::Model::AddTagsToCertificateOutcome outcome =
        acmClient.AddTagsToCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: addTagsToCertificate: " <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Success: Tag with key '" << tagKey <<
            "' and value '" << tagValue <<

```

```

        "" added to certificate with ARN "" <<
        certificateArn << "." << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [AddTagsToCertificate](#)를 참조하세요.

## DeleteCertificate

다음 코드 예시는 DeleteCertificate의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Delete an AWS Certificate Manager (ACM) certificate.
/*!
    \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::ACM::deleteCertificate(const Aws::String &certificateArn,
                                    const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::DeleteCertificateRequest request;
    request.WithCertificateArn(certificateArn);

    Aws::ACM::Model::DeleteCertificateOutcome outcome =
        acmClient.DeleteCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: DeleteCertificate: " <<
            outcome.GetError().GetMessage() << std::endl;
    }
}

```

```

    }
    else {
        std::cout << "Success: The certificate with the ARN '" <<
            certificateArn << "' is deleted." << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteCertificate](#)를 참조하세요.

## DescribeCertificate

다음 코드 예시는 DescribeCertificate의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Describe an AWS Certificate Manager (ACM) certificate.
/*!
    \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
 */
bool AwsDoc::ACM::describeCertificate(const Aws::String &certificateArn,
                                     const Aws::Client::ClientConfiguration
                                     &clientConfiguration) {
    Aws::ACM::ACMClient acm_client(clientConfiguration);

    Aws::ACM::Model::DescribeCertificateRequest request;
    request.WithCertificateArn(certificateArn);

    Aws::ACM::Model::DescribeCertificateOutcome outcome =
        acm_client.DescribeCertificate(request);

    if (!outcome.IsSuccess()) {

```

```

        std::cerr << "Error: DescribeCertificate: " <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else {
        Aws::ACM::Model::CertificateDetail certificate =
            outcome.GetResult().GetCertificate();

        std::cout << "Success: Information about certificate "
            "with ARN '" << certificateArn << "':" << std::endl <<
std::endl;

        std::cout << "ARN:                " << certificate.GetCertificateArn()
            << std::endl;
        std::cout << "Authority ARN:            " <<
            certificate.GetCertificateAuthorityArn() << std::endl;
        std::cout << "Created at (GMT):        " <<
            certificate.GetCreatedAt().ToGmtString(
                Aws::Utils::DateFormat::ISO_8601)
            << std::endl;
        std::cout << "Domain name:            " << certificate.GetDomainName()
            << std::endl;

        Aws::Vector<Aws::ACM::Model::DomainValidation> options =
            certificate.GetDomainValidationOptions();

        if (!options.empty()) {
            std::cout << std::endl << "Domain validation information: "
                << std::endl << std::endl;

            for (auto &validation: options) {
                std::cout << "    Domain name:                " <<
                    validation.GetDomainName() << std::endl;

                const Aws::ACM::Model::ResourceRecord &record =
                    validation.GetResourceRecord();

                std::cout << "    Resource record name:        " <<
                    record.GetName() << std::endl;

                Aws::ACM::Model::RecordType recordType = record.GetType();
                Aws::String type;

                switch (recordType) {
                    case Aws::ACM::Model::RecordType::CNAME:

```

```
        type = "CNAME";
        break;
    case Aws::ACM::Model::RecordType::NOT_SET:
        type = "Not set";
        break;
    default:
        type = "Cannot determine.";
        break;
}

std::cout << "    Resource record type:      " << type <<
    std::endl;

std::cout << "    Resource record value:      " <<
    record.GetValue() << std::endl;

std::cout << "    Validation domain:          " <<
    validation.GetValidationDomain() << std::endl;

Aws::Vector<Aws::String> emails =
    validation.GetValidationEmails();

if (!emails.empty()) {
    std::cout << "    Validation emails:" << std::endl <<
        std::endl;

    for (auto &email: emails) {
        std::cout << "        " << email << std::endl;
    }

    std::cout << std::endl;
}

Aws::ACM::Model::ValidationMethod validationMethod =
    validation.GetValidationMethod();
Aws::String method;

switch (validationMethod) {
    case Aws::ACM::Model::ValidationMethod::DNS:
        method = "DNS";
        break;
    case Aws::ACM::Model::ValidationMethod::EMAIL:
        method = "Email";
        break;
}
```

```

        case Aws::ACM::Model::ValidationMethod::NOT_SET:
            method = "Not set";
            break;
        default:
            method = "Cannot determine";
    }

    std::cout << "    Validation method:          " <<
                method << std::endl;

    Aws::ACM::Model::DomainStatus domainStatus =
        validation.GetValidationStatus();
    Aws::String status;

    switch (domainStatus) {
        case Aws::ACM::Model::DomainStatus::FAILED:
            status = "Failed";
            break;
        case Aws::ACM::Model::DomainStatus::NOT_SET:
            status = "Not set";
            break;
        case Aws::ACM::Model::DomainStatus::PENDING_VALIDATION:
            status = "Pending validation";
            break;
        case Aws::ACM::Model::DomainStatus::SUCCESS:
            status = "Success";
            break;
        default:
            status = "Cannot determine";
    }

    std::cout << "    Domain validation status: " << status <<
                std::endl << std::endl;

    }
}

Aws::Vector<Aws::ACM::Model::ExtendedKeyUsage> usages =
    certificate.GetExtendedKeyUsages();

if (!usages.empty()) {
    std::cout << std::endl << "Extended key usages:" <<
                std::endl << std::endl;
}

```

```
for (auto &usage: usages) {
    Aws::ACM::Model::ExtendedKeyUsageName usageName =
        usage.GetName();
    Aws::String name;

    switch (usageName) {
        case Aws::ACM::Model::ExtendedKeyUsageName::ANY:
            name = "Any";
            break;
        case Aws::ACM::Model::ExtendedKeyUsageName::CODE_SIGNING:
            name = "Code signing";
            break;
        case Aws::ACM::Model::ExtendedKeyUsageName::CUSTOM:
            name = "Custom";
            break;
        case Aws::ACM::Model::ExtendedKeyUsageName::EMAIL_PROTECTION:
            name = "Email protection";
            break;
        case Aws::ACM::Model::ExtendedKeyUsageName::IPSEC_END_SYSTEM:
            name = "IPSEC end system";
            break;
        case Aws::ACM::Model::ExtendedKeyUsageName::IPSEC_TUNNEL:
            name = "IPSEC tunnel";
            break;
        case Aws::ACM::Model::ExtendedKeyUsageName::IPSEC_USER:
            name = "IPSEC user";
            break;
        case Aws::ACM::Model::ExtendedKeyUsageName::NONE:
            name = "None";
            break;
        case Aws::ACM::Model::ExtendedKeyUsageName::NOT_SET:
            name = "Not set";
            break;
        case Aws::ACM::Model::ExtendedKeyUsageName::OCSP_SIGNING:
            name = "OCSP signing";
            break;
        case Aws::ACM::Model::ExtendedKeyUsageName::TIME_STAMPING:
            name = "Time stamping";
            break;
        case
            Aws::ACM::Model::ExtendedKeyUsageName::TLS_WEB_CLIENT_AUTHENTICATION:
            name = "TLS web client authentication";
            break;
```

```

        case
        Aws::ACM::Model::ExtendedKeyUsageName::TLS_WEB_SERVER_AUTHENTICATION:
            name = "TLS web server authentication";
            break;
        default:
            name = "Cannot determine";
    }

    std::cout << "  Name: " << name << std::endl;
    std::cout << "  OID: " << usage.GetOID() <<
        std::endl << std::endl;
}

std::cout << std::endl;
}

Aws::ACM::Model::CertificateStatus certificateStatus =
    certificate.GetStatus();
Aws::String status;

switch (certificateStatus) {
    case Aws::ACM::Model::CertificateStatus::EXPIRED:
        status = "Expired";
        break;
    case Aws::ACM::Model::CertificateStatus::FAILED:
        status = "Failed";
        break;
    case Aws::ACM::Model::CertificateStatus::INACTIVE:
        status = "Inactive";
        break;
    case Aws::ACM::Model::CertificateStatus::ISSUED:
        status = "Issued";
        break;
    case Aws::ACM::Model::CertificateStatus::NOT_SET:
        status = "Not set";
        break;
    case Aws::ACM::Model::CertificateStatus::PENDING_VALIDATION:
        status = "Pending validation";
        break;
    case Aws::ACM::Model::CertificateStatus::REVOKED:
        status = "Revoked";
        break;
    case Aws::ACM::Model::CertificateStatus::VALIDATION_TIMED_OUT:
        status = "Validation timed out";

```

```

        break;
    default:
        status = "Cannot determine";
    }

    std::cout << "Status:          " << status << std::endl;

    if (certificate.GetStatus() ==
        Aws::ACM::Model::CertificateStatus::FAILED) {
        Aws::ACM::Model::FailureReason failureReason =
            certificate.GetFailureReason();
        Aws::String reason;

        switch (failureReason) {
            case
Aws::ACM::Model::FailureReason::ADDITIONAL_VERIFICATION_REQUIRED:
                reason = "Additional verification required";
                break;
            case Aws::ACM::Model::FailureReason::CAA_ERROR:
                reason = "CAA error";
                break;
            case Aws::ACM::Model::FailureReason::DOMAIN_NOT_ALLOWED:
                reason = "Domain not allowed";
                break;
            case Aws::ACM::Model::FailureReason::DOMAIN_VALIDATION_DENIED:
                reason = "Domain validation denied";
                break;
            case Aws::ACM::Model::FailureReason::INVALID_PUBLIC_DOMAIN:
                reason = "Invalid public domain";
                break;
            case Aws::ACM::Model::FailureReason::NOT_SET:
                reason = "Not set";
                break;
            case Aws::ACM::Model::FailureReason::NO_AVAILABLE_CONTACTS:
                reason = "No available contacts";
                break;
            case Aws::ACM::Model::FailureReason::OTHER:
                reason = "Other";
                break;
            case Aws::ACM::Model::FailureReason::PCA_ACCESS_DENIED:
                reason = "PCA access denied";
                break;
            case Aws::ACM::Model::FailureReason::PCA_INVALID_ARGS:
                reason = "PCA invalid args";

```

```

        break;
    case Aws::ACM::Model::FailureReason::PCA_INVALID_ARN:
        reason = "PCA invalid ARN";
        break;
    case Aws::ACM::Model::FailureReason::PCA_INVALID_DURATION:
        reason = "PCA invalid duration";
        break;
    case Aws::ACM::Model::FailureReason::PCA_INVALID_STATE:
        reason = "PCA invalid state";
        break;
    case Aws::ACM::Model::FailureReason::PCA_LIMIT_EXCEEDED:
        reason = "PCA limit exceeded";
        break;
    case
Aws::ACM::Model::FailureReason::PCA_NAME_CONSTRAINTS_VALIDATION:
        reason = "PCA name constraints validation";
        break;
    case Aws::ACM::Model::FailureReason::PCA_REQUEST_FAILED:
        reason = "PCA request failed";
        break;
    case Aws::ACM::Model::FailureReason::PCA_RESOURCE_NOT_FOUND:
        reason = "PCA resource not found";
        break;
    default:
        reason = "Cannot determine";
    }

    std::cout << "Failure reason:      " << reason << std::endl;
}

if (certificate.GetStatus() == Aws::ACM::Model::CertificateStatus::REVOKED)
{
    std::cout << "Revoked at (GMT):      " <<
        certificate.GetRevokedAt().ToGmtString(
            Aws::Utils::DateFormat::ISO_8601)
        << std::endl;

    Aws::ACM::Model::RevocationReason revocationReason =
        certificate.GetRevocationReason();
    Aws::String reason;

    switch (revocationReason) {
        case Aws::ACM::Model::RevocationReason::AFFILIATION_CHANGED:
            reason = "Affiliation changed";

```

```

        break;
    case Aws::ACM::Model::RevocationReason::A_A_COMPROMISE:
        reason = "AA compromise";
        break;
    case Aws::ACM::Model::RevocationReason::CA_COMPROMISE:
        reason = "CA compromise";
        break;
    case Aws::ACM::Model::RevocationReason::CERTIFICATE_HOLD:
        reason = "Certificate hold";
        break;
    case Aws::ACM::Model::RevocationReason::CESSATION_OF_OPERATION:
        reason = "Cessation of operation";
        break;
    case Aws::ACM::Model::RevocationReason::KEY_COMPROMISE:
        reason = "Key compromise";
        break;
    case Aws::ACM::Model::RevocationReason::NOT_SET:
        reason = "Not set";
        break;
    case Aws::ACM::Model::RevocationReason::PRIVILEGE_WITHDRAWN:
        reason = "Privilege withdrawn";
        break;
    case Aws::ACM::Model::RevocationReason::REMOVE_FROM_CRL:
        reason = "Revoke from CRL";
        break;
    case Aws::ACM::Model::RevocationReason::SUPERCEDED:
        reason = "Superceded";
        break;
    case Aws::ACM::Model::RevocationReason::UNSPECIFIED:
        reason = "Unspecified";
        break;
    default:
        reason = "Cannot determine";
    }

    std::cout << "Revocation reason:  " << reason << std::endl;
}

if (certificate.GetType() == Aws::ACM::Model::CertificateType::IMPORTED) {
    std::cout << "Imported at (GMT):  " <<
        certificate.GetImportedAt().ToGmtString(
            Aws::Utils::DateFormat::ISO_8601)
        << std::endl;
}

```

```

    Aws::Vector<Aws::String> inUseBys = certificate.GetInUseBy();

    if (!inUseBys.empty()) {
        std::cout << std::endl << "In use by:" << std::endl << std::endl;

        for (auto &in_use_by: inUseBys) {
            std::cout << "  " << in_use_by << std::endl;
        }

        std::cout << std::endl;
    }

    if (certificate.GetType() == Aws::ACM::Model::CertificateType::AMAZON_ISSUED
    &&
        certificate.GetStatus() == Aws::ACM::Model::CertificateStatus::ISSUED) {
        std::cout << "Issued at (GMT):      " <<
            certificate.GetIssuedAt().ToGmtString(
                Aws::Utils::DateFormat::ISO_8601)
            << std::endl;
    }

    std::cout << "Issuer:                " << certificate.GetIssuer() <<
        std::endl;

    Aws::ACM::Model::KeyAlgorithm keyAlgorithm =
        certificate.GetKeyAlgorithm();
    Aws::String algorithm;

    switch (keyAlgorithm) {
        case Aws::ACM::Model::KeyAlgorithm::EC_prime256v1:
            algorithm = "P-256 (secp256r1, prime256v1)";
            break;
        case Aws::ACM::Model::KeyAlgorithm::EC_secp384r1:
            algorithm = "P-384 (secp384r1)";
            break;
        case Aws::ACM::Model::KeyAlgorithm::EC_secp521r1:
            algorithm = "P-521 (secp521r1)";
            break;
        case Aws::ACM::Model::KeyAlgorithm::NOT_SET:
            algorithm = "Not set";
            break;
        case Aws::ACM::Model::KeyAlgorithm::RSA_1024:
            algorithm = "RSA 1024";
    }

```

```

        break;
    case Aws::ACM::Model::KeyAlgorithm::RSA_2048:
        algorithm = "RSA 2048";
        break;
    case Aws::ACM::Model::KeyAlgorithm::RSA_4096:
        algorithm = "RSA 4096";
        break;
    default:
        algorithm = "Cannot determine";
}

std::cout << "Key algorithm:      " << algorithm << std::endl;

if (certificate.GetStatus() == Aws::ACM::Model::CertificateStatus::ISSUED) {
    std::cout << "Not valid after (GMT): " <<
        certificate.GetNotAfter().ToGmtString(
            Aws::Utils::DateFormat::ISO_8601)
        << std::endl;
    std::cout << "Not valid before (GMT): " <<
        certificate.GetNotBefore().ToGmtString(
            Aws::Utils::DateFormat::ISO_8601)
        << std::endl;
}

Aws::ACM::Model::CertificateTransparencyLoggingPreference loggingPreference
=
certificate.GetOptions().GetCertificateTransparencyLoggingPreference();
    Aws::String preference;

    switch (loggingPreference) {
        case
    Aws::ACM::Model::CertificateTransparencyLoggingPreference::DISABLED:
        preference = "Disabled";
        break;
        case Aws::ACM::Model::CertificateTransparencyLoggingPreference::ENABLED:
        preference = "Enabled";
        break;
        case Aws::ACM::Model::CertificateTransparencyLoggingPreference::NOT_SET:
        preference = "Not set";
        break;
        default:
        preference = "Cannot determine";
    }
}

```

```

std::cout << "Logging preference:  " << preference << std::endl;

std::cout << "Serial:                " << certificate.GetSerial() <<
    std::endl;
std::cout << "Signature algorithm: "
    << certificate.GetSignatureAlgorithm() << std::endl;
std::cout << "Subject:                " << certificate.GetSubject() <<
    std::endl;

Aws::ACM::Model::CertificateType certificateType = certificate.GetType();
Aws::String type;

switch (certificateType) {
    case Aws::ACM::Model::CertificateType::AMAZON_ISSUED:
        type = "Amazon issued";
        break;
    case Aws::ACM::Model::CertificateType::IMPORTED:
        type = "Imported";
        break;
    case Aws::ACM::Model::CertificateType::NOT_SET:
        type = "Not set";
        break;
    case Aws::ACM::Model::CertificateType::PRIVATE_:
        type = "Private";
        break;
    default:
        type = "Cannot determine";
}

std::cout << "Type:                " << type << std::endl;

Aws::Vector<Aws::String> altNames =
    certificate.GetSubjectAlternativeNames();

if (!altNames.empty()) {
    std::cout << std::endl << "Alternative names:" <<
        std::endl << std::endl;

    for (auto &alt_name: altNames) {
        std::cout << "  " << alt_name << std::endl;
    }

    std::cout << std::endl;
}

```

```

    }
}

return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DescribeCertificate](#)를 참조하세요.

## ExportCertificate

다음 코드 예시는 ExportCertificate의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Export an AWS Certificate Manager (ACM) certificate.
/*!
    \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
    \param passphrase: A passphrase to decrypt the exported certificate.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */
bool AwsDoc::ACM::exportCertificate(const Aws::String &certificateArn,
                                    const Aws::String &passphrase,
                                    const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::ACM::ACMClient acm_client(clientConfiguration);

    Aws::ACM::Model::ExportCertificateRequest request;
    Aws::Utils::CryptoBuffer cryptoBuffer(
        reinterpret_cast<const unsigned char *>(passphrase.c_str()),
        passphrase.length());
    request.WithCertificateArn(certificateArn).WithPassphrase(cryptoBuffer);

    Aws::ACM::Model::ExportCertificateOutcome outcome =

```

```

        acm_client.ExportCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: ExportCertificate: " <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Success: Information about certificate with ARN '"
            << certificateArn << "':" << std::endl << std::endl;

        auto result = outcome.GetResult();

        std::cout << "Certificate:          " << std::endl << std::endl <<
            result.GetCertificate() << std::endl << std::endl;
        std::cout << "Certificate chain: " << std::endl << std::endl <<
            result.GetCertificateChain() << std::endl << std::endl;
        std::cout << "Private key:          " << std::endl << std::endl <<
            result.GetPrivateKey() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ExportCertificate](#)를 참조하세요.

## GetCertificate

다음 코드 예시는 GetCertificate의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Get an AWS Certificate Manager (ACM) certificate.
/*!
    \param certificateArn: The Amazon Resource Name (ARN) of a certificate.

```

```

    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */
bool AwsDoc::ACM::getCertificate(const Aws::String &certificateArn,
                                const Aws::Client::ClientConfiguration
                                &clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::GetCertificateRequest request;
    request.WithCertificateArn(certificateArn);

    Aws::ACM::Model::GetCertificateOutcome outcome =
        acmClient.GetCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: GetCertificate: " <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Success: Information about certificate with ARN '"
            << certificateArn << "':" << std::endl << std::endl;

        auto result = outcome.GetResult();

        std::cout << "Certificate: " << std::endl << std::endl <<
            result.GetCertificate() << std::endl;
        std::cout << "Certificate chain: " << std::endl << std::endl <<
            result.GetCertificateChain() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetCertificate](#)를 참조하세요.

## ImportCertificate

다음 코드 예시는 ImportCertificate의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Import an AWS Certificate Manager (ACM) certificate.
/*!
    \param certificateFile: Path to certificate to import.
    \param privateKeyFile: Path to file containing a private key.
    \param certificateChainFile: Path to file containing a PEM encoded certificate
    chain.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::ACM::importCertificate(const Aws::String &certificateFile,
                                    const Aws::String &privateKeyFile,
                                    const Aws::String &certificateChainFile,
                                    const Aws::Client::ClientConfiguration
&clientConfiguration) {
    std::ifstream certificateInStream(certificateFile.c_str());
    if (!certificateInStream) {
        std::cerr << "Error: The certificate file '" << certificateFile <<
            "' does not exist." << std::endl;

        return false;
    }

    std::ifstream privateKeyInStream(privateKeyFile.c_str());
    if (!privateKeyInStream) {
        std::cerr << "Error: The private key file '" << privateKeyFile <<
            "' does not exist." << std::endl;

        return false;
    }

    std::ifstream certificateChainInStream(certificateChainFile.c_str());
    if (!certificateChainInStream) {
        std::cerr << "Error: The certificate chain file '"
            << certificateChainFile << "' does not exist." << std::endl;
    }
}

```

```

        return false;
    }

    Aws::String certificate;
    certificate.assign(std::istreambuf_iterator<char>(certificateInStream),
                     std::istreambuf_iterator<char>());

    Aws::String privateKey;
    privateKey.assign(std::istreambuf_iterator<char>(privateKeyInStream),
                     std::istreambuf_iterator<char>());

    Aws::String certificateChain;

    certificateChain.assign(std::istreambuf_iterator<char>(certificateChainInStream),
                           std::istreambuf_iterator<char>());

    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::ImportCertificateRequest request;

    request.WithCertificate(Aws::Utils::ByteBuffer((unsigned char *)
                                                    certificate.c_str(),
                                                    certificate.size()))
           .WithPrivateKey(Aws::Utils::ByteBuffer((unsigned char *)
                                                    privateKey.c_str(),
                                                    privateKey.size()))
           .WithCertificateChain(Aws::Utils::ByteBuffer((unsigned char *)
                                                         certificateChain.c_str(),
                                                         certificateChain.size()));

    Aws::ACM::Model::ImportCertificateOutcome outcome =
        acmClient.ImportCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: ImportCertificate: " <<
            outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
        std::cout << "Success: Certificate associated with ARN '" <<
            outcome.GetResult().GetCertificateArn() << "' imported."

```

```

        << std::endl;

        return true;
    }
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ImportCertificate](#)를 참조하세요.

## ListCertificates

다음 코드 예시는 ListCertificates의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! List the AWS Certificate Manager (ACM) certificates in an account.
/*!
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::ACM::listCertificates(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::ListCertificatesRequest request;
    Aws::Vector<Aws::ACM::Model::CertificateSummary> allCertificates;
    Aws::String nextToken;
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::ACM::Model::ListCertificatesOutcome outcome =
            acmClient.ListCertificates(request);
    } while (outcome.IsSuccess() && nextToken != outcome.GetResult().GetNextToken());
}

```

```

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: ListCertificates: " <<
            outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
        const Aws::ACM::Model::ListCertificatesResult &result =
outcome.GetResult();

        const Aws::Vector<Aws::ACM::Model::CertificateSummary> &certificates =
            result.GetCertificateSummaryList();
        allCertificates.insert(allCertificates.end(), certificates.begin(),
            certificates.end());

        nextToken = result.GetNextToken();
    }
} while (!nextToken.empty());

if (!allCertificates.empty()) {
    for (const Aws::ACM::Model::CertificateSummary &certificate:
allCertificates) {
        std::cout << "Certificate ARN: " <<
            certificate.GetCertificateArn() << std::endl;
        std::cout << "Domain name:      " <<
            certificate.GetDomainName() << std::endl << std::endl;
    }
}
else {
    std::cout << "No available certificates found in account."
        << std::endl;
}

return true;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ListCertificates](#)를 참조하세요.

## ListTagsForCertificate

다음 코드 예시는 ListTagsForCertificate의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! List the tags for an AWS Certificate Manager (ACM) certificate.
/*!
    \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
 */
bool AwsDoc::ACM::listTagsForCertificate(const Aws::String &certificateArn,
                                         const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::ACM::ACMClient acm_client(clientConfiguration);

    Aws::ACM::Model::ListTagsForCertificateRequest request;
    request.WithCertificateArn(certificateArn);

    Aws::ACM::Model::ListTagsForCertificateOutcome outcome =
        acm_client.ListTagsForCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cout << "Error: ListTagsForCertificate: " <<
            outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
        std::cout << "Success: Information about tags for "
            "certificate with ARN '"
            << certificateArn << "':" << std::endl << std::endl;

        auto result = outcome.GetResult();

        Aws::Vector<Aws::ACM::Model::Tag> tags =
            result.GetTags();

        if (tags.size() > 0) {

```

```

        for (const Aws::ACM::Model::Tag &tag: tags) {
            std::cout << "Key:   " << tag.GetKey() << std::endl;
            std::cout << "Value: " << tag.GetValue()
                << std::endl << std::endl;
        }
    }
    else {
        std::cout << "No tags found." << std::endl;
    }

    return true;
}
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ListTagsForCertificate](#)를 참조하세요.

## RemoveTagsFromCertificate

다음 코드 예시는 RemoveTagsFromCertificate의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

///
//! Remove a tag from an ACM certificate.
/*!
    \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
    \param tagKey: The key for the tag.
    \param tagValue: The value for the tag.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */
bool AwsDoc::ACM::removeTagsFromCertificate(const Aws::String &certificateArn,
                                            const Aws::String &tagKey,
                                            const Aws::String &tagValue,
                                            const Aws::Client::ClientConfiguration
&clientConfiguration) {


```

```
Aws::ACM::ACMClient acmClient(clientConfiguration);

Aws::Vector<Aws::ACM::Model::Tag> tags;

Aws::ACM::Model::Tag tag;
tag.SetKey(tagKey);

tags.push_back(tag);

Aws::ACM::Model::RemoveTagsFromCertificateRequest request;
request.WithCertificateArn(certificateArn)
       .WithTags(tags);

Aws::ACM::Model::RemoveTagsFromCertificateOutcome outcome =
    acmClient.RemoveTagsFromCertificate(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Error: RemoveTagFromCertificate: " <<
        outcome.GetError().GetMessage() << std::endl;

    return false;
}
else {
    std::cout << "Success: Tag with key '" << tagKey << "' removed from "
        << "certificate with ARN '" << certificateArn << "'." <<
std::endl;

    return true;
}
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [RemoveTagsFromCertificate](#)를 참조하세요.

## RenewCertificate

다음 코드 예시는 RenewCertificate의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Renew an AWS Certificate Manager (ACM) certificate.
/*!
 \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::ACM::renewCertificate(const Aws::String &certificateArn,
                                   const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::RenewCertificateRequest request;
    request.SetCertificateArn(certificateArn);

    Aws::ACM::Model::RenewCertificateOutcome outcome =
        acmClient.RenewCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: RenewCertificate: " <<
            outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
        std::cout << "Success: Renewed certificate with ARN '"
            << certificateArn << "'." << std::endl;

        return true;
    }
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [RenewCertificate](#)를 참조하세요.

## RequestCertificate

다음 코드 예시는 RequestCertificate의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Request an AWS Certificate Manager (ACM) certificate.
/*!
    \param domainName: A fully qualified domain name.
    \param idempotencyToken: Customer chosen string for idempotency.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::ACM::requestCertificate(const Aws::String &domainName,
                                     const Aws::String &idempotencyToken,
                                     const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::RequestCertificateRequest request;
    request.WithDomainName(domainName)
        .WithIdempotencyToken(idempotencyToken);

    Aws::ACM::Model::RequestCertificateOutcome outcome =
        acmClient.RequestCertificate(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "RequestCertificate error: " <<
            outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
        std::cout << "Success: The newly requested certificate's "
            "ARN is '" <<
            outcome.GetResult().GetCertificateArn() <<
            "'." << std::endl;
    }
}

```

```

        return true;
    }
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [RequestCertificate](#)를 참조하세요.

## ResendValidationEmail

다음 코드 예시는 ResendValidationEmail의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Resend the email that requests domain ownership validation.
/*!
    \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
    \param domainName: A fully qualified domain name.
    \param validationDomain: The base validation domain that will act as the suffix
                           of the email addresses.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::ACM::resendValidationEmail(const Aws::String &certificateArn,
                                         const Aws::String &domainName,
                                         const Aws::String &validationDomain,
                                         const Aws::Client::ClientConfiguration
                                         &clientConfiguration) {
    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::ResendValidationEmailRequest request;
    request.WithCertificateArn(certificateArn)
           .WithDomain(domainName)
           .WithValidationDomain(validationDomain);

    Aws::ACM::Model::ResendValidationEmailOutcome outcome =

```

```

        acmClient.ResendValidationEmail(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "ResendValidationEmail error: " <<
            outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
        std::cout << "Success: The validation email has been resent."
            << std::endl;

        return true;
    }
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ResendValidationEmail](#)을 참조하세요.

## UpdateCertificateOptions

다음 코드 예시는 UpdateCertificateOptions의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Update an AWS Certificate Manager (ACM) certificate option.
/*!
    \param certificateArn: The Amazon Resource Name (ARN) of a certificate.
    \param loggingEnabled: Boolean specifying logging enabled.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::ACM::updateCertificateOption(const Aws::String &certificateArn,
                                          bool loggingEnabled,
                                          const Aws::Client::ClientConfiguration
&clientConfiguration) {

```

```

    Aws::ACM::ACMClient acmClient(clientConfiguration);

    Aws::ACM::Model::UpdateCertificateOptionsRequest request;
    request.SetCertificateArn(certificateArn);

    Aws::ACM::Model::CertificateOptions options;

    if (loggingEnabled) {
        options.SetCertificateTransparencyLoggingPreference(
            Aws::ACM::Model::CertificateTransparencyLoggingPreference::ENABLED);
    }
    else {
        options.SetCertificateTransparencyLoggingPreference(
            Aws::ACM::Model::CertificateTransparencyLoggingPreference::DISABLED);
    }

    request.SetOptions(options);

    Aws::ACM::Model::UpdateCertificateOptionsOutcome outcome =
        acmClient.UpdateCertificateOptions(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "UpdateCertificateOption error: " <<
            outcome.GetError().GetMessage() << std::endl;

        return false;
    }
    else {
        std::cout << "Success: The option '"
            << (loggingEnabled ? "enabled" : "disabled") << "' has been set
for "
                                                                    "the certificate
with the ARN '"
            << certificateArn << "'."
            << std::endl;

        return true;
    }
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [UpdateCertificateOptions](#)을 참조하세요.

## SDK for C++를 사용한 API Gateway 예제

다음 코드 예제에서는 API Gateway와 AWS SDK for C++ 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

사진을 관리하기 위한 서버리스 애플리케이션 만들기

다음 코드 예시에서는 사용자가 레이블을 사용하여 사진을 관리할 수 있는 서버리스 애플리케이션을 생성하는 방법을 보여줍니다.

SDK for C++

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 [AWS 커뮤니티](#)의 게시물을 참조하세요.

이 예제에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

## SDK for C++를 사용한 Aurora 예제

다음 코드 예제에서는 Aurora와 AWS SDK for C++ 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](#)
- [기본 사항](#)
- [작업](#)
- [시나리오](#)

### 시작하기

Hello Aurora

다음 코드 예제에서는 Aurora를 사용하여 시작하는 방법을 보여줍니다.

SDK for C++

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

CMakeLists.txt CMake 파일의 코드입니다.

```
# Set the minimum required version of CMake for this project.
```

```

cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS rds)

# Set this project's name.
project("hello_aurora")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
  for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
      "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
  endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
  # Copy relevant AWS SDK for C++ libraries into the current binary directory for
  running and debugging.

  # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
  may need to uncomment this

  # and set the proper subdirectory to the
  executables' location.

  AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
    ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
  hello_aurora.cpp)

target_link_libraries(${PROJECT_NAME}
  ${AWSSDK_LINK_LIBRARIES})

```

hello\_aurora.cpp 소스 파일의 코드입니다.

```
#include <aws/core/Aws.h>
#include <aws/rds/RDSCClient.h>
#include <aws/rds/model/DescribeDBClustersRequest.h>
#include <iostream>

/*
 * A "Hello Aurora" starter application which initializes an Amazon Relational
 * Database Service (Amazon RDS) client
 * and describes the Amazon Aurora (Aurora) clusters.
 *
 * main function
 *
 * Usage: 'hello_aurora'
 */
int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::RDS::RDSCClient rdsClient(clientConfig);

        Aws::String marker; // Used for pagination.
        std::vector<Aws::String> clusterIds;
        do {
            Aws::RDS::Model::DescribeDBClustersRequest request;

            Aws::RDS::Model::DescribeDBClustersOutcome outcome =
                rdsClient.DescribeDBClusters(request);

            if (outcome.IsSuccess()) {
                for (auto &cluster: outcome.GetResult().GetDBClusters()) {
                    clusterIds.push_back(cluster.GetDBClusterIdentifier());
                }
            }
        } while (marker.empty());
    }
}
```

```

        }
        marker = outcome.GetResult().GetMarker();
    } else {
        result = 1;
        std::cerr << "Error with Aurora::GDescribeDBClusters. "
                  << outcome.GetError().GetMessage()
                  << std::endl;

        break;
    }
} while (!marker.empty());

std::cout << clusterIds.size() << " Aurora clusters found." << std::endl;
for (auto &clusterId: clusterIds) {
    std::cout << "  clusterId " << clusterId << std::endl;
}
}

Aws::ShutdownAPI(options); // Should only be called once.
return 0;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DescribeDBClusters](#)를 참조하세요.

## 기본 사항

### 기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 사용자 지정 Aurora DB 클러스터 파라미터 그룹을 만들고 파라미터 값을 설정합니다.
- 파라미터 그룹을 사용하는 DB 클러스터를 생성합니다.
- 데이터베이스가 포함된 DB 인스턴스를 생성합니다.
- DB 클러스터의 스냅샷을 만든 다음, 리소스를 정리합니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예제 리포지토리](#)에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    //! Routine which creates an Amazon Aurora DB cluster and demonstrates several
    operations
    //! on that cluster.
    /*
    \sa gettingStartedWithDBClusters()
    \param clientConfiguration: AWS client configuration.
    \return bool: Successful completion.
    */
    bool AwsDoc::Aurora::gettingStartedWithDBClusters(
        const Aws::Client::ClientConfiguration &clientConfig) {
        Aws::RDS::RDSClient client(clientConfig);

        printAsterisksLine();
        std::cout << "Welcome to the Amazon Relational Database Service (Amazon Aurora)"
            << std::endl;
        std::cout << "get started with DB clusters demo." << std::endl;
        printAsterisksLine();

        std::cout << "Checking for an existing DB cluster parameter group named '" <<
            CLUSTER_PARAMETER_GROUP_NAME << "'." << std::endl;
        Aws::String dbParameterGroupFamily("Undefined");
        bool parameterGroupFound = true;
        {
            // 1. Check if the DB cluster parameter group already exists.
            Aws::RDS::Model::DescribeDBClusterParameterGroupsRequest request;
            request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);

            Aws::RDS::Model::DescribeDBClusterParameterGroupsOutcome outcome =
                client.DescribeDBClusterParameterGroups(request);
        }
    }

```

```

        if (outcome.IsSuccess()) {
            std::cout << "DB cluster parameter group named '" <<
                CLUSTER_PARAMETER_GROUP_NAME << "' already exists." <<
std::endl;
            dbParameterGroupFamily =
outcome.GetResult().GetDBClusterParameterGroups()[0].GetDBParameterGroupFamily();
        }
        else if (outcome.GetError().GetErrorType() ==
            Aws::RDS::RDSErrors::D_B_PARAMETER_GROUP_NOT_FOUND_FAULT) {
            std::cout << "DB cluster parameter group named '" <<
                CLUSTER_PARAMETER_GROUP_NAME << "' does not exist." <<
std::endl;
            parameterGroupFound = false;
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameterGroups. "
                << outcome.GetError().GetMessage()
                << std::endl;
            return false;
        }
    }

    if (!parameterGroupFound) {
        Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

        // 2. Get available parameter group families for the specified engine.
        if (!getDBEngineVersions(DB_ENGINE, NO_PARAMETER_GROUP_FAMILY,
            engineVersions, client)) {
            return false;
        }

        std::cout << "Getting available parameter group families for " << DB_ENGINE
            << "."
            << std::endl;
        std::vector<Aws::String> families;
        for (const Aws::RDS::Model::DBEngineVersion &version: engineVersions) {
            Aws::String family = version.GetDBParameterGroupFamily();
            if (std::find(families.begin(), families.end(), family) ==
                families.end()) {
                families.push_back(family);
                std::cout << "  " << families.size() << ": " << family << std::endl;
            }
        }
    }

```

```

        int choice = askQuestionForIntRange("Which family do you want to use? ", 1,
                                            static_cast<int>(families.size()));
        dbParameterGroupFamily = families[choice - 1];
    }
    if (!parameterGroupFound) {
        // 3. Create a DB cluster parameter group.
        Aws::RDS::Model::CreateDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetDBParameterGroupFamily(dbParameterGroupFamily);
        request.SetDescription("Example cluster parameter group.");

        Aws::RDS::Model::CreateDBClusterParameterGroupOutcome outcome =
            client.CreateDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully created."
                      << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBClusterParameterGroup. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    printAsterisksLine();
    std::cout << "Let's set some parameter values in your cluster parameter group."
              << std::endl;

    Aws::Vector<Aws::RDS::Model::Parameter> autoIncrementParameters;
    // 4. Get the parameters in the DB cluster parameter group.
    if (!getDBClusterParameters(CLUSTER_PARAMETER_GROUP_NAME, AUTO_INCREMENT_PREFIX,
                               NO_SOURCE,
                               autoIncrementParameters,
                               client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
        return false;
    }

    Aws::Vector<Aws::RDS::Model::Parameter> updateParameters;

    for (Aws::RDS::Model::Parameter &autoIncParameter: autoIncrementParameters) {
        if (autoIncParameter.GetIsModifiable() &&

```

```

        (autoIncParameter.GetDataType() == "integer")) {
            std::cout << "The " << autoIncParameter.GetParameterName()
                << " is described as: " <<
                autoIncParameter.GetDescription() << "." << std::endl;
            if (autoIncParameter.ParameterValueHasBeenSet()) {
                std::cout << "The current value is "
                    << autoIncParameter.GetParameterValue()
                    << "." << std::endl;
            }
            std::vector<int> splitValues = splitToInts(
                autoIncParameter.GetAllowedValues(), '-');
            if (splitValues.size() == 2) {
                int newValue = askQuestionForIntRange(
                    Aws::String("Enter a new value between ") +
                    autoIncParameter.GetAllowedValues() + ": ",
                    splitValues[0], splitValues[1]);
                autoIncParameter.SetParameterValue(std::to_string(newValue));
                updateParameters.push_back(autoIncParameter);
            }
            else {
                std::cerr << "Error parsing " << autoIncParameter.GetAllowedValues()
                    << std::endl;
            }
        }
    }

    {
        // 5. Modify the auto increment parameters in the DB cluster parameter
        group.
        Aws::RDS::Model::ModifyDBClusterParameterGroupRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetParameters(updateParameters);

        Aws::RDS::Model::ModifyDBClusterParameterGroupOutcome outcome =
            client.ModifyDBClusterParameterGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster parameter group was successfully modified."
                << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::ModifyDBClusterParameterGroup. "
                << outcome.GetError().GetMessage()

```

```

        << std::endl;
    }
}

std::cout
    << "You can get a list of parameters you've set by specifying a source
of 'user'."
    << std::endl;

Aws::Vector<Aws::RDS::Model::Parameter> userParameters;
// 6. Display the modified parameters in the DB cluster parameter group.
if (!getDBClusterParameters(CLUSTER_PARAMETER_GROUP_NAME, NO_NAME_PREFIX,
"user",
                           userParameters,
                           client)) {
    cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
    return false;
}

for (const auto &userParameter: userParameters) {
    std::cout << " " << userParameter.GetParameterName() << ", " <<
        userParameter.GetDescription() << ", parameter value - "
        << userParameter.GetParameterValue() << std::endl;
}

printAsterisksLine();
std::cout << "Checking for an existing DB Cluster." << std::endl;

Aws::RDS::Model::DBCluster dbCluster;
// 7. Check if the DB cluster already exists.
if (!describeDBCluster(DB_CLUSTER_IDENTIFIER, dbCluster, client)) {
    cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
    return false;
}

Aws::String engineVersionName;
Aws::String engineName;
if (dbCluster.DBClusterIdentifierHasBeenSet()) {
    std::cout << "The DB cluster already exists." << std::endl;
    engineVersionName = dbCluster.GetEngineVersion();
    engineName = dbCluster.GetEngine();
}
else {

```

```

std::cout << "Let's create a DB cluster." << std::endl;
const Aws::String administratorName = askQuestion(
    "Enter an administrator username for the database: ");
const Aws::String administratorPassword = askQuestion(
    "Enter a password for the administrator (at least 8 characters): ");
Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

// 8. Get a list of engine versions for the parameter group family.
if (!getDBEngineVersions(DB_ENGINE, dbParameterGroupFamily, engineVersions,
    client)) {
    cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
    return false;
}

std::cout << "The available engines for your parameter group family are:"
    << std::endl;

int index = 1;
for (const Aws::RDS::Model::DBEngineVersion &engineVersion: engineVersions)
{
    std::cout << " " << index << ": " << engineVersion.GetEngineVersion()
        << std::endl;
    ++index;
}
int choice = askQuestionForIntRange("Which engine do you want to use? ", 1,
static_cast<int>(engineVersions.size()));
const Aws::RDS::Model::DBEngineVersion engineVersion = engineVersions[choice
-
                                                                    1];

engineName = engineVersion.GetEngine();
engineVersionName = engineVersion.GetEngineVersion();
std::cout << "Creating a DB cluster named '" << DB_CLUSTER_IDENTIFIER
    << "' and database '" << DB_NAME << "'.\n"
    << "The DB cluster is configured to use your custom cluster
parameter group '"
    << CLUSTER_PARAMETER_GROUP_NAME << "', and \n"
    << "selected engine version " << engineVersion.GetEngineVersion()
    << ".\nThis typically takes several minutes." << std::endl;

Aws::RDS::Model::CreateDBClusterRequest request;
request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);

```

```

request.SetEngine(engineName);
request.SetEngineVersion(engineVersionName);
request.SetMasterUsername(administratorName);
request.SetMasterUserPassword(administratorPassword);

Aws::RDS::Model::CreateDBClusterOutcome outcome =
    client.CreateDBCluster(request);

if (outcome.IsSuccess()) {
    std::cout << "The DB cluster creation has started."
               << std::endl;
}
else {
    std::cerr << "Error with Aurora::CreateDBCluster. "
               << outcome.GetError().GetMessage()
               << std::endl;
    cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
    return false;
}
}

std::cout << "Waiting for the DB cluster to become available." << std::endl;

int counter = 0;
// 11. Wait for the DB cluster to become available.
do {
    std::this_thread::sleep_for(std::chrono::seconds(1));
    ++counter;
    if (counter > 900) {
        std::cerr << "Wait for cluster to become available timed out after "
                  << counter
                  << " seconds." << std::endl;
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                        DB_CLUSTER_IDENTIFIER, "", client);
        return false;
    }

    dbCluster = Aws::RDS::Model::DBCluster();
    if (!describeDBCluster(DB_CLUSTER_IDENTIFIER, dbCluster, client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                        DB_CLUSTER_IDENTIFIER, "", client);
        return false;
    }
}

```

```

        if ((counter % 20) == 0) {
            std::cout << "Current DB cluster status is '"
                << dbCluster.GetStatus()
                << "' after " << counter << " seconds." << std::endl;
        }
    } while (dbCluster.GetStatus() != "available");

    if (dbCluster.GetStatus() == "available") {
        std::cout << "The DB cluster has been created." << std::endl;
    }

    printAsterisksLine();
    Aws::RDS::Model::DBInstance dbInstance;
    // 11. Check if the DB instance already exists.
    if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER, "",
            client);
        return false;
    }

    if (dbInstance.DbInstancePortHasBeenSet()) {
        std::cout << "The DB instance already exists." << std::endl;
    }
    else {
        std::cout << "Let's create a DB instance." << std::endl;

        Aws::String dbInstanceClass;
        // 12. Get a list of instance classes.
        if (!chooseDBInstanceClass(engineName,
            engineVersionName,
            dbInstanceClass,
            client)) {
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
                "",
                client);
            return false;
        }

        std::cout << "Creating a DB instance named '" << DB_INSTANCE_IDENTIFIER
            << "' with selected DB instance class '" << dbInstanceClass
            << "'.\nThis typically takes several minutes." << std::endl;

        // 13. Create a DB instance.
        Aws::RDS::Model::CreateDBInstanceRequest request;

```

```

request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
request.SetEngine(engineName);
request.SetDBInstanceClass(dbInstanceClass);

Aws::RDS::Model::CreateDBInstanceOutcome outcome =
    client.CreateDBInstance(request);

if (outcome.IsSuccess()) {
    std::cout << "The DB instance creation has started."
                << std::endl;
}
else {
    std::cerr << "Error with RDS::CreateDBInstance. "
                << outcome.GetError().GetMessage()
                << std::endl;
    cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
        "",
                    client);
    return false;
}
}

std::cout << "Waiting for the DB instance to become available." << std::endl;

counter = 0;
// 14. Wait for the DB instance to become available.
do {
    std::this_thread::sleep_for(std::chrono::seconds(1));
    ++counter;
    if (counter > 900) {
        std::cerr << "Wait for instance to become available timed out after "
                    << counter
                    << " seconds." << std::endl;
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                        DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER, client);
        return false;
    }

    dbInstance = Aws::RDS::Model::DBInstance();
    if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                        DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER, client);
        return false;
    }
}

```

```

    }

    if ((counter % 20) == 0) {
        std::cout << "Current DB instance status is '"
            << dbInstance.GetDBInstanceStatus()
            << "' after " << counter << " seconds." << std::endl;
    }
} while (dbInstance.GetDBInstanceStatus() != "available");

if (dbInstance.GetDBInstanceStatus() == "available") {
    std::cout << "The DB instance has been created." << std::endl;
}

// 15. Display the connection string that can be used to connect a 'mysql' shell
to the database.
displayConnection(dbCluster);

printAsterisksLine();

if (askYesNoQuestion(
    "Do you want to create a snapshot of your DB cluster (y/n)? ") {
    Aws::String snapshotID(DB_CLUSTER_IDENTIFIER + "-" +
        Aws::String(Aws::Utils::UUID::RandomUUID()));
    {
        std::cout << "Creating a snapshot named " << snapshotID << "." <<
std::endl;
        std::cout << "This typically takes a few minutes." << std::endl;

        // 16. Create a snapshot of the DB cluster. (CreateDBClusterSnapshot)
        Aws::RDS::Model::CreateDBClusterSnapshotRequest request;
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetDBClusterSnapshotIdentifier(snapshotID);

        Aws::RDS::Model::CreateDBClusterSnapshotOutcome outcome =
            client.CreateDBClusterSnapshot(request);

        if (outcome.IsSuccess()) {
            std::cout << "Snapshot creation has started."
                << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBClusterSnapshot. "
                << outcome.GetError().GetMessage()
                << std::endl;
        }
    }
}

```

```

        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                        DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
        return false;
    }
}

std::cout << "Waiting for the snapshot to become available." << std::endl;

Aws::RDS::Model::DBClusterSnapshot snapshot;
counter = 0;
do {
    std::this_thread::sleep_for(std::chrono::seconds(1));
    ++counter;
    if (counter > 600) {
        std::cerr << "Wait for snapshot to be available timed out after "
                    << counter
                    << " seconds." << std::endl;
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                        DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
        return false;
    }

    // 17. Wait for the snapshot to become available.
    Aws::RDS::Model::DescribeDBClusterSnapshotsRequest request;
    request.SetDBClusterSnapshotIdentifier(snapshotID);

    Aws::RDS::Model::DescribeDBClusterSnapshotsOutcome outcome =
        client.DescribeDBClusterSnapshots(request);

    if (outcome.IsSuccess()) {
        snapshot = outcome.GetResult().GetDBClusterSnapshots()[0];
    }
    else {
        std::cerr << "Error with Aurora::DescribeDBClusterSnapshots. "
                    << outcome.GetError().GetMessage()
                    << std::endl;
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                        DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
        return false;
    }
}

```

```

        if ((counter % 20) == 0) {
            std::cout << "Current snapshot status is '"
                << snapshot.GetStatus()
                << "' after " << counter << " seconds." << std::endl;
        }
    } while (snapshot.GetStatus() != "available");

    if (snapshot.GetStatus() != "available") {
        std::cout << "A snapshot has been created." << std::endl;
    }
}

printAsterisksLine();

bool result = true;
if (askYesNoQuestion(
    "Do you want to delete the DB cluster, DB instance, and parameter group
(y/n)? ")) {
    result = cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                            DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
                            client);
}

return result;
}

//! Routine which gets a DB cluster description.
/*!
 \sa describeDBCluster()
 \param dbClusterIdentifier: A DB cluster identifier.
 \param clusterResult: The 'DBCluster' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBCluster(const Aws::String &dbClusterIdentifier,
                                       Aws::RDS::Model::DBCluster &clusterResult,
                                       const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBClustersRequest request;
    request.SetDBClusterIdentifier(dbClusterIdentifier);

    Aws::RDS::Model::DescribeDBClustersOutcome outcome =
        client.DescribeDBClusters(request);

    bool result = true;

```

```

    if (outcome.IsSuccess()) {
        clusterResult = outcome.GetResult().GetDBClusters()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
        Aws::RDS::RDSErrors::D_B_CLUSTER_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::GDescribeDBClusters. "
            << outcome.GetError().GetMessage()
            << std::endl;
    }

    // This example does not log an error if the DB cluster does not exist.
    // Instead, clusterResult is set to empty.
    else {
        clusterResult = Aws::RDS::Model::DBCluster();
    }

    return result;
}

//! Routine which gets DB parameters using the 'DescribeDBClusterParameters' api.
/*!
    \sa getDBClusterParameters()
    \param parameterGroupName: The name of the cluster parameter group.
    \param namePrefix: Prefix string to filter results by parameter name.
    \param source: A source such as 'user', ignored if empty.
    \param parametersResult: Vector of 'Parameter' objects returned by the routine.
    \param client: 'RDSClient' instance.
    \return bool: Successful completion.
    */
bool AwsDoc::Aurora::getDBClusterParameters(const Aws::String &parameterGroupName,
                                             const Aws::String &namePrefix,
                                             const Aws::String &source,
                                             Aws::Vector<Aws::RDS::Model::Parameter>
&parametersResult,
                                             const Aws::RDS::RDSClient &client) {
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeDBClusterParametersRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
    }

```

```

        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBClusterParametersOutcome outcome =
            client.DescribeDBClusterParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameters. "
                << outcome.GetError().GetMessage()
                << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}

//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
    \sa getDBEngineVersions()
    \param engineName: A DB engine name.
    \param parameterGroupFamily: A parameter group family name, ignored if empty.
    \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
    routine.
    \param client: 'RDSClient' instance.

```

```

    \return bool: Successful completion.
    */
bool AwsDoc::Aurora::getDBEngineVersions(const Aws::String &engineName,
                                          const Aws::String &parameterGroupFamily,

                                          Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                          const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // The marker is used for pagination.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
            client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersions =
                outcome.GetResult().GetDBEngineVersions();

            engineVersionsResult.insert(engineVersionsResult.end(),
                                       engineVersions.begin(),
                                       engineVersions.end());
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    } while (!marker.empty());

    return true;
}

//! Routine which gets a DB instance description.

```

```

/!!
\sas describeDBCluster()
\param dbInstanceIdentifier: A DB instance identifier.
\param instanceResult: The 'DBInstance' object containing the description.
\param client: 'RDSClient' instance.
\return bool: Successful completion.
*/
bool AwsDoc::Aurora::describeDBInstance(const Aws::String &dbInstanceIdentifier,
                                         Aws::RDS::Model::DBInstance &instanceResult,
                                         const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
        client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::DescribeDBInstances. "
                   << outcome.GetError().GetMessage()
                   << std::endl;
    }
    // This example does not log an error if the DB instance does not exist.
    // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}

/!! Routine which gets available DB instance classes, displays the list
/!! to the user, and returns the user selection.
/!!
\sas chooseDBInstanceClass()
\param engineName: The DB engine name.
\param engineVersion: The DB engine version.
\param dbInstanceClass: String for DB instance class chosen by the user.

```

```

\param client: 'RDSClient' instance.
\return bool: Successful completion.
*/
bool AwsDoc::Aurora::chooseDBInstanceClass(const Aws::String &engine,
                                             const Aws::String &engineVersion,
                                             Aws::String &dbInstanceClass,
                                             const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
            client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption> &options =
                outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option: options)
            {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (std::find(instanceClasses.begin(), instanceClasses.end(),
                             instanceClass) == instanceClasses.end()) {
                    instanceClasses.push_back(instanceClass);
                }
            }
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeOrderableDBInstanceOptions. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << "The available DB instance classes for your database engine are:"
              << std::endl;
    for (int i = 0; i < instanceClasses.size(); ++i) {

```

```

        std::cout << "    " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
        "Which DB instance class do you want to use? ",
        1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}

//! Routine which deletes resources created by the scenario.
/*!
\sa cleanUpResources()
\param parameterGroupName: A parameter group name, this may be empty.
\param dbInstanceIdentifier: A DB instance identifier, this may be empty.
\param client: 'RDSClient' instance.
\return bool: Successful completion.
*/
bool AwsDoc::Aurora::cleanUpResources(const Aws::String &parameterGroupName,
                                       const Aws::String &dbClusterIdentifier,
                                       const Aws::String &dbInstanceIdentifier,
                                       const Aws::RDS::RDSClient &client) {

    bool result = true;
    bool instanceDeleting = false;
    bool clusterDeleting = false;
    if (!dbInstanceIdentifier.empty()) {
        {
            // 18. Delete the DB instance.
            Aws::RDS::Model::DeleteDBInstanceRequest request;
            request.SetDBInstanceIdentifier(dbInstanceIdentifier);
            request.SetSkipFinalSnapshot(true);
            request.SetDeleteAutomatedBackups(true);

            Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                client.DeleteDBInstance(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB instance deletion has started."
                          << std::endl;
                instanceDeleting = true;
                std::cout
                    << "Waiting for DB instance to delete before deleting the
parameter group."
                    << std::endl;
            }
        }
    }
}

```

```

    }
    else {
        std::cerr << "Error with Aurora::DeleteDBInstance. "
                   << outcome.GetError().GetMessage()
                   << std::endl;
        result = false;
    }
}

}

if (!dbClusterIdentifier.empty()) {
    {
        // 19. Delete the DB cluster.
        Aws::RDS::Model::DeleteDBClusterRequest request;
        request.SetDBClusterIdentifier(dbClusterIdentifier);
        request.SetSkipFinalSnapshot(true);

        Aws::RDS::Model::DeleteDBClusterOutcome outcome =
            client.DeleteDBCluster(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB cluster deletion has started."
                      << std::endl;
            clusterDeleting = true;
            std::cout
                << "Waiting for DB cluster to delete before deleting the
parameter group."
                << std::endl;
            std::cout << "This may take a while." << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::DeleteDBCluster. "
                       << outcome.GetError().GetMessage()
                       << std::endl;
            result = false;
        }
    }
}

int counter = 0;

while (clusterDeleting || instanceDeleting) {
    // 20. Wait for the DB cluster and instance to be deleted.
    std::this_thread::sleep_for(std::chrono::seconds(1));
    ++counter;
}

```

```

    if (counter > 800) {
        std::cerr << "Wait for instance to delete timed out after " << counter
                    << " seconds." << std::endl;
        return false;
    }

    Aws::RDS::Model::DBInstance dbInstance = Aws::RDS::Model::DBInstance();
    if (instanceDeleting) {
        if (!describeDBInstance(dbInstanceIdentifier, dbInstance, client)) {
            return false;
        }
        instanceDeleting = dbInstance.DBInstanceIdentifierHasBeenSet();
    }

    Aws::RDS::Model::DBCluster dbCluster = Aws::RDS::Model::DBCluster();
    if (clusterDeleting) {
        if (!describeDBCluster(dbClusterIdentifier, dbCluster, client)) {
            return false;
        }

        clusterDeleting = dbCluster.DBClusterIdentifierHasBeenSet();
    }

    if ((counter % 20) == 0) {
        if (instanceDeleting) {
            std::cout << "Current DB instance status is '"
                      << dbInstance.GetDBInstanceStatus() << "'" << std::endl;
        }

        if (clusterDeleting) {
            std::cout << "Current DB cluster status is '"
                      << dbCluster.GetStatus() << "'" << std::endl;
        }
    }
}

if (!parameterGroupName.empty()) {
    // 21. Delete the DB cluster parameter group.
    Aws::RDS::Model::DeleteDBClusterParameterGroupRequest request;
    request.SetDBClusterParameterGroupName(parameterGroupName);

    Aws::RDS::Model::DeleteDBClusterParameterGroupOutcome outcome =
        client.DeleteDBClusterParameterGroup(request);
}

```

```
    if (outcome.IsSuccess()) {
        std::cout << "The DB parameter group was successfully deleted."
                    << std::endl;
    }
    else {
        std::cerr << "Error with Aurora::DeleteDBClusterParameterGroup. "
                    << outcome.GetError().GetMessage()
                    << std::endl;
        result = false;
    }
}

return result;
}
```

• API 세부 정보는 AWS SDK for C++ API 참조의 다음 항목을 참조하세요.

- [CreateDBCluster](#)
- [CreateDBClusterParameterGroup](#)
- [CreateDBClusterSnapshot](#)
- [CreateDBInstance](#)
- [DeleteDBCluster](#)
- [DeleteDBClusterParameterGroup](#)
- [DeleteDBInstance](#)
- [DescribeDBClusterParameterGroups](#)
- [DescribeDBClusterParameters](#)
- [DescribeDBClusterSnapshots](#)
- [DescribeDBClusters](#)
- [DescribeDBEngineVersions](#)
- [DescribeDBInstances](#)
- [DescribeOrderableDBInstanceOptions](#)
- [ModifyDBClusterParameterGroup](#)

## 작업

### CreateDBCluster

다음 코드 예시는 CreateDBCluster의 사용 방법을 보여줍니다.

SDK for C++

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

Aws::RDS::Model::CreateDBClusterRequest request;
request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
request.SetEngine(engineName);
request.SetEngineVersion(engineVersionName);
request.SetMasterUsername(administratorName);
request.SetMasterUserPassword(administratorPassword);

Aws::RDS::Model::CreateDBClusterOutcome outcome =
    client.CreateDBCluster(request);

if (outcome.IsSuccess()) {
    std::cout << "The DB cluster creation has started."
              << std::endl;
}
else {
    std::cerr << "Error with Aurora::CreateDBCluster. "
              << outcome.GetError().GetMessage()
              << std::endl;
    cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
    return false;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CreateDBCluster](#)를 참조하세요.

## CreateDBClusterParameterGroup

다음 코드 예시는 CreateDBClusterParameterGroup의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

Aws::RDS::Model::CreateDBClusterParameterGroupRequest request;
request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
request.SetDBParameterGroupFamily(dbParameterGroupFamily);
request.SetDescription("Example cluster parameter group.");

Aws::RDS::Model::CreateDBClusterParameterGroupOutcome outcome =
    client.CreateDBClusterParameterGroup(request);

if (outcome.IsSuccess()) {
    std::cout << "The DB cluster parameter group was successfully created."
                << std::endl;
}
else {
    std::cerr << "Error with Aurora::CreateDBClusterParameterGroup. "
                << outcome.GetError().GetMessage()
                << std::endl;
    return false;
}
```

- API 세부 정보는 AWS SDK for C++ SDK for Rust API 참조의 [CreateDBClusterParameterGroup](#)를 참조하세요.

## CreateDBClusterSnapshot

다음 코드 예시는 CreateDBClusterSnapshot의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::CreateDBClusterSnapshotRequest request;
    request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
    request.SetDBClusterSnapshotIdentifier(snapshotID);

    Aws::RDS::Model::CreateDBClusterSnapshotOutcome outcome =
        client.CreateDBClusterSnapshot(request);

    if (outcome.IsSuccess()) {
        std::cout << "Snapshot creation has started."
                  << std::endl;
    }
    else {
        std::cerr << "Error with Aurora::CreateDBClusterSnapshot. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                        DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
        return false;
    }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CreateDBClusterSnapshot](#)을 참조하세요.

## CreateDBInstance

다음 코드 예시는 CreateDBInstance의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

Aws::RDS::Model::CreateDBInstanceRequest request;
request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
request.SetEngine(engineName);
request.SetDBInstanceClass(dbInstanceClass);

Aws::RDS::Model::CreateDBInstanceOutcome outcome =
    client.CreateDBInstance(request);

if (outcome.IsSuccess()) {
    std::cout << "The DB instance creation has started."
                << std::endl;
}
else {
    std::cerr << "Error with RDS::CreateDBInstance. "
                << outcome.GetError().GetMessage()
                << std::endl;
    cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
        "",
        client);
}
```

```
    return false;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CreateDBInstance](#)를 참조하세요.

## DeleteDBCluster

다음 코드 예시는 DeleteDBCluster의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::DeleteDBClusterRequest request;
    request.SetDBClusterIdentifier(dbClusterIdentifier);
    request.SetSkipFinalSnapshot(true);

    Aws::RDS::Model::DeleteDBClusterOutcome outcome =
        client.DeleteDBCluster(request);

    if (outcome.IsSuccess()) {
        std::cout << "DB cluster deletion has started."
                    << std::endl;
        clusterDeleting = true;
        std::cout
            << "Waiting for DB cluster to delete before deleting the
parameter group."
            << std::endl;
        std::cout << "This may take a while." << std::endl;
    }
    else {
```

```

        std::cerr << "Error with Aurora::DeleteDBCluster. "
                    << outcome.GetError().GetMessage()
                    << std::endl;
        result = false;
    }

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteDBCluster](#)를 참조하세요.

## DeleteDBClusterParameterGroup

다음 코드 예시는 DeleteDBClusterParameterGroup의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

Aws::RDS::Model::DeleteDBClusterParameterGroupRequest request;
request.SetDBClusterParameterGroupName(parameterGroupName);

Aws::RDS::Model::DeleteDBClusterParameterGroupOutcome outcome =
    client.DeleteDBClusterParameterGroup(request);

if (outcome.IsSuccess()) {
    std::cout << "The DB parameter group was successfully deleted."
                << std::endl;
}
else {
    std::cerr << "Error with Aurora::DeleteDBClusterParameterGroup. "
                << outcome.GetError().GetMessage()
                << std::endl;
    result = false;
}

```

```
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteDBClusterParameterGroup](#)을 참조하세요.

## DeleteDBInstance

다음 코드 예시는 DeleteDBInstance의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::DeleteDBInstanceRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);
    request.SetSkipFinalSnapshot(true);
    request.SetDeleteAutomatedBackups(true);

    Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
        client.DeleteDBInstance(request);

    if (outcome.IsSuccess()) {
        std::cout << "DB instance deletion has started."
                    << std::endl;
        instanceDeleting = true;
        std::cout
            << "Waiting for DB instance to delete before deleting the
parameter group."
            << std::endl;
    }
```

```

    else {
        std::cerr << "Error with Aurora::DeleteDBInstance. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        result = false;
    }

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteDBInstance](#)를 참조하세요.

## DescribeDBClusterParameterGroups

다음 코드 예시는 DescribeDBClusterParameterGroups의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

Aws::RDS::Model::DescribeDBClusterParameterGroupsRequest request;
request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);

Aws::RDS::Model::DescribeDBClusterParameterGroupsOutcome outcome =
    client.DescribeDBClusterParameterGroups(request);

if (outcome.IsSuccess()) {
    std::cout << "DB cluster parameter group named '" <<
                CLUSTER_PARAMETER_GROUP_NAME << "' already exists." <<
    std::endl;
    dbParameterGroupFamily =
outcome.GetResult().GetDBClusterParameterGroups()[0].GetDBParameterGroupFamily();
}

```

```

    else {
        std::cerr << "Error with Aurora::DescribeDBClusterParameterGroups. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        return false;
    }

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DescribeDBClusterParameterGroups](#)를 참조하세요.

## DescribeDBClusterParameters

다음 코드 예시는 DescribeDBClusterParameters의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

    //! Routine which gets DB parameters using the 'DescribeDBClusterParameters' api.
    /*
    \sa getDBClusterParameters()
    \param parameterGroupName: The name of the cluster parameter group.
    \param namePrefix: Prefix string to filter results by parameter name.
    \param source: A source such as 'user', ignored if empty.
    \param parametersResult: Vector of 'Parameter' objects returned by the routine.
    \param client: 'RDSClient' instance.
    \return bool: Successful completion.
    */
    bool AwsDoc::Aurora::getDBClusterParameters(const Aws::String &parameterGroupName,

```

```

const Aws::String &namePrefix,
const Aws::String &source,
Aws::Vector<Aws::RDS::Model::Parameter>
&parametersResult,

const Aws::RDS::RDSClient &client) {
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeDBClusterParametersRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBClusterParametersOutcome outcome =
            client.DescribeDBClusterParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBClusterParameters. "
                << outcome.GetError().GetMessage()
                << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}

```

```
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DescribeDBClusterParameters](#)를 참조하세요.

## DescribeDBClusterSnapshots

다음 코드 예시는 DescribeDBClusterSnapshots의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::DescribeDBClusterSnapshotsRequest request;
    request.SetDBClusterSnapshotIdentifier(snapshotID);

    Aws::RDS::Model::DescribeDBClusterSnapshotsOutcome outcome =
        client.DescribeDBClusterSnapshots(request);

    if (outcome.IsSuccess()) {
        snapshot = outcome.GetResult().GetDBClusterSnapshots()[0];
    }
    else {
        std::cerr << "Error with Aurora::DescribeDBClusterSnapshots. "
            << outcome.GetError().GetMessage()
            << std::endl;
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
            DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
        return false;
    }

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DescribeDBClusterSnapshots](#)를 참조하세요.

## DescribeDBClusters

다음 코드 예시는 DescribeDBClusters의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

//! Routine which gets a DB cluster description.
/*!
 \sa describeDBCluster()
 \param dbClusterIdentifier: A DB cluster identifier.
 \param clusterResult: The 'DBCluster' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBCluster(const Aws::String &dbClusterIdentifier,
                                       Aws::RDS::Model::DBCluster &clusterResult,
                                       const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBClustersRequest request;
    request.SetDBClusterIdentifier(dbClusterIdentifier);

    Aws::RDS::Model::DescribeDBClustersOutcome outcome =
        client.DescribeDBClusters(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        clusterResult = outcome.GetResult().GetDBClusters()[0];
    }
}
```

```

    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_CLUSTER_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::GDescribeDBClusters. "
                   << outcome.GetError().GetMessage()
                   << std::endl;
    }

    // This example does not log an error if the DB cluster does not exist.
    // Instead, clusterResult is set to empty.
    else {
        clusterResult = Aws::RDS::Model::DBCluster();
    }

    return result;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DescribeDBClusters](#)를 참조하세요.

## DescribeDBEngineVersions

다음 코드 예시는 DescribeDBEngineVersions의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

    //! Routine which gets available DB engine versions for an engine name and
    //! an optional parameter group family.

```

```

/ * !
\sa getDBEngineVersions()
\param engineName: A DB engine name.
\param parameterGroupFamily: A parameter group family name, ignored if empty.
\param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
routine.
\param client: 'RDSClient' instance.
\return bool: Successful completion.
*/
bool AwsDoc::Aurora::getDBEngineVersions(const Aws::String &engineName,
                                         const Aws::String &parameterGroupFamily,

                                         Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                         const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // The marker is used for pagination.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
            client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersions =
                outcome.GetResult().GetDBEngineVersions();

            engineVersionsResult.insert(engineVersionsResult.end(),
                                       engineVersions.begin(),
                                       engineVersions.end());
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBEngineVersionsRequest. "
                       << outcome.GetError().GetMessage()
                       << std::endl;
        }
    }
}

```

```

    } while (!marker.empty());

    return true;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DescribeDBEngineVersions](#)을 참조하세요.

## DescribeDBInstances

다음 코드 예시는 DescribeDBInstances의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

    //! Routine which gets a DB instance description.
    /*!
    \sa describeDBCluster()
    \param dbInstanceIdIdentifier: A DB instance identifier.
    \param instanceResult: The 'DBInstance' object containing the description.
    \param client: 'RDSClient' instance.
    \return bool: Successful completion.
    */
    bool AwsDoc::Aurora::describeDBInstance(const Aws::String &dbInstanceIdIdentifier,
                                             Aws::RDS::Model::DBInstance &instanceResult,
                                             const Aws::RDS::RDSClient &client) {
        Aws::RDS::Model::DescribeDBInstancesRequest request;
        request.SetDBInstanceIdIdentifier(dbInstanceIdIdentifier);

        Aws::RDS::Model::DescribeDBInstancesOutcome outcome =

```

```

        client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
        Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::DescribeDBInstances. "
            << outcome.GetError().GetMessage()
            << std::endl;
    }

    // This example does not log an error if the DB instance does not exist.
    // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DescribeDBInstances](#)를 참조하세요.

## DescribeOrderableDBInstanceOptions

다음 코드 예시는 DescribeOrderableDBInstanceOptions의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

```

```

//! Routine which gets available DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
\sa chooseDBInstanceClass()
\param engineName: The DB engine name.
\param engineVersion: The DB engine version.
\param dbInstanceClass: String for DB instance class chosen by the user.
\param client: 'RDSClient' instance.
\return bool: Successful completion.
*/
bool AwsDoc::Aurora::chooseDBInstanceClass(const Aws::String &engine,
                                           const Aws::String &engineVersion,
                                           Aws::String &dbInstanceClass,
                                           const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
            client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption> &options =
                outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option: options)
            {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (std::find(instanceClasses.begin(), instanceClasses.end(),
                             instanceClass) == instanceClasses.end()) {
                    instanceClasses.push_back(instanceClass);
                }
            }
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeOrderableDBInstanceOptions. "

```

```

        << outcome.GetError().GetMessage()
        << std::endl;
    return false;
}
} while (!marker.empty());

std::cout << "The available DB instance classes for your database engine are:"
    << std::endl;
for (int i = 0; i < instanceClasses.size(); ++i) {
    std::cout << "    " << i + 1 << ": " << instanceClasses[i] << std::endl;
}

int choice = askQuestionForIntRange(
    "Which DB instance class do you want to use? ",
    1, static_cast<int>(instanceClasses.size()));
dbInstanceClass = instanceClasses[choice - 1];
return true;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DescribeOrderableDBInstanceOptions](#)를 참조하세요.

## ModifyDBClusterParameterGroup

다음 코드 예시는 ModifyDBClusterParameterGroup의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

```

```

Aws::RDS::Model::ModifyDBClusterParameterGroupRequest request;
request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
request.SetParameters(updateParameters);

Aws::RDS::Model::ModifyDBClusterParameterGroupOutcome outcome =
    client.ModifyDBClusterParameterGroup(request);

if (outcome.IsSuccess()) {
    std::cout << "The DB cluster parameter group was successfully modified."
              << std::endl;
}
else {
    std::cerr << "Error with Aurora::ModifyDBClusterParameterGroup. "
              << outcome.GetError().GetMessage()
              << std::endl;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ModifyDBClusterParameterGroup](#)을 참조하세요.

## 시나리오

### Aurora 서버리스 작업 항목 트래커 만들기

다음 코드 예제에서는 Amazon Aurora Serverless 데이터베이스에서 작업 항목을 추적하고 Amazon Simple Email Service(Amazon SES)를 사용하여 보고서를 보내는 웹 애플리케이션을 만드는 방법을 보여줍니다.

#### SDK for C++

Amazon Aurora Serverless 데이터베이스에 저장된 작업 항목을 추적하고 보고하는 웹 애플리케이션을 만드는 방법을 보여줍니다.

Amazon Aurora Serverless 데이터를 쿼리하고 React 애플리케이션에서 사용하도록 C++ REST API를 설정하는 방법에 대한 지침과 전체 소스 코드는 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Aurora
- Amazon RDS
- Amazon RDS 데이터 서비스

- Amazon SES

## SDK for C++를 사용한 Auto Scaling 예제

다음 코드 예제에서는 Auto Scaling과 AWS SDK for C++ 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

Auto Scaling 시작

다음 코드 예제에서는 Auto Scaling 사용을 시작하는 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

CMakeLists.txt CMake 파일의 코드입니다.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)
```

```
# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS autoscaling)

# Set this project's name.
project("hello_autoscaling")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
"${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory for
    running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
    may need to uncomment this

                                # and set the proper subdirectory to the
    executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
"${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR}")
endif ()

add_executable(${PROJECT_NAME}
    hello_autoscaling.cpp)

target_link_libraries(${PROJECT_NAME}
    ${AWSSDK_LINK_LIBRARIES})
```

hello\_autoscaling.cpp 소스 파일의 코드입니다.

```
#include <aws/core/Aws.h>
#include <aws/autoscaling/AutoScalingClient.h>
#include <aws/autoscaling/model/DescribeAutoScalingGroupsRequest.h>
#include <iostream>

/*
 * A "Hello Autoscaling" starter application which initializes an Amazon EC2 Auto
 * Scaling client and describes the
 * Amazon EC2 Auto Scaling groups.
 *
 * main function
 *
 * Usage: 'hello_autoscaling'
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::AutoScaling::AutoScalingClient autoscalingClient(clientConfig);

        std::vector<Aws::String> groupNames;
        Aws::String nextToken; // Used for pagination.

        do {

            Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }

            Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
                autoscalingClient.DescribeAutoScalingGroups(request);
```

```

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup>
&autoScalingGroups =
                outcome.GetResult().GetAutoScalingGroups();
            for (auto &group: autoScalingGroups) {
                groupNames.push_back(group.GetAutoScalingGroupName());
            }
            nextToken = outcome.GetResult().GetNextToken();
        } else {
            std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups. "
                << outcome.GetError().GetMessage()
                << std::endl;
            result = 1;
            break;
        }
    } while (!nextToken.empty());

    std::cout << "Found " << groupNames.size() << " AutoScaling groups." <<
std::endl;
    for (auto &groupName: groupNames) {
        std::cout << "AutoScaling group: " << groupName << std::endl;
    }

}

Aws::ShutdownAPI(options); // Should only be called once.
return result;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DescribeAutoScalingGroups](#)을 참조하세요.

## 기본 사항

### 기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 시작 템플릿과 가용 영역이 있는 Amazon EC2 Auto Scaling 그룹을 생성하고 실행 중인 인스턴스에 대한 정보를 가져옵니다.
- Amazon CloudWatch 지표 수집 활성화

- 그룹의 원하는 용량을 업데이트하고 인스턴스가 시작될 때까지 기다립니다.
- 그룹에서 인스턴스를 종료합니다.
- 사용자 요청 및 용량 변경에 따라 발생하는 조정 활동을 나열합니다.
- CloudWatch 지표에 대한 통계를 가져온 다음 리소스를 정리합니다.

## SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예제 리포지토리](#)에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Routine which demonstrates using an Auto Scaling group
//! to manage Amazon EC2 instances.
/*!
 \sa groupsAndInstancesScenario()
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::groupsAndInstancesScenario(
    const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::String templateName;
    Aws::EC2::EC2Client ec2Client(clientConfig);

    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " "
        << std::endl;
    std::cout
        << "Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) Auto
Scaling "
        << "demo for managing groups and instances." << std::endl;
    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " \n"
        << std::endl;

    std::cout << "This example requires an EC2 launch template." << std::endl;
    if (askYesNoQuestion(
        "Would you like to use an existing EC2 launch template (y/n)? ")) {

        // 1. Specify the name of an existing EC2 launch template.
        templateName = askQuestion(

```

```

        "Enter the name of the existing EC2 launch template.  ");

    Aws::EC2::Model::DescribeLaunchTemplatesRequest request;
    request.AddLaunchTemplateName(templateName);
    Aws::EC2::Model::DescribeLaunchTemplatesOutcome outcome =
        ec2Client.DescribeLaunchTemplates(request);

    if (outcome.IsSuccess()) {
        std::cout << "Validated the EC2 launch template '" << templateName
                    << "' exists by calling DescribeLaunchTemplate." << std::endl;
    }
    else {
        std::cerr << "Error validating the existence of the launch template. "
                    << outcome.GetError().GetMessage()
                    << std::endl;
    }
}
else { // 2.  Or create a new EC2 launch template.
    templateName = askQuestion("Enter the name for a new EC2 launch template:
");

    Aws::EC2::Model::CreateLaunchTemplateRequest request;
    request.SetLaunchTemplateName(templateName);

    Aws::EC2::Model::RequestLaunchTemplateData requestLaunchTemplateData;

    requestLaunchTemplateData.SetInstanceType(EC2_LAUNCH_TEMPLATE_INSTANCE_TYPE);
    requestLaunchTemplateData.SetImageId(EC2_LAUNCH_TEMPLATE_IMAGE_ID);

    request.SetLaunchTemplateData(requestLaunchTemplateData);

    Aws::EC2::Model::CreateLaunchTemplateOutcome outcome =
        ec2Client.CreateLaunchTemplate(request);

    if (outcome.IsSuccess()) {
        std::cout << "The EC2 launch template '" << templateName << " was
created."
                    << std::endl;
    }
    else if (outcome.GetError().GetExceptionName() ==
        "InvalidLaunchTemplateName.AlreadyExistsException") {
        std::cout << "The EC2 template '" << templateName << "' already exists"
                    << std::endl;
    }
}

```

```

        else {
            std::cerr << "Error with EC2::CreateLaunchTemplate. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }
    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);
    std::cout << "Let's create an Auto Scaling group." << std::endl;
    Aws::String groupName = askQuestion(
        "Enter a name for the Auto Scaling group: ");
    // 3. Retrieve a list of EC2 Availability Zones.
    Aws::Vector<Aws::EC2::Model::AvailabilityZone> availabilityZones;
    {
        Aws::EC2::Model::DescribeAvailabilityZonesRequest request;

        Aws::EC2::Model::DescribeAvailabilityZonesOutcome outcome =
            ec2Client.DescribeAvailabilityZones(request);

        if (outcome.IsSuccess()) {
            std::cout
                << "EC2 instances can be created in the following Availability
Zones:"
                << std::endl;

            availabilityZones = outcome.GetResult().GetAvailabilityZones();
            for (size_t i = 0; i < availabilityZones.size(); ++i) {
                std::cout << "    " << i + 1 << ". "
                          << availabilityZones[i].GetZoneName() << std::endl;
            }
        }
        else {
            std::cerr << "Error with EC2::DescribeAvailabilityZones. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanupResources("", templateName, autoScalingClient, ec2Client);
            return false;
        }
    }

    int availabilityZoneChoice = askQuestionForIntRange(
        "Choose an Availability Zone: ", 1,
        static_cast<int>(availabilityZones.size()));
    // 4. Create an Auto Scaling group with the specified Availability Zone.
    {

```

```

    Aws::AutoScaling::Model::CreateAutoScalingGroupRequest request;
    request.SetAutoScalingGroupName(groupName);
    Aws::Vector<Aws::String> availabilityGroupZones;
    availabilityGroupZones.push_back(
        availabilityZones[availabilityZoneChoice - 1].GetZoneName());
    request.SetAvailabilityZones(availabilityGroupZones);
    request.SetMaxSize(1);
    request.SetMinSize(1);

    Aws::AutoScaling::Model::LaunchTemplateSpecification
launchTemplateSpecification;
    launchTemplateSpecification.SetLaunchTemplateName(templateName);
    request.SetLaunchTemplate(launchTemplateSpecification);

    Aws::AutoScaling::Model::CreateAutoScalingGroupOutcome outcome =
        autoScalingClient.CreateAutoScalingGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "Created Auto Scaling group '" << groupName << "'..."
        << std::endl;
    }
    else if (outcome.GetError().GetErrorType() ==
        Aws::AutoScaling::AutoScalingErrors::ALREADY_EXISTS_FAULT) {
        std::cout << "Auto Scaling group '" << groupName << "' already exists."
        << std::endl;
    }
    else {
        std::cerr << "Error with AutoScaling::CreateAutoScalingGroup. "
        << outcome.GetError().GetMessage()
        << std::endl;
        cleanupResources("", templateName, autoScalingClient, ec2Client);
        return false;
    }
}

Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> autoScalingGroups;
if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
        autoScalingClient)) {
    std::cout << "Here is the Auto Scaling group description." << std::endl;
    if (!autoScalingGroups.empty()) {
        logAutoScalingGroupInfo(autoScalingGroups);
    }
}
else {

```

```

        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }

    std::cout
        << "Waiting for the EC2 instance in the Auto Scaling group to become
active..."
        << std::endl;
    if (!waitForInstances(groupName, autoScalingGroups, autoScalingClient)) {
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }

    bool enableMetrics = askYesNoQuestion(
        "Do you want to collect metrics about the A"
        "Auto Scaling group during this demo (y/n)? ");
    // 7. Optionally enable metrics collection for the Auto Scaling group.
    if (enableMetrics) {
        Aws::AutoScaling::Model::EnableMetricsCollectionRequest request;
        request.SetAutoScalingGroupName(groupName);

        request.AddMetrics("GroupMinSize");
        request.AddMetrics("GroupMaxSize");
        request.AddMetrics("GroupDesiredCapacity");
        request.AddMetrics("GroupInServiceInstances");
        request.AddMetrics("GroupTotalInstances");
        request.SetGranularity("1Minute");

        Aws::AutoScaling::Model::EnableMetricsCollectionOutcome outcome =
            autoScalingClient.EnableMetricsCollection(request);
        if (outcome.IsSuccess()) {
            std::cout << "Auto Scaling metrics have been enabled."
                << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::EnableMetricsCollection. "
                << outcome.GetError().GetMessage()
                << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
            return false;
        }
    }
}

```

```

    std::cout << "Let's update the maximum number of EC2 instances in '" <<
groupName <<
    "' from 1 to 3." << std::endl;
    askQuestion("Press enter to continue: ", alwaysTrueTest);
    // 8. Update the Auto Scaling group, setting a new maximum size.
    {
        Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
        request.SetAutoScalingGroupName(groupName);
        request.SetMaxSize(3);

        Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
            autoScalingClient.UpdateAutoScalingGroup(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                << outcome.GetError().GetMessage()
                << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
            return false;
        }
    }

    if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
        autoScalingClient)) {
        if (!autoScalingGroups.empty()) {
            const auto &instances = autoScalingGroups[0].GetInstances();
            std::cout
                << "The group still has one running EC2 instance, but it can
have up to 3.\n"
                << std::endl;
            logAutoScalingGroupInfo(autoScalingGroups);
        }
        else {
            std::cerr
                << "No EC2 launch groups were retrieved from DescribeGroup
request."
                << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
            return false;
        }
    }

    std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << "\n"
        << std::endl;

```

```

std::cout << "Let's update the desired capacity in '" << groupName <<
    "' from 1 to 2." << std::endl;
askQuestion("Press enter to continue: ", alwaysTrueTest);
// 9. Update the Auto Scaling group, setting a new desired capacity.
{
    Aws::AutoScaling::Model::SetDesiredCapacityRequest request;
    request.SetAutoScalingGroupName(groupName);
    request.SetDesiredCapacity(2);

    Aws::AutoScaling::Model::SetDesiredCapacityOutcome outcome =
        autoScalingClient.SetDesiredCapacity(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error with AutoScaling::SetDesiredCapacityRequest. "
            << outcome.GetError().GetMessage()
            << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }
}

if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
    autoScalingClient)) {
    if (!autoScalingGroups.empty()) {
        std::cout
            << "Here is the current state of the group." << std::endl;
        logAutoScalingGroupInfo(autoScalingGroups);
    }
    else {
        std::cerr
            << "No EC2 launch groups were retrieved from DescribeGroup
request."
            << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }
}

std::cout << "Waiting for the new EC2 instance to start..." << std::endl;
waitForInstances(groupName, autoScalingGroups, autoScalingClient);

std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << "\n"
    << std::endl;

```

```

std::cout << "Let's terminate one of the EC2 instances in " << groupName << "."
    << std::endl;
std::cout << "Because the desired capacity is 2, another EC2 instance will start
"
    << "to replace the terminated EC2 instance."
    << std::endl;
std::cout << "The currently running EC2 instances are:" << std::endl;

if (autoScalingGroups.empty()) {
    std::cerr << "Error describing groups. No groups returned." << std::endl;
    cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
    return false;
}

int instanceNumber = 1;
Aws::Vector<Aws::String> instanceIDs = instancesToInstanceIDs(
    autoScalingGroups[0].GetInstances());
for (const Aws::String &instanceID: instanceIDs) {
    std::cout << "    " << instanceNumber << ". " << instanceID << std::endl;
    ++instanceNumber;
}

instanceNumber = askQuestionForIntRange("Which EC2 instance do you want to stop?
",
                                     1,
                                     static_cast<int>(instanceIDs.size()));

// 10. Terminate an EC2 instance in the Auto Scaling group.
{
    Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest request;
    request.SetInstanceId(instanceIDs[instanceNumber - 1]);
    request.SetShouldDecrementDesiredCapacity(false);

    Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome outcome
=
    autoScalingClient.TerminateInstanceInAutoScalingGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "Waiting for EC2 instance with ID '"
            << instanceIDs[instanceNumber - 1] << "' to terminate..."
            << std::endl;
    }
    else {

```

```

        std::cerr << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                << outcome.GetError().GetMessage()
                << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }
}

waitForInstances(groupName, autoScalingGroups, autoScalingClient);

std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << "\n"
        << std::endl;
std::cout << "Let's get a report of scaling activities for EC2 launch group '"
        << groupName << "'."
        << std::endl;
askQuestion("Press enter to continue: ", alwaysTrueTest);
// 11. Get a description of activities for the Auto Scaling group.
{
    Aws::AutoScaling::Model::DescribeScalingActivitiesRequest request;
    request.SetAutoScalingGroupName(groupName);

    Aws::Vector<Aws::AutoScaling::Model::Activity> allActivities;
    Aws::String nextToken; // Used for pagination;
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }
        Aws::AutoScaling::Model::DescribeScalingActivitiesOutcome outcome =
            autoScalingClient.DescribeScalingActivities(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::AutoScaling::Model::Activity> &activities =
                outcome.GetResult().GetActivities();
            allActivities.insert(allActivities.end(), activities.begin(),
activities.end());
            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cerr << "Error with AutoScaling::DescribeScalingActivities. "
                    << outcome.GetError().GetMessage()
                    << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);

```

```

        return false;
    }
} while (!nextToken.empty());

std::cout << "Found " << allActivities.size() << " activities."
    << std::endl;
std::cout << "Activities are ordered with the most recent first."
    << std::endl;
for (const Aws::AutoScaling::Model::Activity &activity: allActivities) {
    std::cout << activity.GetDescription() << std::endl;
    std::cout << activity.GetDetails() << std::endl;
}
}

if (enableMetrics) {
    if (!logAutoScalingMetrics(groupName, clientConfig)) {
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }
}

std::cout << "Let's clean up." << std::endl;
askQuestion("Press enter to continue: ", alwaysTrueTest);

// 13. Disable metrics collection if enabled.
if (enableMetrics) {
    Aws::AutoScaling::Model::DisableMetricsCollectionRequest request;
    request.SetAutoScalingGroupName(groupName);

    Aws::AutoScaling::Model::DisableMetricsCollectionOutcome outcome =
        autoScalingClient.DisableMetricsCollection(request);

    if (outcome.IsSuccess()) {
        std::cout << "Metrics collection has been disabled." << std::endl;
    }
    else {
        std::cerr << "Error with AutoScaling::DisableMetricsCollection. "
            << outcome.GetError().GetMessage()
            << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }
}
}

```

```

    return cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
}

//! Routine which waits for EC2 instances in an Auto Scaling group to
//! complete startup or shutdown.
//!
\sa waitForInstances()
\param groupName: An Auto Scaling group name.
\param autoScalingGroups: Vector to receive 'AutoScalingGroup' records.
\param client: 'AutoScalingClient' instance.
\return bool: Successful completion.
*/
bool AwsDoc::AutoScaling::waitForInstances(const Aws::String &groupName,

    Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> &autoScalingGroups,
    const Aws::AutoScaling::AutoScalingClient
&client) {
    bool ready = false;
    const std::vector<Aws::String> READY_STATES = {"InService", "Terminated"};

    int count = 0;
    int desiredCapacity = 0;
    std::this_thread::sleep_for(std::chrono::seconds(4));
    while (!ready) {
        if (WAIT_FOR_INSTANCES_TIMEOUT < count) {
            std::cerr << "Wait for instance timed out." << std::endl;
            return false;
        }

        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++count;
        if (!describeGroup(groupName, autoScalingGroups, client)) {
            return false;
        }
        Aws::Vector<Aws::String> instanceIDs;
        if (!autoScalingGroups.empty()) {
            instanceIDs =
instancesToInstanceIDs(autoScalingGroups[0].GetInstances());
            desiredCapacity = autoScalingGroups[0].GetDesiredCapacity();
        }

        if (instanceIDs.empty()) {
            if (desiredCapacity == 0) {
                break;
            }
        }
    }
}

```

```

    }
    else {
        if ((count % 5) == 0) {
            std::cout << "No instance IDs returned for group." << std::endl;
        }

        continue;
    }
}

// 6. Check lifecycle state of the instances using
DescribeAutoScalingInstances.
Aws::AutoScaling::Model::DescribeAutoScalingInstancesRequest request;
request.SetInstanceIds(instanceIDs);

Aws::AutoScaling::Model::DescribeAutoScalingInstancesOutcome outcome =
    client.DescribeAutoScalingInstances(request);

if (outcome.IsSuccess()) {
    const Aws::Vector<Aws::AutoScaling::Model::AutoScalingInstanceDetails>
&instancesDetails =
        outcome.GetResult().GetAutoScalingInstances();
    ready = instancesDetails.size() >= desiredCapacity;
    for (const Aws::AutoScaling::Model::AutoScalingInstanceDetails &details:
instancesDetails) {
        if (!stringInVector(details.GetLifecycleState(), READY_STATES)) {
            ready = false;
            break;
        }
    }
    // Log the status while waiting.
    if (((count % 5) == 1) || ready) {
        logInstancesLifecycleState(instancesDetails);
    }
}
else {
    std::cerr << "Error with AutoScaling::DescribeAutoScalingInstances. "
        << outcome.GetError().GetMessage()
        << std::endl;
    return false;
}
}

if (!describeGroup(groupName, autoScalingGroups, client)) {

```

```

        return false;
    }

    return true;
}

//! Routine to cleanup resources created in 'groupsAndInstancesScenario'.
/*!
\sa cleanupResources()
\param groupName: Optional Auto Scaling group name.
\param templateName: Optional EC2 launch template name.
\param autoScalingClient: 'AutoScalingClient' instance.
\param ec2Client: 'EC2Client' instance.
\return bool: Successful completion.
*/
bool AwsDoc::AutoScaling::cleanupResources(const Aws::String &groupName,
                                           const Aws::String &templateName,
                                           const Aws::AutoScaling::AutoScalingClient
&autoScalingClient,
                                           const Aws::EC2::EC2Client &ec2Client) {

    bool result = true;

    // 14. Delete the Auto Scaling group.
    if (!groupName.empty() &&
        (askYesNoQuestion(
            Aws::String("Delete the Auto Scaling group '" + groupName +
                "' (y/n)?")))) {
        {
            Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
            request.SetAutoScalingGroupName(groupName);
            request.SetMinSize(0);
            request.SetDesiredCapacity(0);

            Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
                autoScalingClient.UpdateAutoScalingGroup(request);

            if (outcome.IsSuccess()) {
                std::cout
                    << "The minimum size and desired capacity of the Auto
Scaling group "
                    << "was set to zero before terminating the instances."
                    << std::endl;
            }
            else {

```

```

        std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                    << outcome.GetError().GetMessage() << std::endl;
        result = false;
    }
}

Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> autoScalingGroups;
if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                       autoScalingClient)) {
    if (!autoScalingGroups.empty()) {
        Aws::Vector<Aws::String> instanceIDs = instancesToInstanceIDs(
            autoScalingGroups[0].GetInstances());
        for (const Aws::String &instanceID: instanceIDs) {

            Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest request;
            request.SetInstanceId(instanceID);
            request.SetShouldDecrementDesiredCapacity(true);

            Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome outcome =
                autoScalingClient.TerminateInstanceInAutoScalingGroup(
                    request);

            if (outcome.IsSuccess()) {
                std::cout << "Initiating termination of EC2 instance '"
                            << instanceID << "'." << std::endl;
            }
            else {
                std::cerr
                    << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                    << outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
        }
    }

    std::cout
        << "Waiting for the EC2 instances to terminate before deleting
the "

        << "Auto Scaling group..." << std::endl;
    waitForInstances(groupName, autoScalingGroups, autoScalingClient);
}

```

```

    {
        Aws::AutoScaling::Model::DeleteAutoScalingGroupRequest request;
        request.SetAutoScalingGroupName(groupName);

        Aws::AutoScaling::Model::DeleteAutoScalingGroupOutcome outcome =
            autoScalingClient.DeleteAutoScalingGroup(request);

        if (outcome.IsSuccess()) {
            std::cout << "Auto Scaling group '" << groupName << "' was deleted."
                << std::endl;
        }
        else {
            std::cerr << "Error with AutoScaling::DeleteAutoScalingGroup. "
                << outcome.GetError().GetMessage()
                << std::endl;
            result = false;
        }
    }
}

// 15. Delete the EC2 launch template.
if (!templateName.empty() && (askYesNoQuestion(
    Aws::String("Delete the EC2 launch template '" + templateName +
        "' (y/n)?")))) {
    Aws::EC2::Model::DeleteLaunchTemplateRequest request;
    request.SetLaunchTemplateName(templateName);

    Aws::EC2::Model::DeleteLaunchTemplateOutcome outcome =
        ec2Client.DeleteLaunchTemplate(request);

    if (outcome.IsSuccess()) {
        std::cout << "EC2 launch template '" << templateName << "' was deleted."
            << std::endl;
    }
    else {
        std::cerr << "Error with EC2::DeleteLaunchTemplate. "
            << outcome.GetError().GetMessage()
            << std::endl;
        result = false;
    }
}

return result;
}

```

```

//! Routine which retrieves Auto Scaling group descriptions.
/*!
\sa describeGroup()
\param groupName: An Auto Scaling group name.
\param autoScalingGroups: Vector to receive 'AutoScalingGroup' records.
\param client: 'AutoScalingClient' instance.
\return bool: Successful completion.
*/
bool AwsDoc::AutoScaling::describeGroup(const Aws::String &groupName,

    Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> &autoScalingGroup,
                                   const Aws::AutoScaling::AutoScalingClient
    &client) {
    // 5. Retrieve a description of the Auto Scaling group.
    Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
    Aws::Vector<Aws::String> groupNames;
    groupNames.push_back(groupName);
    request.SetAutoScalingGroupNames(groupNames);

    Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
        client.DescribeAutoScalingGroups(request);

    if (outcome.IsSuccess()) {
        autoScalingGroup = outcome.GetResult().GetAutoScalingGroups();
    }
    else {
        std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups. "
                    << outcome.GetError().GetMessage()
                    << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 다음 주제를 참조하십시오.

- [CreateAutoScalingGroup](#)
- [DeleteAutoScalingGroup](#)
- [DescribeAutoScalingGroups](#)
- [DescribeAutoScalingInstances](#)

- [DescribeScalingActivities](#)
- [DisableMetricsCollection](#)
- [EnableMetricsCollection](#)
- [SetDesiredCapacity](#)
- [TerminateInstanceInAutoScalingGroup](#)
- [UpdateAutoScalingGroup](#)

## 작업

### CreateAutoScalingGroup

다음 코드 예시는 CreateAutoScalingGroup의 사용 방법을 보여줍니다.

SDK for C++

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::CreateAutoScalingGroupRequest request;
request.SetAutoScalingGroupName(groupName);
Aws::Vector<Aws::String> availabilityGroupZones;
availabilityGroupZones.push_back(
    availabilityZones[availabilityZoneChoice - 1].GetZoneName());
request.SetAvailabilityZones(availabilityGroupZones);
request.SetMaxSize(1);
request.SetMinSize(1);

Aws::AutoScaling::Model::LaunchTemplateSpecification
launchTemplateSpecification;
launchTemplateSpecification.SetLaunchTemplateName(templateName);
```

```

request.SetLaunchTemplate(launchTemplateSpecification);

Aws::AutoScaling::Model::CreateAutoScalingGroupOutcome outcome =
    autoScalingClient.CreateAutoScalingGroup(request);

if (outcome.IsSuccess()) {
    std::cout << "Created Auto Scaling group '" << groupName << "'..."
               << std::endl;
}
else if (outcome.GetError().GetErrorType() ==
         Aws::AutoScaling::AutoScalingErrors::ALREADY_EXISTS_FAULT) {
    std::cout << "Auto Scaling group '" << groupName << "' already exists."
               << std::endl;
}
else {
    std::cerr << "Error with AutoScaling::CreateAutoScalingGroup. "
               << outcome.GetError().GetMessage()
               << std::endl;
}
}

```

- API에 대한 자세한 설명은 AWS SDK for C++ API 참조 문서의 [CreateAutoScalingGroup](#)을 참조하세요.

## DeleteAutoScalingGroup

다음 코드 예시는 DeleteAutoScalingGroup의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

```

```

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

    Aws::AutoScaling::Model::DeleteAutoScalingGroupRequest request;
    request.SetAutoScalingGroupName(groupName);

    Aws::AutoScaling::Model::DeleteAutoScalingGroupOutcome outcome =
        autoScalingClient.DeleteAutoScalingGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "Auto Scaling group '" << groupName << "' was deleted."
            << std::endl;
    }
    else {
        std::cerr << "Error with AutoScaling::DeleteAutoScalingGroup. "
            << outcome.GetError().GetMessage()
            << std::endl;
        result = false;
    }
}

```

- API에 대한 자세한 설명은 AWS SDK for C++ API 참조 문서의 [DeleteAutoScalingGroup](#)을 참조하세요.

## DescribeAutoScalingGroups

다음 코드 예시는 DescribeAutoScalingGroups의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

```

```

Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
Aws::Vector<Aws::String> groupNames;
groupNames.push_back(groupName);
request.SetAutoScalingGroupNames(groupNames);

Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
    client.DescribeAutoScalingGroups(request);

if (outcome.IsSuccess()) {
    autoScalingGroup = outcome.GetResult().GetAutoScalingGroups();
}
else {
    std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups. "
                << outcome.GetError().GetMessage()
                << std::endl;
}

```

- API에 대한 세부 정보는 AWS SDK for C++ API 참조의 [DescribeAutoScalingGroups](#)를 참조하세요.

## DescribeAutoScalingInstances

다음 코드 예시는 DescribeAutoScalingInstances의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::DescribeAutoScalingInstancesRequest request;

```

```

request.SetInstanceIds(instanceIDs);

Aws::AutoScaling::Model::DescribeAutoScalingInstancesOutcome outcome =
    client.DescribeAutoScalingInstances(request);

if (outcome.IsSuccess()) {
    const Aws::Vector<Aws::AutoScaling::Model::AutoScalingInstanceDetails>
&instancesDetails =
    outcome.GetResult().GetAutoScalingInstances();

}
else {
    std::cerr << "Error with AutoScaling::DescribeAutoScalingInstances. "
        << outcome.GetError().GetMessage()
        << std::endl;
    return false;
}

```

- API에 대한 세부 정보는 AWS SDK for C++ API 참조의 [DescribeAutoScalingInstances](#)를 참조하세요.

## DescribeScalingActivities

다음 코드 예시는 DescribeScalingActivities의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::DescribeScalingActivitiesRequest request;

```

```

request.SetAutoScalingGroupName(groupName);

Aws::Vector<Aws::AutoScaling::Model::Activity> allActivities;
Aws::String nextToken; // Used for pagination;
do {
    if (!nextToken.empty()) {
        request.SetNextToken(nextToken);
    }
    Aws::AutoScaling::Model::DescribeScalingActivitiesOutcome outcome =
        autoScalingClient.DescribeScalingActivities(request);

    if (outcome.IsSuccess()) {
        const Aws::Vector<Aws::AutoScaling::Model::Activity> &activities =
            outcome.GetResult().GetActivities();
        allActivities.insert(allActivities.end(), activities.begin(),
activities.end());
        nextToken = outcome.GetResult().GetNextToken();
    }
    else {
        std::cerr << "Error with AutoScaling::DescribeScalingActivities. "
                    << outcome.GetError().GetMessage()
                    << std::endl;
    }
} while (!nextToken.empty());

std::cout << "Found " << allActivities.size() << " activities."
          << std::endl;
std::cout << "Activities are ordered with the most recent first."
          << std::endl;
for (const Aws::AutoScaling::Model::Activity &activity: allActivities) {
    std::cout << activity.GetDescription() << std::endl;
    std::cout << activity.GetDetails() << std::endl;
}

```

- API에 대한 세부 정보는 AWS SDK for C++ API 참조의 [DescribeScalingActivities](#)를 참조하세요.

## DisableMetricsCollection

다음 코드 예시는 DisableMetricsCollection의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::DisableMetricsCollectionRequest request;
request.SetAutoScalingGroupName(groupName);

Aws::AutoScaling::Model::DisableMetricsCollectionOutcome outcome =
    autoScalingClient.DisableMetricsCollection(request);

if (outcome.IsSuccess()) {
    std::cout << "Metrics collection has been disabled." << std::endl;
}
else {
    std::cerr << "Error with AutoScaling::DisableMetricsCollection. "
        << outcome.GetError().GetMessage()
        << std::endl;
}
```

- API에 대한 세부 정보는 AWS SDK for C++ API 참조의 [DisableMetricsCollection](#)을 참조하세요.

## EnableMetricsCollection

다음 코드 예시는 EnableMetricsCollection의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::EnableMetricsCollectionRequest request;
request.SetAutoScalingGroupName(groupName);

request.AddMetrics("GroupMinSize");
request.AddMetrics("GroupMaxSize");
request.AddMetrics("GroupDesiredCapacity");
request.AddMetrics("GroupInServiceInstances");
request.AddMetrics("GroupTotalInstances");
request.SetGranularity("1Minute");

Aws::AutoScaling::Model::EnableMetricsCollectionOutcome outcome =
    autoScalingClient.EnableMetricsCollection(request);
if (outcome.IsSuccess()) {
    std::cout << "Auto Scaling metrics have been enabled."
              << std::endl;
}
else {
    std::cerr << "Error with AutoScaling::EnableMetricsCollection. "
              << outcome.GetError().GetMessage()
              << std::endl;
}
```

- API에 대한 세부 정보는 AWS SDK for C++ API 참조의 [EnableMetricsCollection](#)을 참조하세요.

## SetDesiredCapacity

다음 코드 예시는 SetDesiredCapacity의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::SetDesiredCapacityRequest request;
request.SetAutoScalingGroupName(groupName);
request.SetDesiredCapacity(2);

Aws::AutoScaling::Model::SetDesiredCapacityOutcome outcome =
    autoScalingClient.SetDesiredCapacity(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Error with AutoScaling::SetDesiredCapacityRequest. "
                << outcome.GetError().GetMessage()
                << std::endl;
}
```

- API에 대한 세부 정보는 AWS SDK for C++ API 참조의 [SetDesiredCapacity](#)를 참조하세요.

## TerminateInstanceInAutoScalingGroup

다음 코드 예시는 TerminateInstanceInAutoScalingGroup의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

    Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest request;
    request.SetInstanceId(instanceIDs[instanceNumber - 1]);
    request.SetShouldDecrementDesiredCapacity(false);

    Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome outcome
=
        autoScalingClient.TerminateInstanceInAutoScalingGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "Waiting for EC2 instance with ID '"
                    << instanceIDs[instanceNumber - 1] << "' to terminate..."
                    << std::endl;
    }
    else {
        std::cerr << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
                    << outcome.GetError().GetMessage()
                    << std::endl;
    }
}

```

- API에 대한 세부 정보는 AWS SDK for C++ API 참조의 [TerminateInstanceInAutoScalingGroup](#)을 참조하세요.

## UpdateAutoScalingGroup

다음 코드 예시는 UpdateAutoScalingGroup의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);

Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
request.SetAutoScalingGroupName(groupName);
request.SetMaxSize(3);

Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
    autoScalingClient.UpdateAutoScalingGroup(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                << outcome.GetError().GetMessage()
                << std::endl;
}
```

- API에 대한 자세한 설명은 AWS SDK for C++ API 참조 문서의 [UpdateAutoScalingGroup](#)을 참조하세요.

## SDK for C++를 사용한 CloudTrail 예제

다음 코드 예제에서는 CloudTrail과 AWS SDK for C++ 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 CreateTrail의 사용 방법을 보여줍니다.

SDK for C++

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Routine which creates an AWS CloudTrail trail.
/*!
 \param trailName: The name of the CloudTrail trail.
 \param bucketName: The Amazon S3 bucket designate for publishing logs.
 \param clientConfig: Aws client configuration.
 \return bool: Function succeeded.
*/
bool AwsDoc::CloudTrail::createTrail(const Aws::String trailName,
                                     const Aws::String bucketName,
                                     const Aws::Client::ClientConfiguration
                                     &clientConfig) {
    Aws::CloudTrail::CloudTrailClient trailClient(clientConfig);
    Aws::CloudTrail::Model::CreateTrailRequest request;
    request.SetName(trailName);
    request.SetS3BucketName(bucketName);

    Aws::CloudTrail::Model::CreateTrailOutcome outcome = trailClient.CreateTrail(
```

```

        request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created trail " << trailName << std::endl;
    }
    else {
        std::cerr << "Failed to create trail " << trailName <<
            ": " << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CreateTrail](#)을 참조하세요.

## DeleteTrail

다음 코드 예시는 DeleteTrail의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

// Routine which deletes an AWS CloudTrail trail.
/*!
 \param trailName: The name of the CloudTrail trail.
 \param clientConfig: Aws client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::CloudTrail::deleteTrail(const Aws::String trailName,
                                     const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::CloudTrail::CloudTrailClient trailClient(clientConfig);

    Aws::CloudTrail::Model::DeleteTrailRequest request;
    request.SetName(trailName);

    auto outcome = trailClient.DeleteTrail(request);
}

```

```

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted trail " << trailName << std::endl;
    }
    else {
        std::cerr << "Error deleting trail " << trailName << " " <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteTrail](#)을 참조하세요.

## DescribeTrail

다음 코드 예시는 DescribeTrail의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

// Routine which describes the AWS CloudTrail trails in an account.
/*!
 \param clientConfig: Aws client configuration.
 \return bool: Function succeeded.
 */

bool AwsDoc::CloudTrail::describeTrails(
    const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::CloudTrail::CloudTrailClient cloudTrailClient(clientConfig);
    Aws::CloudTrail::Model::DescribeTrailsRequest request;

    auto outcome = cloudTrailClient.DescribeTrails(request);
    if (outcome.IsSuccess()) {
        const Aws::Vector<Aws::CloudTrail::Model::Trail> &trails =
outcome.GetResult().GetTrailList();
        std::cout << trails.size() << " trail(s) found." << std::endl;
    }
}

```

```

        for (const Aws::CloudTrail::Model::Trail &trail: trails) {
            std::cout << trail.GetName() << std::endl;
        }
    }
    else {
        std::cerr << "Failed to describe trails." << outcome.GetError().GetMessage()
            << std::endl;
    }
    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DescribeTrail](#)을 참조하세요.

## LookupEvents

다음 코드 예시는 LookupEvents의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

// Routine which looks up events captured by AWS CloudTrail.
/*!
 \param clientConfig: Aws client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::CloudTrail::lookupEvents(
    const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::CloudTrail::CloudTrailClient cloudtrail(clientConfig);

    Aws::String nextToken; // Used for pagination.
    Aws::Vector<Aws::CloudTrail::Model::Event> allEvents;

    Aws::CloudTrail::Model::LookupEventsRequest request;

    size_t count = 0;
    do {

```

```

        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::CloudTrail::Model::LookupEventsOutcome outcome =
cloudtrail.LookupEvents(
    request);
        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::CloudTrail::Model::Event> &events =
outcome.GetResult().GetEvents();
            count += events.size();
            allEvents.insert(allEvents.end(), events.begin(), events.end());
            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cerr << "Error: " << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    } while (!nextToken.empty() && count <= 50); // Limit to 50 events.

    std::cout << "Found " << allEvents.size() << " event(s)." << std::endl;

    for (auto &event: allEvents) {
        std::cout << "Event name: " << event.GetEventName() << std::endl;
        std::cout << "Event source: " << event.GetEventSource() << std::endl;
        std::cout << "Event id: " << event.GetEventId() << std::endl;
        std::cout << "Resources: " << std::endl;
        for (auto &resource: event.GetResources()) {
            std::cout << "    " << resource.GetResourceName() << std::endl;
        }
    }

    return true;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [LookupEvents](#)를 참조하세요.

## SDK for C++를 사용한 CloudWatch 예제

다음 코드 예제에서는 CloudWatch와 AWS SDK for C++ 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

### DeleteAlarms

다음 코드 예시는 DeleteAlarms의 사용 방법을 보여줍니다.

SDK for C++

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

필수 파일을 포함합니다.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DeleteAlarmsRequest.h>
#include <iostream>
```

경보를 삭제합니다.

```
Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::DeleteAlarmsRequest request;
request.AddAlarmNames(alarm_name);

auto outcome = cw.DeleteAlarms(request);
```

```

    if (!outcome.IsSuccess())
    {
        std::cout << "Failed to delete CloudWatch alarm:" <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else
    {
        std::cout << "Successfully deleted CloudWatch alarm " << alarm_name
            << std::endl;
    }
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteAlarms](#)를 참조하세요.

## DescribeAlarmsForMetric

다음 코드 예시는 DescribeAlarmsForMetric의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

필수 파일을 포함합니다.

```

#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DescribeAlarmsRequest.h>
#include <aws/monitoring/model/DescribeAlarmsResult.h>
#include <iomanip>
#include <iostream>

```

경보를 설명합니다.

```

Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::DescribeAlarmsRequest request;
request.SetMaxRecords(1);

```

```

bool done = false;
bool header = false;
while (!done)
{
    auto outcome = cw.DescribeAlarms(request);
    if (!outcome.IsSuccess())
    {
        std::cout << "Failed to describe CloudWatch alarms:" <<
            outcome.GetError().GetMessage() << std::endl;
        break;
    }

    if (!header)
    {
        std::cout << std::left <<
            std::setw(32) << "Name" <<
            std::setw(64) << "Arn" <<
            std::setw(64) << "Description" <<
            std::setw(20) << "LastUpdated" <<
            std::endl;
        header = true;
    }

    const auto &alarms = outcome.GetResult().GetMetricAlarms();
    for (const auto &alarm : alarms)
    {
        std::cout << std::left <<
            std::setw(32) << alarm.GetAlarmName() <<
            std::setw(64) << alarm.GetAlarmArn() <<
            std::setw(64) << alarm.GetAlarmDescription() <<
            std::setw(20) <<
            alarm.GetAlarmConfigurationUpdatedTimestamp().ToGmtString(
                SIMPLE_DATE_FORMAT_STR) <<
            std::endl;
    }

    const auto &next_token = outcome.GetResult().GetNextToken();
    request.SetNextToken(next_token);
    done = next_token.empty();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DescribeAlarmsForMetric](#)을 참조하세요.

## DisableAlarmActions

다음 코드 예시는 DisableAlarmActions의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

필수 파일을 포함합니다.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/DisableAlarmActionsRequest.h>
#include <iostream>
```

경보 작업 사용을 중지합니다.

```
Aws::CloudWatch::CloudWatchClient cw;

Aws::CloudWatch::Model::DisableAlarmActionsRequest
disableAlarmActionsRequest;
disableAlarmActionsRequest.AddAlarmNames(alarm_name);

auto disableAlarmActionsOutcome =
cw.DisableAlarmActions(disableAlarmActionsRequest);
if (!disableAlarmActionsOutcome.IsSuccess())
{
    std::cout << "Failed to disable actions for alarm " << alarm_name <<
        ": " << disableAlarmActionsOutcome.GetError().GetMessage() <<
        std::endl;
}
else
{
    std::cout << "Successfully disabled actions for alarm " <<
        alarm_name << std::endl;
}
```

- API 세부 정보에 대한 내용은 AWS SDK for C++ API 참조의 [DisableAlarmActions](#)를 참조하세요.

## EnableAlarmActions

다음 코드 예시는 EnableAlarmActions의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

필수 파일을 포함합니다.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/EnableAlarmActionsRequest.h>
#include <aws/monitoring/model/PutMetricAlarmRequest.h>
#include <iostream>
```

경보 작업을 사용합니다.

```
Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::PutMetricAlarmRequest request;
request.SetAlarmName(alarm_name);
request.SetComparisonOperator(
    Aws::CloudWatch::Model::ComparisonOperator::GreaterThanThreshold);
request.SetEvaluationPeriods(1);
request.SetMetricName("CPUUtilization");
request.SetNamespace("AWS/EC2");
request.SetPeriod(60);
request.SetStatistic(Aws::CloudWatch::Model::Statistic::Average);
request.SetThreshold(70.0);
request.SetActionsEnabled(false);
request.SetAlarmDescription("Alarm when server CPU exceeds 70%");
request.SetUnit(Aws::CloudWatch::Model::StandardUnit::Seconds);
request.AddAlarmActions(actionArn);

Aws::CloudWatch::Model::Dimension dimension;
```

```

dimension.SetName("InstanceId");
dimension.SetValue(instanceId);
request.AddDimensions(dimension);

auto outcome = cw.PutMetricAlarm(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to create CloudWatch alarm:" <<
        outcome.GetError().GetMessage() << std::endl;
    return;
}

Aws::CloudWatch::Model::EnableAlarmActionsRequest enable_request;
enable_request.AddAlarmNames(alarm_name);

auto enable_outcome = cw.EnableAlarmActions(enable_request);
if (!enable_outcome.IsSuccess())
{
    std::cout << "Failed to enable alarm actions:" <<
        enable_outcome.GetError().GetMessage() << std::endl;
    return;
}

std::cout << "Successfully created alarm " << alarm_name <<
    " and enabled actions on it." << std::endl;

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [EnableAlarmActions](#)를 참조하세요.

## ListMetrics

다음 코드 예시는 ListMetrics의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

필수 파일을 포함합니다.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/ListMetricsRequest.h>
#include <aws/monitoring/model/ListMetricsResult.h>
#include <iomanip>
#include <iostream>
```

지표를 나열합니다.

```
Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::ListMetricsRequest request;

if (argc > 1)
{
    request.SetMetricName(argv[1]);
}

if (argc > 2)
{
    request.SetNamespace(argv[2]);
}

bool done = false;
bool header = false;
while (!done)
{
    auto outcome = cw.ListMetrics(request);
    if (!outcome.IsSuccess())
    {
        std::cout << "Failed to list CloudWatch metrics:" <<
            outcome.GetError().GetMessage() << std::endl;
        break;
    }

    if (!header)
    {
        std::cout << std::left << std::setw(48) << "MetricName" <<
            std::setw(32) << "Namespace" << "DimensionNameValuePairs" <<
            std::endl;
        header = true;
    }
}
```

```

const auto &metrics = outcome.GetResult().GetMetrics();
for (const auto &metric : metrics)
{
    std::cout << std::left << std::setw(48) <<
        metric.GetMetricName() << std::setw(32) <<
        metric.GetNamespace();
    const auto &dimensions = metric.GetDimensions();
    for (auto iter = dimensions.cbegin();
        iter != dimensions.cend(); ++iter)
    {
        const auto &dimkv = *iter;
        std::cout << dimkv.GetName() << " = " << dimkv.GetValue();
        if (iter + 1 != dimensions.cend())
        {
            std::cout << ", ";
        }
    }
    std::cout << std::endl;
}

const auto &next_token = outcome.GetResult().GetNextToken();
request.SetNextToken(next_token);
done = next_token.empty();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ListMetrics](#)를 참조하세요.

## PutMetricAlarm

다음 코드 예시는 PutMetricAlarm의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

필수 파일을 포함합니다.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/PutMetricAlarmRequest.h>
#include <iostream>
```

지표를 감시할 경보를 생성합니다.

```
Aws::CloudWatch::CloudWatchClient cw;
Aws::CloudWatch::Model::PutMetricAlarmRequest request;
request.SetAlarmName(alarm_name);
request.SetComparisonOperator(
    Aws::CloudWatch::Model::ComparisonOperator::GreaterThanThreshold);
request.SetEvaluationPeriods(1);
request.SetMetricName("CPUUtilization");
request.SetNamespace("AWS/EC2");
request.SetPeriod(60);
request.SetStatistic(Aws::CloudWatch::Model::Statistic::Average);
request.SetThreshold(70.0);
request.SetActionsEnabled(false);
request.SetAlarmDescription("Alarm when server CPU exceeds 70%");
request.SetUnit(Aws::CloudWatch::Model::StandardUnit::Seconds);

Aws::CloudWatch::Model::Dimension dimension;
dimension.SetName("InstanceId");
dimension.SetValue(instanceId);

request.AddDimensions(dimension);

auto outcome = cw.PutMetricAlarm(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to create CloudWatch alarm:" <<
        outcome.GetError().GetMessage() << std::endl;
}
else
{
    std::cout << "Successfully created CloudWatch alarm " << alarm_name
        << std::endl;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [PutMetricAlarm](#)을 참조하세요.

## PutMetricData

다음 코드 예시는 PutMetricData의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

필수 파일을 포함합니다.

```
#include <aws/core/Aws.h>
#include <aws/monitoring/CloudWatchClient.h>
#include <aws/monitoring/model/PutMetricDataRequest.h>
#include <iostream>
```

지표에 데이터를 입력합니다.

```
Aws::CloudWatch::CloudWatchClient cw;

Aws::CloudWatch::Model::Dimension dimension;
dimension.SetName("UNIQUE_PAGES");
dimension.SetValue("URLS");

Aws::CloudWatch::Model::MetricDatum datum;
datum.SetMetricName("PAGES_VISITED");
datum.SetUnit(Aws::CloudWatch::Model::StandardUnit::None);
datum.SetValue(data_point);
datum.AddDimensions(dimension);

Aws::CloudWatch::Model::PutMetricDataRequest request;
request.SetNamespace("SITE/TRAFFIC");
request.AddMetricData(datum);

auto outcome = cw.PutMetricData(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to put sample metric data:" <<
```

```

        outcome.GetError().GetMessage() << std::endl;
    }
    else
    {
        std::cout << "Successfully put sample metric data" << std::endl;
    }
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [PutMetricData](#)를 참조하세요.

## SDK for C++를 사용한 CloudWatch Logs 예제

다음 코드 예제에서는 CloudWatch Logs와 AWS SDK for C++ 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

#### DeleteSubscriptionFilter

다음 코드 예시는 DeleteSubscriptionFilter의 사용 방법을 보여줍니다.

SDK for C++

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

필수 파일을 포함합니다.

```
#include <aws/core/Aws.h>
#include <aws/core/utils/Outcome.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/DeleteSubscriptionFilterRequest.h>
#include <iostream>
```

구독 필터를 삭제합니다.

```
Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
Aws::CloudWatchLogs::Model::DeleteSubscriptionFilterRequest request;
request.SetFilterName(filter_name);
request.SetLogGroupName(log_group);

auto outcome = cwl.DeleteSubscriptionFilter(request);
if (!outcome.IsSuccess()) {
    std::cout << "Failed to delete CloudWatch log subscription filter "
                << filter_name << ": " << outcome.GetError().GetMessage() <<
                std::endl;
} else {
    std::cout << "Successfully deleted CloudWatch logs subscription " <<
                "filter " << filter_name << std::endl;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteSubscriptionFilter](#) 참조하세요.

## DescribeSubscriptionFilters

다음 코드 예시는 DescribeSubscriptionFilters의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

필수 파일을 포함합니다.

```
#include <aws/core/Aws.h>
#include <aws/core/utils/Outcome.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/DescribeSubscriptionFiltersRequest.h>
#include <aws/logs/model/DescribeSubscriptionFiltersResult.h>
#include <iostream>
#include <iomanip>
```

구독 필터를 나열합니다.

```
Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
Aws::CloudWatchLogs::Model::DescribeSubscriptionFiltersRequest request;
request.SetLogGroupName(log_group);
request.SetLimit(1);

bool done = false;
bool header = false;
while (!done) {
    auto outcome = cwl.DescribeSubscriptionFilters(
        request);
    if (!outcome.IsSuccess()) {
        std::cout << "Failed to describe CloudWatch subscription filters "
            << "for log group " << log_group << ": " <<
            outcome.GetError().GetMessage() << std::endl;
        break;
    }

    if (!header) {
        std::cout << std::left << std::setw(32) << "Name" <<
            std::setw(64) << "FilterPattern" << std::setw(64) <<
            "DestinationArn" << std::endl;
        header = true;
    }

    const auto &filters = outcome.GetResult().GetSubscriptionFilters();
    for (const auto &filter : filters) {
        std::cout << std::left << std::setw(32) <<
            filter.GetFilterName() << std::setw(64) <<
            filter.GetFilterPattern() << std::setw(64) <<
            filter.GetDestinationArn() << std::endl;
    }
}
```

```

    }

    const auto &next_token = outcome.GetResult().GetNextToken();
    request.SetNextToken(next_token);
    done = next_token.empty();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DescribeSubscriptionFilters](#) 참조하세요.

## PutSubscriptionFilter

다음 코드 예시는 PutSubscriptionFilter의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

필수 파일을 포함합니다.

```

#include <aws/core/Aws.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/PutSubscriptionFilterRequest.h>
#include <aws/core/Utils/Outcome.h>
#include <iostream>

```

구독 필터를 생성합니다.

```

Aws::CloudWatchLogs::CloudWatchLogsClient cwl;
Aws::CloudWatchLogs::Model::PutSubscriptionFilterRequest request;
request.SetFilterName(filter_name);
request.SetFilterPattern(filter_pattern);
request.SetLogGroupName(log_group);
request.SetDestinationArn(dest_arn);
auto outcome = cwl.PutSubscriptionFilter(request);

```

```

    if (!outcome.IsSuccess())
    {
        std::cout << "Failed to create CloudWatch logs subscription filter "
                    << filter_name << ": " << outcome.GetError().GetMessage() <<
                    std::endl;
    }
    else
    {
        std::cout << "Successfully created CloudWatch logs subscription " <<
                    "filter " << filter_name << std::endl;
    }

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [PutSubscriptionFilter](#) 참조하세요.

## SDK for C++를 사용한 CodeBuild 예제

다음 코드 예제에서는 CodeBuild와 AWS SDK for C++ 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

### ListBuilds

다음 코드 예시는 ListBuilds의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! List the CodeBuild builds.
/*!
    \param sortType: 'SortOrderType' type.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::CodeBuild::listBuilds(Aws::CodeBuild::Model::SortOrderType sortType,
                                     const Aws::Client::ClientConfiguration
                                     &clientConfiguration) {
    Aws::CodeBuild::CodeBuildClient codeBuildClient(clientConfiguration);

    Aws::CodeBuild::Model::ListBuildsRequest listBuildsRequest;
    listBuildsRequest.SetSortOrder(sortType);

    Aws::String nextToken; // Used for pagination.

    do {
        if (!nextToken.empty()) {
            listBuildsRequest.SetNextToken(nextToken);
        }

        Aws::CodeBuild::Model::ListBuildsOutcome listBuildsOutcome =
        codeBuildClient.ListBuilds(
            listBuildsRequest);

        if (listBuildsOutcome.IsSuccess()) {
            const Aws::Vector<Aws::String> &ids =
            listBuildsOutcome.GetResult().GetIds();
            if (!ids.empty()) {

                std::cout << "Information about each build:" << std::endl;
                Aws::CodeBuild::Model::BatchGetBuildsRequest getBuildsRequest;
                getBuildsRequest.SetIds(listBuildsOutcome.GetResult().GetIds());
            }
        }
    } while (nextToken != "");
}

```

```

        Aws::CodeBuild::Model::BatchGetBuildsOutcome getBuildsOutcome =
codeBuildClient.BatchGetBuilds(
    getBuildsRequest);

    if (getBuildsOutcome.IsSuccess()) {
        const Aws::Vector<Aws::CodeBuild::Model::Build> &builds =
getBuildsOutcome.GetResult().GetBuilds();
        std::cout << builds.size() << " build(s) found." << std::endl;
        for (auto val: builds) {
            std::cout << val.GetId() << std::endl;
        }
    } else {
        std::cerr << "Error getting builds"
            << getBuildsOutcome.GetError().GetMessage() <<
std::endl;

        return false;
    }
} else {
    std::cout << "No builds found." << std::endl;
}

// Get the next token for pagination.

nextToken = listBuildsOutcome.GetResult().GetNextToken();
} else {
    std::cerr << "Error listing builds"
        << listBuildsOutcome.GetError().GetMessage()
        << std::endl;
    return false;
}

} while (!nextToken.

    empty()

);

return true;
}

```

- API에 대한 세부 정보는 AWS SDK for C++ API 참조의 [ListBuilds](#)를 참조하세요.

## ListProjects

다음 코드 예시는 ListProjects의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
//! List the CodeBuild projects.
/*!
  \param sortType: 'SortOrderType' type.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::CodeBuild::listProjects(Aws::CodeBuild::Model::SortOrderType sortType,
                                     const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::CodeBuild::CodeBuildClient codeBuildClient(clientConfiguration);

    Aws::CodeBuild::Model::ListProjectsRequest listProjectsRequest;
    listProjectsRequest.SetSortOrder(sortType);

    Aws::String nextToken; // Next token for pagination.
    Aws::Vector<Aws::String> allProjects;

    do {
        if (!nextToken.empty()) {
            listProjectsRequest.SetNextToken(nextToken);
        }

        Aws::CodeBuild::Model::ListProjectsOutcome outcome =
codeBuildClient.ListProjects(
    listProjectsRequest);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::String> &projects =
outcome.GetResult().GetProjects();
            allProjects.insert(allProjects.end(), projects.begin(), projects.end());
            nextToken = outcome.GetResult().GetNextToken();
        }
    } while (true);
}
```

```

    }

    else {
        std::cerr << "Error listing projects" << outcome.GetError().GetMessage()
            << std::endl;
    }

} while (!nextToken.empty());

std::cout << allProjects.size() << " project(s) found." << std::endl;
for (auto project: allProjects) {
    std::cout << project << std::endl;
}

return true;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ListProjects](#)를 참조하세요.

## StartBuild

다음 코드 예시는 StartBuild의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Start an AWS CodeBuild project build.
/*!
 \param projectName: A CodeBuild project name.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::CodeBuild::startBuild(const Aws::String &projectName,
                                   const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::CodeBuild::CodeBuildClient codeBuildClient(clientConfiguration);

```

```

    Aws::CodeBuild::Model::StartBuildRequest startBuildRequest;
    startBuildRequest.SetProjectName(projectName);

    Aws::CodeBuild::Model::StartBuildOutcome outcome = codeBuildClient.StartBuild(
        startBuildRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully started build" << std::endl;
        std::cout << "Build ID: " << outcome.GetResult().GetBuild().GetId()
            << std::endl;
    }

    else {
        std::cerr << "Error starting build" << outcome.GetError().GetMessage()
            << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [StartBuild](#)를 참조하세요.

## SDK for C++를 사용한 Amazon Cognito 자격 증명 공급자 예제

다음 코드 예제에서는 Amazon Cognito 자격 증명 공급자와 AWS SDK for C++ 함께를 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

- [시작하기](#)
- [작업](#)

- [시나리오](#)

## 시작하기

Hello Amazon Cognito

다음 코드 예시에서는 Amazon Cognito 사용을 시작하는 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

CMakeLists.txt CMake 파일의 코드입니다.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS cognito-idp)

# Set this project's name.
project("hello_cognito")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
"${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
```

```

find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory for
    # running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
    # may need to uncomment this
                                # and set the proper subdirectory to the
    executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
        ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
    hello_cognito.cpp)

target_link_libraries(${PROJECT_NAME}
    ${AWSSDK_LINK_LIBRARIES})

```

hello\_cognito.cpp 소스 파일의 코드입니다.

```

#include <aws/core/Aws.h>
#include <aws/cognito-idp/CognitoIdentityProviderClient.h>
#include <aws/cognito-idp/model/ListUserPoolsRequest.h>
#include <iostream>

/*
 * A "Hello Cognito" starter application which initializes an Amazon Cognito client
 * and lists the Amazon Cognito
 * user pools.
 *
 * main function
 *
 * Usage: 'hello_cognito'
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.

```

```

// options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
Aws::InitAPI(options); // Should only be called once.
int result = 0;
{
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
cognitoClient(clientConfig);

    Aws::String nextToken; // Used for pagination.
    std::vector<Aws::String> userPools;

    do {
        Aws::CognitoIdentityProvider::Model::ListUserPoolsRequest
listUserPoolsRequest;
        if (!nextToken.empty()) {
            listUserPoolsRequest.SetNextToken(nextToken);
        }

        Aws::CognitoIdentityProvider::Model::ListUserPoolsOutcome
listUserPoolsOutcome =
            cognitoClient.ListUserPools(listUserPoolsRequest);

        if (listUserPoolsOutcome.IsSuccess()) {
            for (auto &userPool:
listUserPoolsOutcome.GetResult().GetUserPools()) {

                userPools.push_back(userPool.GetName());
            }

            nextToken = listUserPoolsOutcome.GetResult().GetNextToken();
        } else {
            std::cerr << "ListUserPools error: " <<
listUserPoolsOutcome.GetError().GetMessage() << std::endl;
            result = 1;
            break;
        }

    } while (!nextToken.empty());
    std::cout << userPools.size() << " user pools found." << std::endl;
    for (auto &userPool: userPools) {

```

```

        std::cout << "    user pool: " << userPool << std::endl;
    }
}

Aws::ShutdownAPI(options); // Should only be called once.
return result;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ListUserPools](#)을 참조하세요.

## 작업

### AdminGetUser

다음 코드 예시는 AdminGetUser의 사용 방법을 보여줍니다.

SDK for C++

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
    client(clientConfig);

    Aws::CognitoIdentityProvider::Model::AdminGetUserRequest request;
    request.SetUsername(userName);
    request.SetUserPoolId(userPoolID);

    Aws::CognitoIdentityProvider::Model::AdminGetUserOutcome outcome =
        client.AdminGetUser(request);

    if (outcome.IsSuccess()) {
        std::cout << "The status for " << userName << " is " <<

```

```

Aws::CognitoIdentityProvider::Model::UserStatusTypeMapper::GetNameForUserStatusType(
    outcome.GetResult().GetUserStatus()) << std::endl;
    std::cout << "Enabled is " << outcome.GetResult().GetEnabled() << std::endl;
}
else {
    std::cerr << "Error with CognitoIdentityProvider::AdminGetUser. "
        << outcome.GetError().GetMessage()
        << std::endl;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [AdminGetUser](#)를 참조하세요.

## AdminInitiateAuth

다음 코드 예시는 AdminInitiateAuth의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

Aws::CognitoIdentityProvider::Model::AdminInitiateAuthRequest request;
request.SetClientId(clientID);
request.SetUserPoolId(userPoolID);
request.AddAuthParameters("USERNAME", userName);
request.AddAuthParameters("PASSWORD", password);
request.SetAuthFlow(

Aws::CognitoIdentityProvider::Model::AuthFlowType::ADMIN_USER_PASSWORD_AUTH);

```

```

Aws::CognitoIdentityProvider::Model::AdminInitiateAuthOutcome outcome =
    client.AdminInitiateAuth(request);

if (outcome.IsSuccess()) {
    std::cout << "Call to AdminInitiateAuth was successful." << std::endl;
    sessionResult = outcome.GetResult().GetSession();
}
else {
    std::cerr << "Error with CognitoIdentityProvider::AdminInitiateAuth. "
        << outcome.GetError().GetMessage()
        << std::endl;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [AdminInitiateAuth](#)를 참조하세요.

## AdminRespondToAuthChallenge

다음 코드 예시는 AdminRespondToAuthChallenge의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeRequest
request;
request.AddChallengeResponses("USERNAME", userName);
request.AddChallengeResponses("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

```

```

        request.SetChallengeName(
            Aws::CognitoIdentityProvider::Model::ChallengeNameType::SOFTWARE_TOKEN_MFA);
        request.SetClientId(clientID);
        request.SetUserPoolId(userPoolID);
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeOutcome
outcome =
            client.AdminRespondToAuthChallenge(request);

        if (outcome.IsSuccess()) {
            std::cout << "Here is the response to the challenge.\n" <<
                outcome.GetResult().GetAuthenticationResult().Jsonize().View().WriteReadable()
                << std::endl;

            accessToken =
outcome.GetResult().GetAuthenticationResult().GetAccessToken();
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::AdminRespondToAuthChallenge. "
                << outcome.GetError().GetMessage()
                << std::endl;
            return false;
        }
    }
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [AdminRespondToAuthChallenge](#)를 참조하세요.

## AssociateSoftwareToken

다음 코드 예시는 AssociateSoftwareToken의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
    client(clientConfig);

    Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenRequest request;
    request.SetSession(session);

    Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenOutcome outcome =
        client.AssociateSoftwareToken(request);

    if (outcome.IsSuccess()) {
        std::cout
            << "Enter this setup key into an authenticator app, for example
Google Authenticator."
            << std::endl;
        std::cout << "Setup key: " << outcome.GetResult().GetSecretCode()
            << std::endl;
#ifdef USING_QR
        printAsterisksLine();
        std::cout << "\nOr scan the QR code in the file '" << QR_CODE_PATH <<
            "."
            << std::endl;

        saveQRCode(std::string("otpauth://totp/") + userName + "?secret=" +
            outcome.GetResult().GetSecretCode());
#endif // USING_QR
        session = outcome.GetResult().GetSession();
    }
    else {
        std::cerr << "Error with
CognitoIdentityProvider::AssociateSoftwareToken. "
            << outcome.GetError().GetMessage()
            << std::endl;
        return false;
    }
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [AssociateSoftwareToken](#)을 참조하세요.

## ConfirmSignUp

다음 코드 예시는 ConfirmSignUp의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

Aws::CognitoIdentityProvider::Model::ConfirmSignUpRequest request;
request.SetClientId(clientID);
request.SetConfirmationCode(confirmationCode);
request.SetUsername(userName);

Aws::CognitoIdentityProvider::Model::ConfirmSignUpOutcome outcome =
    client.ConfirmSignUp(request);

if (outcome.IsSuccess()) {
    std::cout << "ConfirmSignup was Successful."
              << std::endl;
}
else {
    std::cerr << "Error with CognitoIdentityProvider::ConfirmSignUp. "
              << outcome.GetError().GetMessage()
              << std::endl;
    return false;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ConfirmSignUp](#)을 참조하세요.

## DeleteUser

다음 코드 예시는 DeleteUser의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

Aws::CognitoIdentityProvider::Model::DeleteUserRequest request;
request.SetAccessToken(accessToken);

Aws::CognitoIdentityProvider::Model::DeleteUserOutcome outcome =
    client.DeleteUser(request);

if (outcome.IsSuccess()) {
    std::cout << "The user " << userName << " was deleted."
              << std::endl;
}
else {
    std::cerr << "Error with CognitoIdentityProvider::DeleteUser. "
              << outcome.GetError().GetMessage()
              << std::endl;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteUser](#)를 참조하세요.

## ResendConfirmationCode

다음 코드 예시는 ResendConfirmationCode의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeRequest request;
request.SetUsername(userName);
request.SetClientId(clientID);

Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeOutcome outcome =
    client.ResendConfirmationCode(request);

if (outcome.IsSuccess()) {
    std::cout
        << "CognitoIdentityProvider::ResendConfirmationCode was
successful."
        << std::endl;
}
else {
    std::cerr << "Error with
CognitoIdentityProvider::ResendConfirmationCode. "
        << outcome.GetError().GetMessage()
        << std::endl;
    return false;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ResendConfirmationCode](#)를 참조하세요.

## SignUp

다음 코드 예시는 SignUp의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

Aws::CognitoIdentityProvider::Model::SignUpRequest request;
request.AddUserAttributes(
    Aws::CognitoIdentityProvider::Model::AttributeType().WithName(
        "email").WithValue(email));
request.SetUsername(userName);
request.SetPassword(password);
request.SetClientId(clientID);
Aws::CognitoIdentityProvider::Model::SignUpOutcome outcome =
    client.SignUp(request);

if (outcome.IsSuccess()) {
    std::cout << "The signup request for " << userName << " was successful."
        << std::endl;
}
else if (outcome.GetError().GetErrorType() ==
    Aws::CognitoIdentityProvider::CognitoIdentityProviderErrors::USERNAME_EXISTS) {
    std::cout
        << "The username already exists. Please enter a different
username."
        << std::endl;
    userExists = true;
}
else {
```

```

        std::cerr << "Error with CognitoIdentityProvider::SignUpRequest. "
                    << outcome.GetError().GetMessage()
                    << std::endl;
    return false;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [SignUp](#)를 참조하세요.

## VerifySoftwareToken

다음 코드 예시는 VerifySoftwareToken의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenRequest request;
request.SetUserCode(userCode);
request.SetSession(session);

Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenOutcome outcome =
    client.VerifySoftwareToken(request);

if (outcome.IsSuccess()) {
    std::cout << "Verification of the code was successful."
                << std::endl;
    session = outcome.GetResult().GetSession();
}
else {
    std::cerr << "Error with CognitoIdentityProvider::VerifySoftwareToken. "

```

```

        << outcome.GetError().GetMessage()
        << std::endl;
    return false;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [VerifySoftwareToken](#)을 참조하세요.

## 시나리오

### MFA가 필요한 사용자 풀에 사용자 가입시키기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 사용자 이름, 암호 및 이메일 주소로 사용자를 가입시키고 확인합니다.
- MFA 애플리케이션을 사용자와 연결하여 다중 인증을 설정합니다.
- 암호와 MFA 코드를 사용하여 로그인합니다.

### SDK for C++

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예제 리포지토리](#)에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    //! Scenario that adds a user to an Amazon Cognito user pool.
    /*!
    \sa gettingStartedWithUserPools()
    \param clientID: Client ID associated with an Amazon Cognito user pool.
    \param userPoolID: An Amazon Cognito user pool ID.
    \param clientConfig: Aws client configuration.
    \return bool: Successful completion.
    */
    bool AwsDoc::Cognito::gettingStartedWithUserPools(const Aws::String &clientID,
                                                       const Aws::String &userPoolID,

```

```

const
Aws::Client::ClientConfiguration &clientConfig) {
    printAsterisksLine();
    std::cout
        << "Welcome to the Amazon Cognito example scenario."
        << std::endl;
    printAsterisksLine();

    std::cout
        << "This scenario will add a user to an Amazon Cognito user pool."
        << std::endl;
    const Aws::String userName = askQuestion("Enter a new username: ");
    const Aws::String password = askQuestion("Enter a new password: ");
    const Aws::String email = askQuestion("Enter a valid email for the user: ");

    std::cout << "Signing up " << userName << std::endl;

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);
    bool userExists = false;
    do {
        // 1. Add a user with a username, password, and email address.
        Aws::CognitoIdentityProvider::Model::SignUpRequest request;
        request.AddUserAttributes(
            Aws::CognitoIdentityProvider::Model::AttributeType().WithName(
                "email").WithValue(email));
        request.SetUsername(userName);
        request.SetPassword(password);
        request.SetClientId(clientID);
        Aws::CognitoIdentityProvider::Model::SignUpOutcome outcome =
            client.SignUp(request);

        if (outcome.IsSuccess()) {
            std::cout << "The signup request for " << userName << " was successful."
                << std::endl;
        }
        else if (outcome.GetError().GetErrorType() ==
            Aws::CognitoIdentityProvider::CognitoIdentityProviderErrors::USERNAME_EXISTS) {
            std::cout
                << "The username already exists. Please enter a different
username."
                << std::endl;
            userExists = true;
        }
    } while (userExists);
}

```

```

    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::SignUpRequest. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        return false;
    }
} while (userExists);

printAsterisksLine();
std::cout << "Retrieving status of " << userName << " in the user pool."
          << std::endl;
// 2. Confirm that the user was added to the user pool.
if (!checkAdminUserStatus(userName, userPoolID, client)) {
    return false;
}

std::cout << "A confirmation code was sent to " << email << "." << std::endl;

bool resend = askYesNoQuestion("Would you like to send a new code? (y/n) ");
if (resend) {
    // Request a resend of the confirmation code to the email address.
    (ResendConfirmationCode)
        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeRequest request;
        request.SetUsername(userName);
        request.SetClientId(clientID);

        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeOutcome outcome =
            client.ResendConfirmationCode(request);

        if (outcome.IsSuccess()) {
            std::cout
                << "CognitoIdentityProvider::ResendConfirmationCode was
successful."
                << std::endl;
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::ResendConfirmationCode. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }
}

```

```

printAsterisksLine();

{
    // 4. Send the confirmation code that's received in the email.
    (ConfirmSignUp)
        const Aws::String confirmationCode = askQuestion(
            "Enter the confirmation code that was emailed: ");
        Aws::CognitoIdentityProvider::Model::ConfirmSignUpRequest request;
        request.SetClientId(clientID);
        request.SetConfirmationCode(confirmationCode);
        request.SetUsername(userName);

        Aws::CognitoIdentityProvider::Model::ConfirmSignUpOutcome outcome =
            client.ConfirmSignUp(request);

        if (outcome.IsSuccess()) {
            std::cout << "ConfirmSignup was Successful."
                << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::ConfirmSignUp. "
                << outcome.GetError().GetMessage()
                << std::endl;
            return false;
        }
    }

    std::cout << "Rechecking the status of " << userName << " in the user pool."
        << std::endl;
    if (!checkAdminUserStatus(userName, userPoolID, client)) {
        return false;
    }

    printAsterisksLine();

    std::cout << "Initiating authorization using the username and password."
        << std::endl;

    Aws::String session;
    // 5. Initiate authorization with username and password. (AdminInitiateAuth)
    if (!adminInitiateAuthorization(clientID, userPoolID, userName, password,
        session, client)) {
        return false;
    }
}

```

```

    }

    printAsterisksLine();

    std::cout
        << "Starting setup of time-based one-time password (TOTP) multi-factor
authentication (MFA)."
        << std::endl;

    {
        // 6. Request a setup key for one-time password (TOTP)
        //    multi-factor authentication (MFA). (AssociateSoftwareToken)
        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenRequest request;
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenOutcome outcome =
            client.AssociateSoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout
                << "Enter this setup key into an authenticator app, for example
Google Authenticator."
                << std::endl;
            std::cout << "Setup key: " << outcome.GetResult().GetSecretCode()
                << std::endl;
#ifdef USING_QR
            printAsterisksLine();
            std::cout << "\nOr scan the QR code in the file '" << QR_CODE_PATH <<
                ". "
                << std::endl;

            saveQRCode(std::string("otpauth://totp/") + userName + "?secret=" +
                outcome.GetResult().GetSecretCode());
#endif // USING_QR
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::AssociateSoftwareToken. "
                << outcome.GetError().GetMessage()
                << std::endl;
            return false;
        }
    }
}

```

```

    askQuestion("Type enter to continue...", alwaysTrueTest);

    printAsterisksLine();

    {
        Aws::String userCode = askQuestion(
            "Enter the 6 digit code displayed in the authenticator app: ");

        // 7. Send the MFA code copied from an authenticator app.
    (VerifySoftwareToken)
        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenRequest request;
        request.SetUserCode(userCode);
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenOutcome outcome =
            client.VerifySoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout << "Verification of the code was successful."
                << std::endl;
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::VerifySoftwareToken. "
                << outcome.GetError().GetMessage()
                << std::endl;
            return false;
        }
    }

    printAsterisksLine();
    std::cout << "You have completed the MFA authentication setup." << std::endl;
    std::cout << "Now, sign in." << std::endl;

    // 8. Initiate authorization again with username and password.
    (AdminInitiateAuth)
    if (!adminInitiateAuthorization(clientID, userPoolID, userName, password,
        session, client)) {
        return false;
    }

    Aws::String accessToken;
    {
        Aws::String mfaCode = askQuestion(

```

```

        "Re-enter the 6 digit code displayed in the authenticator app: ");

    // 9. Send a new MFA code copied from an authenticator app.
    (AdminRespondToAuthChallenge)
    Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeRequest
    request;
    request.AddChallengeResponses("USERNAME", userName);
    request.AddChallengeResponses("SOFTWARE_TOKEN_MFA_CODE", mfaCode);
    request.SetChallengeName(

    Aws::CognitoIdentityProvider::Model::ChallengeNameType::SOFTWARE_TOKEN_MFA);
    request.SetClientId(clientID);
    request.SetUserPoolId(userPoolID);
    request.SetSession(session);

    Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeOutcome
    outcome =

        client.AdminRespondToAuthChallenge(request);

    if (outcome.IsSuccess()) {
        std::cout << "Here is the response to the challenge.\n" <<

    outcome.GetResult().GetAuthenticationResult().Jsonize().View().WriteReadable()
        << std::endl;

        accessToken =
    outcome.GetResult().GetAuthenticationResult().GetAccessToken();
    }
    else {
        std::cerr << "Error with
    CognitoIdentityProvider::AdminRespondToAuthChallenge. "
        << outcome.GetError().GetMessage()
        << std::endl;
        return false;
    }

    std::cout << "You have successfully added a user to Amazon Cognito."
        << std::endl;
}

if (askYesNoQuestion("Would you like to delete the user that you just added? (y/
n) ")) {
    // 10. Delete the user that you just added. (DeleteUser)
    Aws::CognitoIdentityProvider::Model::DeleteUserRequest request;

```

```

        request.SetAccessToken(accessToken);

        Aws::CognitoIdentityProvider::Model::DeleteUserOutcome outcome =
            client.DeleteUser(request);

        if (outcome.IsSuccess()) {
            std::cout << "The user " << userName << " was deleted."
                << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::DeleteUser. "
                << outcome.GetError().GetMessage()
                << std::endl;
        }
    }

    return true;
}

//! Routine which checks the user status in an Amazon Cognito user pool.
/*!
    \sa checkAdminUserStatus()
    \param userName: A username.
    \param userPoolID: An Amazon Cognito user pool ID.
    \return bool: Successful completion.
    */
bool AwsDoc::Cognito::checkAdminUserStatus(const Aws::String &userName,
                                            const Aws::String &userPoolID,
                                            const
        Aws::CognitoIdentityProvider::CognitoIdentityProviderClient &client) {
    Aws::CognitoIdentityProvider::Model::AdminGetUserRequest request;
    request.SetUsername(userName);
    request.SetUserPoolId(userPoolID);

    Aws::CognitoIdentityProvider::Model::AdminGetUserOutcome outcome =
        client.AdminGetUser(request);

    if (outcome.IsSuccess()) {
        std::cout << "The status for " << userName << " is " <<

        Aws::CognitoIdentityProvider::Model::UserStatusTypeMapper::GetNameForUserStatusType(
            outcome.GetResult().GetUserStatus()) << std::endl;
        std::cout << "Enabled is " << outcome.GetResult().GetEnabled() << std::endl;
    }
}

```

```

    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminGetUser. "
                   << outcome.GetError().GetMessage()
                   << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which starts authorization of an Amazon Cognito user.
//! This routine requires administrator credentials.
/*!
 \sa adminInitiateAuthorization()
 \param clientID: Client ID of tracked device.
 \param userPoolID: An Amazon Cognito user pool ID.
 \param userName: A username.
 \param password: A password.
 \param sessionResult: String to receive a session token.
 \return bool: Successful completion.
 */
bool AwsDoc::Cognito::adminInitiateAuthorization(const Aws::String &clientID,
                                                  const Aws::String &userPoolID,
                                                  const Aws::String &userName,
                                                  const Aws::String &password,
                                                  Aws::String &sessionResult,
                                                  const
Aws::CognitoIdentityProvider::CognitoIdentityProviderClient &client) {
    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthRequest request;
    request.SetClientId(clientID);
    request.SetUserPoolId(userPoolID);
    request.AddAuthParameters("USERNAME", userName);
    request.AddAuthParameters("PASSWORD", password);
    request.SetAuthFlow(

    Aws::CognitoIdentityProvider::Model::AuthFlowType::ADMIN_USER_PASSWORD_AUTH);

    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthOutcome outcome =
        client.AdminInitiateAuth(request);

    if (outcome.IsSuccess()) {
        std::cout << "Call to AdminInitiateAuth was successful." << std::endl;
        sessionResult = outcome.GetResult().GetSession();
    }
}

```

```

    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminInitiateAuth. "
                    << outcome.GetError().GetMessage()
                    << std::endl;
    }

    return outcome.IsSuccess();
}

```

• API 세부 정보는 AWS SDK for C++ API 참조의 다음 항목을 참조하세요.

- [AdminGetUser](#)
- [AdminInitiateAuth](#)
- [AdminRespondToAuthChallenge](#)
- [AssociateSoftwareToken](#)
- [ConfirmDevice](#)
- [ConfirmSignUp](#)
- [InitiateAuth](#)
- [ListUsers](#)
- [ResendConfirmationCode](#)
- [RespondToAuthChallenge](#)
- [SignUp](#)
- [VerifySoftwareToken](#)

## SDK for C++를 사용한 DynamoDB 예제

다음 코드 예제에서는 DynamoDB와 AWS SDK for C++ 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](#)
- [기본 사항](#)
- [작업](#)
- [시나리오](#)

## 시작하기

### Hello DynamoDB

다음 코드 예제는 DynamoDB를 사용하여 시작하는 방법을 보여 줍니다.

### SDK for C++

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

CMakeLists.txt CMake 파일의 코드입니다.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS dynamodb)

# Set this project's name.
project("hello_dynamodb")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
```

```

set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
"${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory for
    running and debugging.

    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you may
    need to uncomment this

                                # and set the proper subdirectory to the
    executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
"${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR}")
endif ()

add_executable(${PROJECT_NAME}
    hello_dynamodb.cpp)

target_link_libraries(${PROJECT_NAME}
    ${AWSSDK_LINK_LIBRARIES})

```

hello\_dynamodb.cpp 소스 파일의 코드입니다.

```

#include <aws/core/Aws.h>
#include <aws/dynamodb/DynamoDBClient.h>
#include <aws/dynamodb/model/ListTablesRequest.h>
#include <iostream>

/*
 * A "Hello DynamoDB" starter application which initializes an Amazon DynamoDB
 (DynamoDB) client and lists the
 * DynamoDB tables.

```

```

*
*  main function
*
*  Usage: 'hello_dynamodb'
*
*/

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.

    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::DynamoDB::DynamoDBClient dynamodbClient(clientConfig);
        Aws::DynamoDB::Model::ListTablesRequest listTablesRequest;
        listTablesRequest.SetLimit(50);
        do {
            const Aws::DynamoDB::Model::ListTablesOutcome &outcome =
dynamodbClient.ListTables(
                listTablesRequest);
            if (!outcome.IsSuccess()) {
                std::cout << "Error: " << outcome.GetError().GetMessage() <<
std::endl;
                result = 1;
                break;
            }

            for (const auto &tableName: outcome.GetResult().GetTableNames()) {
                std::cout << tableName << std::endl;
            }

            listTablesRequest.SetExclusiveStartTableName(
                outcome.GetResult().GetLastEvaluatedTableName());

        } while (!listTablesRequest.GetExclusiveStartTableName().empty());
    }
}

```

```

    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ListTables](#)를 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 C++

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예제 리포지토리](#)에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```

{
    Aws::Client::ClientConfiguration clientConfig;
    // 1. Create a table with partition: year (N) and sort: title (S).
    (CreateTable)
    if (AwsDoc::DynamoDB::createMoviesDynamoDBTable(clientConfig)) {

        AwsDoc::DynamoDB::dynamodbGettingStartedScenario(clientConfig);

        // 9. Delete the table. (DeleteTable)
        AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(clientConfig);
    }
}

```

```

    }
}

//! Scenario to modify and query a DynamoDB table.
/*!
 \sa dynamodbGettingStartedScenario()
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::DynamoDB::dynamodbGettingStartedScenario(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " "
        << std::endl;
    std::cout << "Welcome to the Amazon DynamoDB getting started demo." <<
std::endl;
    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " "
        << std::endl;

    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    // 2. Add a new movie.
    Aws::String title;
    float rating;
    int year;
    Aws::String plot;
    {
        title = askQuestion(
            "Enter the title of a movie you want to add to the table: ");
        year = askQuestionForInt("What year was it released? ");
        rating = askQuestionForFloatRange("On a scale of 1 - 10, how do you rate it?",
            1, 10);
        plot = askQuestion("Summarize the plot for me: ");

        Aws::DynamoDB::Model::PutItemRequest putItemRequest;
        putItemRequest.SetTableName(MOVIE_TABLE_NAME);

        putItemRequest.AddItem(YEAR_KEY,
            Aws::DynamoDB::Model::AttributeValue().SetN(year));
        putItemRequest.AddItem(TITLE_KEY,
            Aws::DynamoDB::Model::AttributeValue().SetS(title));

        // Create attribute for the info map.
        Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

```

```

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
        Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
            ALLOCATION_TAG.c_str());
        ratingAttribute->SetN(rating);
        infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
        Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
            ALLOCATION_TAG.c_str());
        plotAttribute->SetS(plot);
        infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);

        putItemRequest.AddItem(INFO_KEY, infoMapAttribute);

        Aws::DynamoDB::Model::PutItemOutcome outcome = dynamoClient.PutItem(
            putItemRequest);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to add an item: " <<
            outcome.GetError().GetMessage()
                << std::endl;
            return false;
        }
    }

    std::cout << "\nAdded '" << title << "' to '" << MOVIE_TABLE_NAME << "'."
        << std::endl;

// 3. Update the rating and plot of the movie by using an update expression.
{
    rating = askQuestionForFloatRange(
        Aws::String("\nLet's update your movie.\nYou rated it ") +
        std::to_string(rating)
        + ", what new rating would you give it? ", 1, 10);
    plot = askQuestion(Aws::String("You summarized the plot as '" + plot +
        "'.\nWhat would you say now? ");

    Aws::DynamoDB::Model::UpdateItemRequest request;
    request.SetTableName(MOVIE_TABLE_NAME);
    request.AddKey(TITLE_KEY,
        Aws::DynamoDB::Model::AttributeValue().SetS(title));
    request.AddKey(YEAR_KEY, Aws::DynamoDB::Model::AttributeValue().SetN(year));
    std::stringstream expressionStream;
    expressionStream << "set " << INFO_KEY << "." << RATING_KEY << " =:r, "

```

```

        << INFO_KEY << "." << PLOT_KEY << " =:p";
    request.SetUpdateExpression(expressionStream.str());
    request.SetExpressionAttributeValues({
        {":r",
    Aws::DynamoDB::Model::AttributeValue().SetN(
        rating)},
        {":p",
    Aws::DynamoDB::Model::AttributeValue().SetS(
        plot)}}
    });

    request.SetReturnValues(Aws::DynamoDB::Model::ReturnValue::UPDATED_NEW);

    const Aws::DynamoDB::Model::UpdateItemOutcome &result =
    dynamoClient.UpdateItem(
        request);
    if (!result.IsSuccess()) {
        std::cerr << "Error updating movie " + result.GetError().GetMessage()
            << std::endl;
        return false;
    }
}

std::cout << "\nUpdated '" << title << "' with new attributes:" << std::endl;

// 4. Put 250 movies in the table from moviedata.json.
{
    std::cout << "Adding movies from a json file to the database." << std::endl;
    const size_t MAX_SIZE_FOR_BATCH_WRITE = 25;
    const size_t MOVIES_TO_WRITE = 10 * MAX_SIZE_FOR_BATCH_WRITE;
    Aws::String jsonString = getMovieJSON();
    if (!jsonString.empty()) {
        Aws::Utils::Json::JsonValue json(jsonString);
        Aws::Utils::Array<Aws::Utils::Json::JsonView> movieJsons =
    json.View().AsArray();
        Aws::Vector<Aws::DynamoDB::Model::WriteRequest> writeRequests;

        // To add movies with a cross-section of years, use an appropriate
increment
        // value for iterating through the database.
        size_t increment = movieJsons.GetLength() / MOVIES_TO_WRITE;
        for (size_t i = 0; i < movieJsons.GetLength(); i += increment) {
            writeRequests.push_back(Aws::DynamoDB::Model::WriteRequest());

```

```

        Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> putItems
= movieJsonViewToAttributeMap(
            movieJsons[i]);
    Aws::DynamoDB::Model::PutRequest putRequest;
    putRequest.SetItem(putItems);
    writeRequests.back().SetPutRequest(putRequest);
    if (writeRequests.size() == MAX_SIZE_FOR_BATCH_WRITE) {
        Aws::DynamoDB::Model::BatchWriteItemRequest request;
        request.AddRequestItems(MOVIE_TABLE_NAME, writeRequests);
        const Aws::DynamoDB::Model::BatchWriteItemOutcome &outcome =
dynamoClient.BatchWriteItem(
            request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Unable to batch write movie data: "
                << outcome.GetError().GetMessage()
                << std::endl;
            writeRequests.clear();
            break;
        }
        else {
            std::cout << "Added batch of " << writeRequests.size()
                << " movies to the database."
                << std::endl;
        }
        writeRequests.clear();
    }
}

}

}

std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " "
    << std::endl;

// 5. Get a movie by Key (partition + sort).
{
    Aws::String titleToGet("King Kong");
    Aws::String answer = askQuestion(Aws::String(
        "Let's move on...Would you like to get info about '" + titleToGet +
        "'? (y/n) ");
    if (answer == "y") {
        Aws::DynamoDB::Model::GetItemRequest request;
        request.SetTableName(MOVIE_TABLE_NAME);
        request.AddKey(TITLE_KEY,
            Aws::DynamoDB::Model::AttributeValue().SetS(titleToGet));
    }
}

```

```

        request.AddKey(YEAR_KEY,
        Aws::DynamoDB::Model::AttributeValue().SetN(1933));

        const Aws::DynamoDB::Model::GetItemOutcome &result =
dynamoClient.GetItem(
            request);
        if (!result.IsSuccess()) {
            std::cerr << "Error " << result.GetError().GetMessage();
        }
        else {
            const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
&item = result.GetResult().GetItem();
            if (!item.empty()) {
                std::cout << "\nHere's what I found:" << std::endl;
                printMovieInfo(item);
            }
            else {
                std::cout << "\nThe movie was not found in the database."
                << std::endl;
            }
        }
    }
}

// 6. Use Query with a key condition expression to return all movies
//    released in a given year.
Aws::String doAgain = "n";
do {
    Aws::DynamoDB::Model::QueryRequest req;

    req.SetTableName(MOVIE_TABLE_NAME);

    // "year" is a DynamoDB reserved keyword and must be replaced with an
    // expression attribute name.
    req.SetKeyConditionExpression("#dynobase_year = :valueToMatch");
    req.SetExpressionAttributeNames({{"#dynobase_year", YEAR_KEY}});

    int yearToMatch = askQuestionForIntRange(
        "\nLet's get a list of movies released in"
        " a given year. Enter a year between 1972 and 2018 ",
        1972, 2018);
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> attributeValues;
    attributeValues.emplace(":valueToMatch",
        Aws::DynamoDB::Model::AttributeValue().SetN(

```

```

        yearToMatch));
req.SetExpressionAttributeValues(attributeValues);

const Aws::DynamoDB::Model::QueryOutcome &result = dynamoClient.Query(req);
if (result.IsSuccess()) {
    const Aws::Vector<Aws::Map<Aws::String,
    Aws::DynamoDB::Model::AttributeValue>> &items = result.GetResult().GetItems();
    if (!items.empty()) {
        std::cout << "\nThere were " << items.size()
            << " movies in the database from "
            << yearToMatch << "." << std::endl;
        for (const auto &item: items) {
            printMovieInfo(item);
        }
        doAgain = "n";
    }
    else {
        std::cout << "\nNo movies from " << yearToMatch
            << " were found in the database"
            << std::endl;
        doAgain = askQuestion(Aws::String("Try another year? (y/n) "));
    }
}
else {
    std::cerr << "Failed to Query items: " << result.GetError().GetMessage()
        << std::endl;
}

} while (doAgain == "y");

// 7. Use Scan to return movies released within a range of years.
// Show how to paginate data using ExclusiveStartKey. (Scan +
FilterExpression)
{
    int startYear = askQuestionForIntRange("\nNow let's scan a range of years "
        "for movies in the database. Enter a
start year: ",
        1972, 2018);
    int endYear = askQuestionForIntRange("\nEnter an end year: ",
        startYear, 2018);
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
exclusiveStartKey;
    do {
        Aws::DynamoDB::Model::ScanRequest scanRequest;

```

```

scanRequest.SetTableName(MOVIE_TABLE_NAME);
scanRequest.SetFilterExpression(
    "#dynobase_year >= :startYear AND #dynobase_year <= :endYear");
scanRequest.SetExpressionAttributeNames({{"#dynobase_year", YEAR_KEY}});

Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
attributeValues;
attributeValues.emplace(":startYear",
    Aws::DynamoDB::Model::AttributeValue().SetN(
        startYear));
attributeValues.emplace(":endYear",
    Aws::DynamoDB::Model::AttributeValue().SetN(
        endYear));
scanRequest.SetExpressionAttributeValues(attributeValues);

if (!exclusiveStartKey.empty()) {
    scanRequest.SetExclusiveStartKey(exclusiveStartKey);
}

const Aws::DynamoDB::Model::ScanOutcome &result = dynamoClient.Scan(
    scanRequest);
if (result.IsSuccess()) {
    const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetResult().GetItems();
    if (!items.empty()) {
        std::stringstream stringStream;
        stringStream << "\nFound " << items.size() << " movies in one
scan."

        << " How many would you like to see? ";
        size_t count = askQuestionForInt(stringStream.str());
        for (size_t i = 0; i < count && i < items.size(); ++i) {
            printMovieInfo(items[i]);
        }
    }
    else {
        std::cout << "\nNo movies in the database between " << startYear
<<

        " and " << endYear << "." << std::endl;
    }

    exclusiveStartKey = result.GetResult().GetLastEvaluatedKey();
    if (!exclusiveStartKey.empty()) {
        std::cout << "Not all movies were retrieved. Scanning for more."
<< std::endl;
    }
}

```

```

        }
        else {
            std::cout << "All movies were retrieved with this scan."
                << std::endl;
        }
    }
    else {
        std::cerr << "Failed to Scan movies: "
            << result.GetError().GetMessage() << std::endl;
    }
} while (!exclusiveStartKey.empty());
}

// 8. Delete a movie. (DeleteItem)
{
    std::stringstream stringStream;
    stringStream << "\nWould you like to delete the movie " << title
        << " from the database? (y/n) ";
    Aws::String answer = askQuestion(stringStream.str());
    if (answer == "y") {
        Aws::DynamoDB::Model::DeleteItemRequest request;
        request.AddKey(YEAR_KEY,
            Aws::DynamoDB::Model::AttributeValue().SetN(year));
        request.AddKey(TITLE_KEY,
            Aws::DynamoDB::Model::AttributeValue().SetS(title));
        request.SetTableName(MOVIE_TABLE_NAME);

        const Aws::DynamoDB::Model::DeleteItemOutcome &result =
            dynamoClient.DeleteItem(
                request);
        if (result.IsSuccess()) {
            std::cout << "\nRemoved \"" << title << "\" from the database."
                << std::endl;
        }
        else {
            std::cerr << "Failed to delete the movie: "
                << result.GetError().GetMessage()
                << std::endl;
        }
    }
}

return true;
}

```

```

//! Routine to convert a JsonView object to an attribute map.
/*!
    \sa movieJsonViewToAttributeMap()
    \param jsonView: Json view object.
    \return map: Map that can be used in a DynamoDB request.
*/
Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
AwsDoc::DynamoDB::movieJsonViewToAttributeMap(
    const Aws::Utils::Json::JsonView &jsonView) {
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> result;

    if (jsonView.KeyExists(YEAR_KEY)) {
        result[YEAR_KEY].SetN(jsonView.GetInteger(YEAR_KEY));
    }
    if (jsonView.KeyExists(TITLE_KEY)) {
        result[TITLE_KEY].SetS(jsonView.GetString(TITLE_KEY));
    }
    if (jsonView.KeyExists(INFO_KEY)) {
        Aws::Map<Aws::String, const
std::shared_ptr<Aws::DynamoDB::Model::AttributeValue>> infoMap;
        Aws::Utils::Json::JsonView infoView = jsonView.GetObject(INFO_KEY);
        if (infoView.KeyExists(RATING_KEY)) {
            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> attributeValue =
std::make_shared<Aws::DynamoDB::Model::AttributeValue>();
            attributeValue->SetN(infoView.GetDouble(RATING_KEY));
            infoMap.emplace(std::make_pair(RATING_KEY, attributeValue));
        }
        if (infoView.KeyExists(PLOT_KEY)) {
            std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> attributeValue =
std::make_shared<Aws::DynamoDB::Model::AttributeValue>();
            attributeValue->SetS(infoView.GetString(PLOT_KEY));
            infoMap.emplace(std::make_pair(PLOT_KEY, attributeValue));
        }

        result[INFO_KEY].SetM(infoMap);
    }

    return result;
}

//! Create a DynamoDB table to be used in sample code scenarios.
/*!
    \sa createMoviesDynamoDBTable()

```

```

    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::createMoviesDynamoDBTable(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    bool movieTableAlreadyExisted = false;

    {
        Aws::DynamoDB::Model::CreateTableRequest request;

        Aws::DynamoDB::Model::AttributeDefinition yearAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(YEAR_KEY);
        yearAttributeDefinition.SetAttributeType(
            Aws::DynamoDB::Model::ScalarAttributeType::N);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::AttributeDefinition titleAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(TITLE_KEY);
        yearAttributeDefinition.SetAttributeType(
            Aws::DynamoDB::Model::ScalarAttributeType::S);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::KeySchemaElement yearKeySchema;
        yearKeySchema.WithAttributeName(YEAR_KEY).WithKeyType(
            Aws::DynamoDB::Model::KeyType::HASH);
        request.AddKeySchema(yearKeySchema);

        Aws::DynamoDB::Model::KeySchemaElement titleKeySchema;
        yearKeySchema.WithAttributeName(TITLE_KEY).WithKeyType(
            Aws::DynamoDB::Model::KeyType::RANGE);
        request.AddKeySchema(yearKeySchema);

        Aws::DynamoDB::Model::ProvisionedThroughput throughput;
        throughput.WithReadCapacityUnits(
            PROVISIONED_THROUGHPUT_UNITS).WithWriteCapacityUnits(
            PROVISIONED_THROUGHPUT_UNITS);
        request.SetProvisionedThroughput(throughput);
        request.SetTableName(MOVIE_TABLE_NAME);

        std::cout << "Creating table '" << MOVIE_TABLE_NAME << "'..." << std::endl;
        const Aws::DynamoDB::Model::CreateTableOutcome &result =
            dynamoClient.CreateTable(

```

```

        request);
    if (!result.IsSuccess()) {
        if (result.GetError().GetErrorType() ==
            Aws::DynamoDB::DynamoDBErrors::RESOURCE_IN_USE) {
            std::cout << "Table already exists." << std::endl;
            movieTableAlreadyExisted = true;
        }
        else {
            std::cerr << "Failed to create table: "
                << result.GetError().GetMessage();
            return false;
        }
    }
}

// Wait for table to become active.
if (!movieTableAlreadyExisted) {
    std::cout << "Waiting for table '" << MOVIE_TABLE_NAME
        << "' to become active...." << std::endl;
    if (!AwsDoc::DynamoDB::waitTableActive(MOVIE_TABLE_NAME,
clientConfiguration)) {
        return false;
    }
    std::cout << "Table '" << MOVIE_TABLE_NAME << "' created and active."
        << std::endl;
}

return true;
}

//! Delete the DynamoDB table used for sample code scenarios.
/*!
    \sa deleteMoviesDynamoDBTable()
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(MOVIE_TABLE_NAME);

```

```

    const Aws::DynamoDB::Model::DeleteTableOutcome &result =
dynamoClient.DeleteTable(
    request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
            << result.GetResult().GetTableDescription().GetTableName()
            << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
            << std::endl;
    }

    return result.IsSuccess();
}

//! Query a newly created DynamoDB table until it is active.
/*!
    \sa waitTableActive()
    \param waitTableActive: The DynamoDB table's name.
    \param dynamoClient: A DynamoDB client.
    \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
&dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
dynamoClient.DescribeTable(
            request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
        }
    }
}

```

```
        else {  
            return true;  
        }  
    }  
    else {  
        std::cerr << "Error DynamoDB::waitTableActive "  
                    << result.GetError().GetMessage() << std::endl;  
        return false;  
    }  
    count++;  
}  
return false;  
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 다음 주제를 참조하세요.
  - [BatchWriteItem](#)
  - [CreateTable](#)
  - [DeleteItem](#)
  - [DeleteTable](#)
  - [DescribeTable](#)
  - [GetItem](#)
  - [PutItem](#)
  - [Query](#)
  - [Scan](#)
  - [UpdateItem](#)

## 작업

### BatchExecuteStatement

다음 코드 예시는 BatchExecuteStatement의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

INSERT 문 배치를 사용하여 항목을 추가합니다.

```
// 2. Add multiple movies using "Insert" statements. (BatchExecuteStatement)
Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

std::vector<Aws::String> titles;
std::vector<float> ratings;
std::vector<int> years;
std::vector<Aws::String> plots;
Aws::String doAgain = "n";
do {
    Aws::String aTitle = askQuestion(
        "Enter the title of a movie you want to add to the table: ");
    titles.push_back(aTitle);
    int aYear = askQuestionForInt("What year was it released? ");
    years.push_back(aYear);
    float aRating = askQuestionForFloatRange(
        "On a scale of 1 - 10, how do you rate it? ",
        1, 10);
    ratings.push_back(aRating);
    Aws::String aPlot = askQuestion("Summarize the plot for me: ");
    plots.push_back(aPlot);

    doAgain = askQuestion(Aws::String("Would you like to add more movies? (y/n)
"));
} while (doAgain == "y");

std::cout << "Adding " << titles.size()
    << (titles.size() == 1 ? " movie " : " movies ")
    << "to the table using a batch \"INSERT\" statement." << std::endl;

{
    Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
        titles.size());
```

```

std::stringstream sqlStream;
sqlStream << "INSERT INTO \"" << MOVIE_TABLE_NAME << "\" VALUE {'"
    << TITLE_KEY << "': ?, '" << YEAR_KEY << "': ?, '"
    << INFO_KEY << "': ?}";

std::string sql(sqlStream.str());

for (size_t i = 0; i < statements.size(); ++i) {
    statements[i].SetStatement(sql);

    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
    attributes.push_back(
        Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));

    // Create attribute for the info map.
    Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

    std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
    Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
        ALLOCATION_TAG.c_str());
    ratingAttribute->SetN(ratings[i]);
    infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

    std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
    Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
        ALLOCATION_TAG.c_str());
    plotAttribute->SetS(plots[i]);
    infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
    attributes.push_back(infoMapAttribute);
    statements[i].SetParameters(attributes);
}

Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

request.SetStatements(statements);

Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
    request);
if (!outcome.IsSuccess()) {
    std::cerr << "Failed to add the movies: " <<
outcome.GetError().GetMessage()

```

```

        << std::endl;
    return false;
}
}

```

SELECT 문 배치를 사용하여 항목을 가져옵니다.

```

// 3. Get the data for multiple movies using "Select" statements.
(BatchExecuteStatement)
{
    Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
        titles.size());
    std::stringstream sqlStream;
    sqlStream << "SELECT * FROM \"" << MOVIE_TABLE_NAME << "\" WHERE "
        << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

    std::string sql(sqlStream.str());

    for (size_t i = 0; i < statements.size(); ++i) {
        statements[i].SetStatement(sql);
        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
        statements[i].SetParameters(attributes);
    }

    Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

    request.SetStatements(statements);

    Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
    dynamoClient.BatchExecuteStatement(
        request);
    if (outcome.IsSuccess()) {
        const Aws::DynamoDB::Model::BatchExecuteStatementResult &result =
        outcome.GetResult();

        const Aws::Vector<Aws::DynamoDB::Model::BatchStatementResponse>
        &responses = result.GetResponses();
    }
}

```

```

        for (const Aws::DynamoDB::Model::BatchStatementResponse &response:
responses) {
            const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
&item = response.GetItem();

            printMovieInfo(item);
        }
    }
    else {
        std::cerr << "Failed to retrieve the movie information: "
            << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
}

```

UPDATE 문 배치를 사용하여 항목을 업데이트합니다.

```

// 4. Update the data for multiple movies using "Update" statements.
(BatchExecuteStatement)

for (size_t i = 0; i < titles.size(); ++i) {
    ratings[i] = askQuestionForFloatRange(
        Aws::String("\nLet's update your the movie, \"" + titles[i] +
            ".\nYou rated it " + std::to_string(ratings[i])
            + ", what new rating would you give it? ", 1, 10);
}

std::cout << "Updating the movie with a batch \"UPDATE\" statement." <<
std::endl;

{
    Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
        titles.size());

    std::stringstream sqlStream;
    sqlStream << "UPDATE \"" << MOVIE_TABLE_NAME << "\" SET "
        << INFO_KEY << "." << RATING_KEY << "=? WHERE "
        << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";

    std::string sql(sqlStream.str());
}

```

```

    for (size_t i = 0; i < statements.size(); ++i) {
        statements[i].SetStatement(sql);

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetN(ratings[i]));
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
        statements[i].SetParameters(attributes);
    }

    Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

    request.SetStatements(statements);
    Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
    dynamoClient.BatchExecuteStatement(
        request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to update movie information: "
            << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
}

```

DELETE 문 배치를 사용하여 항목을 삭제합니다.

```

// 6. Delete multiple movies using "Delete" statements. (BatchExecuteStatement)
{
    Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
        titles.size());
    std::stringstream sqlStream;
    sqlStream << "DELETE FROM \"" << MOVIE_TABLE_NAME << "\" WHERE "
        << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

    std::string sql(sqlStream.str());

    for (size_t i = 0; i < statements.size(); ++i) {
        statements[i].SetStatement(sql);
        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(

```

```

        Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
    statements[i].SetParameters(attributes);
}

Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

request.SetStatements(statements);

Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
    request);

if (!outcome.IsSuccess()) {
    std::cerr << "Failed to delete the movies: "
                << outcome.GetError().GetMessage() << std::endl;
    return false;
}
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [BatchExecuteStatement](#)를 참조하세요.

## BatchGetItem

다음 코드 예시는 BatchGetItem의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Batch get items from different Amazon DynamoDB tables.
/*!
    \sa batchGetItem()
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */

```

```

bool AwsDoc::DynamoDB::batchGetItem(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::BatchGetItemRequest request;

    // Table1: Forum.
    Aws::String table1Name = "Forum";
    Aws::DynamoDB::Model::KeysAndAttributes table1KeysAndAttributes;

    // Table1: Projection expression.
    table1KeysAndAttributes.SetProjectionExpression("#n, Category, Messages, #v");

    // Table1: Expression attribute names.
    Aws::Http::HeaderValueCollection headerValueCollection;
    headerValueCollection.emplace("#n", "Name");
    headerValueCollection.emplace("#v", "Views");
    table1KeysAndAttributes.SetExpressionAttributeNames(headerValueCollection);

    // Table1: Set key name, type, and value to search.
    std::vector<Aws::String> nameValues = {"Amazon DynamoDB", "Amazon S3"};
    for (const Aws::String &name: nameValues) {
        Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> keys;
        Aws::DynamoDB::Model::AttributeValue key;
        key.SetS(name);
        keys.emplace("Name", key);
        table1KeysAndAttributes.AddKeys(keys);
    }

    Aws::Map<Aws::String, Aws::DynamoDB::Model::KeysAndAttributes> requestItems;
    requestItems.emplace(table1Name, table1KeysAndAttributes);

    // Table2: ProductCatalog.
    Aws::String table2Name = "ProductCatalog";
    Aws::DynamoDB::Model::KeysAndAttributes table2KeysAndAttributes;
    table2KeysAndAttributes.SetProjectionExpression("Title, Price, Color");

    // Table2: Set key name, type, and value to search.
    std::vector<Aws::String> idValues = {"102", "103", "201"};
    for (const Aws::String &id: idValues) {
        Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> keys;
        Aws::DynamoDB::Model::AttributeValue key;
        key.SetN(id);
        keys.emplace("Id", key);
    }
}

```

```

        table2KeysAndAttributes.AddKeys(keys);
    }

    requestItems.emplace(table2Name, table2KeysAndAttributes);

    bool result = true;
    do { // Use a do loop to handle pagination.
        request.SetRequestItems(requestItems);
        const Aws::DynamoDB::Model::BatchGetItemOutcome &outcome =
dynamoClient.BatchGetItem(
            request);

        if (outcome.IsSuccess()) {
            for (const auto &responsesMapEntry: outcome.GetResult().GetResponses())
            {
                Aws::String tableName = responsesMapEntry.first;
                const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &tableResults = responsesMapEntry.second;
                std::cout << "Retrieved " << tableResults.size()
                    << " responses for table '" << tableName << "'.\n"
                    << std::endl;
                if (tableName == "Forum") {

                    std::cout << "Name | Category | Message | Views" << std::endl;
                    for (const Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue> &item: tableResults) {
                        std::cout << item.at("Name").GetS() << " | ";
                        std::cout << item.at("Category").GetS() << " | ";
                        std::cout << (item.count("Message") == 0 ? "" : item.at(
                            "Messages").GetN()) << " | ";
                        std::cout << (item.count("Views") == 0 ? "" : item.at(
                            "Views").GetN()) << std::endl;
                    }
                }
            }
            else {
                std::cout << "Title | Price | Color" << std::endl;
                for (const Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue> &item: tableResults) {
                    std::cout << item.at("Title").GetS() << " | ";
                    std::cout << (item.count("Price") == 0 ? "" : item.at(
                        "Price").GetN());
                    if (item.count("Color")) {
                        std::cout << " | ";

```

```

        for (const
std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> &listItem: item.at(
            "Color").GetL())
            std::cout << listItem->GetS() << " ";
        }
        std::cout << std::endl;
    }
    std::cout << std::endl;
}

// If necessary, repeat request for remaining items.
requestItems = outcome.GetResult().GetUnprocessedKeys();
}
else {
    std::cerr << "Batch get item failed: " <<
outcome.GetError().GetMessage()
        << std::endl;
    result = false;
    break;
}
} while (!requestItems.empty());

return result;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [BatchGetItem](#)을 참조하세요.

## BatchWriteItem

다음 코드 예시는 BatchWriteItem의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Batch write items from a JSON file.

```

```

/!*
 \sa batchWriteItem()
 \param jsonFilePath: JSON file path.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */

/*
 * The input for this routine is a JSON file that you can download from the
 following URL:
 * https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SampleData.html.
 *
 * The JSON data uses the BatchWriteItem API request syntax. The JSON strings are
 * converted to AttributeValue objects. These AttributeValue objects will then
 generate
 * JSON strings when constructing the BatchWriteItem request, essentially outputting
 * their input.
 *
 * This is perhaps an artificial example, but it demonstrates the APIs.
 */

bool AwsDoc::DynamoDB::batchWriteItem(const Aws::String &jsonFilePath,
                                       const Aws::Client::ClientConfiguration
&clientConfiguration) {
    std::ifstream fileStream(jsonFilePath);

    if (!fileStream) {
        std::cerr << "Error: could not open file '" << jsonFilePath << "'."
                  << std::endl;
    }

    std::stringstream stringStream;
    stringStream << fileStream.rdbuf();
    Aws::Utils::Json::JsonValue jsonValue(stringStream);

    Aws::DynamoDB::Model::BatchWriteItemRequest batchWriteItemRequest;
    Aws::Map<Aws::String, Aws::Utils::Json::JsonView> level1Map =
jsonValue.View().GetAllObjects();
    for (const auto &level1Entry: level1Map) {
        const Aws::Utils::Json::JsonView &entriesView = level1Entry.second;
        const Aws::String &tableName = level1Entry.first;
        // The JSON entries at this level are as follows:
        // key - table name
        // value - list of request objects

```

```

        if (!entriesView.IsListType()) {
            std::cerr << "Error: JSON file entry '"
                << tableName << "' is not a list." << std::endl;
            continue;
        }

        Aws::Utils::Array<Aws::Utils::Json::JsonValue> entries =
entriesView.AsArray();

        Aws::Vector<Aws::DynamoDB::Model::WriteRequest> writeRequests;
        if (AwsDoc::DynamoDB::addWriteRequests(tableName, entries,
            writeRequests)) {
            batchWriteItemRequest.AddRequestItems(tableName, writeRequests);
        }
    }

    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::BatchWriteItemOutcome outcome =
dynamoClient.BatchWriteItem(
    batchWriteItemRequest);

    if (outcome.IsSuccess()) {
        std::cout << "DynamoDB::BatchWriteItem was successful." << std::endl;
    }
    else {
        std::cerr << "Error with DynamoDB::BatchWriteItem. "
            << outcome.GetError().GetMessage()
            << std::endl;
        return false;
    }

    return outcome.IsSuccess();
}

//! Convert requests in JSON format to a vector of WriteRequest objects.
/*!
    \sa addWriteRequests()
    \param tableName: Name of the table for the write operations.
    \param requestsJson: Request data in JSON format.
    \param writeRequests: Vector to receive the WriteRequest objects.
    \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::addWriteRequests(const Aws::String &tableName,

```

```

        const
        Aws::Utils::Array<Aws::Utils::Json::JsonValue> &requestsJson,

        Aws::Vector<Aws::DynamoDB::Model::WriteRequest> &writeRequests) {
    for (size_t i = 0; i < requestsJson.GetLength(); ++i) {
        const Aws::Utils::Json::JsonValue &requestsEntry = requestsJson[i];
        if (!requestsEntry.IsObject()) {
            std::cerr << "Error: incorrect requestsEntry type "
                << requestsEntry.WriteReadable() << std::endl;
            return false;
        }

        Aws::Map<Aws::String, Aws::Utils::Json::JsonValue> requestsMap =
        requestsEntry.GetAllObjects();

        for (const auto &request: requestsMap) {
            const Aws::String &requestType = request.first;
            const Aws::Utils::Json::JsonValue &requestJsonView = request.second;

            if (requestType == "PutRequest") {
                if (!requestJsonView.ValueExists("Item")) {
                    std::cerr << "Error: item key missing for requests "
                        << requestJsonView.WriteReadable() << std::endl;
                    return false;
                }
                Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
attributes;
                if (!getAttributeObjectsMap(requestJsonView.GetObject("Item"),
                    attributes)) {
                    std::cerr << "Error getting attributes "
                        << requestJsonView.WriteReadable() << std::endl;
                    return false;
                }

                Aws::DynamoDB::Model::PutRequest putRequest;
                putRequest.SetItem(attributes);
                writeRequests.push_back(
                    Aws::DynamoDB::Model::WriteRequest().WithPutRequest(
                        putRequest));
            }
            else {
                std::cerr << "Error: unimplemented request type '" << requestType
                    << "'." << std::endl;
            }
        }
    }
}

```

```

    }
}

return true;
}

//! Generate a map of AttributeValue objects from JSON records.
/*!
 \sa getAttributeObjectsMap()
 \param jsonView: JSONView of attribute records.
 \param writeRequests: Map to receive the AttributeValue objects.
 \return bool: Function succeeded.
 */
bool
AwsDoc::DynamoDB::getAttributeObjectsMap(const Aws::Utils::Json::JsonView &jsonView,
                                           Aws::Map<Aws::String,
                                           Aws::DynamoDB::Model::AttributeValue> &attributes) {
    Aws::Map<Aws::String, Aws::Utils::Json::JsonView> objectsMap =
    jsonView.GetAllObjects();
    for (const auto &entry: objectsMap) {
        const Aws::String &attributeKey = entry.first;
        const Aws::Utils::Json::JsonView &attributeJsonView = entry.second;

        if (!attributeJsonView.IsObject()) {
            std::cerr << "Error: attribute not an object "
                      << attributeJsonView.WriteReadable() << std::endl;
            return false;
        }

        attributes.emplace(attributeKey,
                           Aws::DynamoDB::Model::AttributeValue(attributeJsonView));
    }

    return true;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [BatchWriteItem](#)을 참조하세요.

## CreateTable

다음 코드 예시는 CreateTable의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Create an Amazon DynamoDB table.
/*!
    \sa createTable()
    \param tableName: Name for the DynamoDB table.
    \param primaryKey: Primary key for the DynamoDB table.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::createTable(const Aws::String &tableName,
                                    const Aws::String &primaryKey,
                                    const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    std::cout << "Creating table " << tableName <<
        " with a simple primary key: \"" << primaryKey << "\"." << std::endl;

    Aws::DynamoDB::Model::CreateTableRequest request;

    Aws::DynamoDB::Model::AttributeDefinition hashKey;
    hashKey.SetAttributeName(primaryKey);
    hashKey.SetAttributeType(Aws::DynamoDB::Model::ScalarAttributeType::S);
    request.AddAttributeDefinitions(hashKey);

    Aws::DynamoDB::Model::KeySchemaElement keySchemaElement;
    keySchemaElement.WithAttributeName(primaryKey).WithKeyType(
        Aws::DynamoDB::Model::KeyType::HASH);
    request.AddKeySchema(keySchemaElement);

    Aws::DynamoDB::Model::ProvisionedThroughput throughput;
    throughput.WithReadCapacityUnits(5).WithWriteCapacityUnits(5);
    request.SetProvisionedThroughput(throughput);
    request.SetTableName(tableName);

```

```

    const Aws::DynamoDB::Model::CreateTableOutcome &outcome =
dynamoClient.CreateTable(
    request);
    if (outcome.IsSuccess()) {
        std::cout << "Table \""
            << outcome.GetResult().GetTableDescription().GetTableName() <<
            " created!" << std::endl;
    }
    else {
        std::cerr << "Failed to create table: " << outcome.GetError().GetMessage()
            << std::endl;
        return false;
    }

    return waitTableActive(tableName, dynamoClient);
}

```

테이블이 활성화될 때까지 대기하는 코드입니다.

```

//! Query a newly created DynamoDB table until it is active.
/*!
    \sa waitTableActive()
    \param waitTableActive: The DynamoDB table's name.
    \param dynamoClient: A DynamoDB client.
    \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                       const Aws::DynamoDB::DynamoDBClient
&dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
dynamoClient.DescribeTable(
            request);
        if (result.IsSuccess()) {

```

```

        Aws::DynamoDB::Model::TableStatus status =
result.GetResult().GetTable().GetTableStatus();

        if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
            std::this_thread::sleep_for(std::chrono::seconds(1));
        }
        else {
            return true;
        }
    }
    else {
        std::cerr << "Error DynamoDB::waitTableActive "
                    << result.GetError().GetMessage() << std::endl;
        return false;
    }
    count++;
}
return false;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CreateTable](#)을 참조하세요.

## DeleteItem

다음 코드 예시는 DeleteItem의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Delete an item from an Amazon DynamoDB table.
/*!
    \sa deleteItem()
    \param tableName: The table name.
    \param partitionKey: The partition key.
    \param partitionValue: The value for the partition key.

```

```

    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */

bool AwsDoc::DynamoDB::deleteItem(const Aws::String &tableName,
                                   const Aws::String &partitionKey,
                                   const Aws::String &partitionValue,
                                   const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteItemRequest request;

    request.AddKey(partitionKey,
                   Aws::DynamoDB::Model::AttributeValue().SetS(partitionValue));
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::DeleteItemOutcome &outcome =
dynamoClient.DeleteItem(
    request);
    if (outcome.IsSuccess()) {
        std::cout << "Item \"" << partitionValue << "\" deleted!" << std::endl;
    }
    else {
        std::cerr << "Failed to delete item: " << outcome.GetError().GetMessage()
        << std::endl;
        return false;
    }

    return waitTableActive(tableName, dynamoClient);
}

```

테이블이 활성화될 때까지 대기하는 코드입니다.

```

//! Query a newly created DynamoDB table until it is active.
/*!
    \sa waitTableActive()
    \param waitTableActive: The DynamoDB table's name.
    \param dynamoClient: A DynamoDB client.
    \return bool: Function succeeded.
    */
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,

```

```

const Aws::DynamoDB::DynamoDBClient
&dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
dynamoClient.DescribeTable(
            request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteItem](#)을 참조하세요.

## DeleteTable

다음 코드 예시는 DeleteTable의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Delete an Amazon DynamoDB table.
/*!
    \sa deleteTable()
    \param tableName: The DynamoDB table name.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::deleteTable(const Aws::String &tableName,
                                    const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result =
dynamoClient.DeleteTable(
    request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
                    << result.GetResult().GetTableDescription().GetTableName()
                    << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
                    << std::endl;
    }

    return result.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteTable](#)을 참조하세요.

## DescribeTable

다음 코드 예시는 DescribeTable의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Describe an Amazon DynamoDB table.
/*!
    \sa describeTable()
    \param tableName: The DynamoDB table name.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::describeTable(const Aws::String &tableName,
                                     const Aws::Client::ClientConfiguration
                                     &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    const Aws::DynamoDB::Model::DescribeTableOutcome &outcome =
    dynamoClient.DescribeTable(
        request);

    if (outcome.IsSuccess()) {
        const Aws::DynamoDB::Model::TableDescription &td =
    outcome.GetResult().GetTable();
        std::cout << "Table name   : " << td.GetTableName() << std::endl;
        std::cout << "Table ARN    : " << td.GetTableArn() << std::endl;
        std::cout << "Status      : "
            << Aws::DynamoDB::Model::TableStatusMapper::GetNameForTableStatus(
                td.GetTableStatus()) << std::endl;
        std::cout << "Item count   : " << td.GetItemCount() << std::endl;
        std::cout << "Size (bytes): " << td.GetTableSizeBytes() << std::endl;
    }
}

```

```

        const Aws::DynamoDB::Model::ProvisionedThroughputDescription &ptd =
td.GetProvisionedThroughput();
        std::cout << "Throughput" << std::endl;
        std::cout << "  Read Capacity : " << ptd.GetReadCapacityUnits() <<
std::endl;
        std::cout << "  Write Capacity: " << ptd.GetWriteCapacityUnits() <<
std::endl;

        const Aws::Vector<Aws::DynamoDB::Model::AttributeDefinition> &ad =
td.GetAttributeDefinitions();
        std::cout << "Attributes" << std::endl;
        for (const auto &a: ad)
            std::cout << "  " << a.GetAttributeName() << " (" <<

Aws::DynamoDB::Model::ScalarAttributeTypeMapper::GetNameForScalarAttributeType(
            a.GetAttributeType()) <<
            ")" << std::endl;
    }
    else {
        std::cerr << "Failed to describe table: " <<
outcome.GetError().GetMessage();
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DescribeTable](#)을 참조하세요.

## ExecuteStatement

다음 코드 예시는 ExecuteStatement의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

INSERT 문을 사용하여 항목을 추가합니다.

```

Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

// 2. Add a new movie using an "Insert" statement. (ExecuteStatement)
Aws::String title;
float rating;
int year;
Aws::String plot;
{
    title = askQuestion(
        "Enter the title of a movie you want to add to the table: ");
    year = askQuestionForInt("What year was it released? ");
    rating = askQuestionForFloatRange("On a scale of 1 - 10, how do you rate it?",
        1, 10);
    plot = askQuestion("Summarize the plot for me: ");

    Aws::DynamoDB::Model::ExecuteStatementRequest request;
    std::stringstream sqlStream;
    sqlStream << "INSERT INTO \" << MOVIE_TABLE_NAME << "\" VALUE {\""
        << TITLE_KEY << "': ?, '\" << YEAR_KEY << "': ?, '\"
        << INFO_KEY << "': ?}";

    request.SetStatement(sqlStream.str());

    // Create the parameter attributes.
    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));

    Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

    std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
    Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
        ALLOCATION_TAG.c_str());
    ratingAttribute->SetN(rating);
    infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

    std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
    Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
        ALLOCATION_TAG.c_str());
    plotAttribute->SetS(plot);
    infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
    attributes.push_back(infoMapAttribute);
}

```

```

        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
            request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to add a movie: " <<
outcome.GetError().GetMessage()
                << std::endl;
            return false;
        }
    }
}

```

SELECT 문을 사용하여 항목을 가져옵니다.

```

// 3. Get the data for the movie using a "Select" statement. (ExecuteStatement)
{
    Aws::DynamoDB::Model::ExecuteStatementRequest request;
    std::stringstream sqlStream;
    sqlStream << "SELECT * FROM \"" << MOVIE_TABLE_NAME << "\" WHERE "
        << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

    request.SetStatement(sqlStream.str());

    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
    request.SetParameters(attributes);

    Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
    request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to retrieve movie information: "
            << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
    else {
        // Print the retrieved movie information.
    }
}

```

```

        const Aws::DynamoDB::Model::ExecuteStatementResult &result =
outcome.GetResult();

        const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetItems();

        if (items.size() == 1) {
            printMovieInfo(items[0]);
        }
        else {
            std::cerr << "Error: " << items.size() << " movies were retrieved. "
<< " There should be only one movie." << std::endl;
        }
    }
}

```

UPDATE 문을 사용하여 항목을 업데이트합니다.

```

// 4. Update the data for the movie using an "Update" statement.
(ExecuteStatement)
{
    rating = askQuestionForFloatRange(
        Aws::String("\nLet's update your movie.\nYou rated it  ") +
        std::to_string(rating)
        + ", what new rating would you give it? ", 1, 10);

    Aws::DynamoDB::Model::ExecuteStatementRequest request;
    std::stringstream sqlStream;
    sqlStream << "UPDATE \"" << MOVIE_TABLE_NAME << "\" SET "
        << INFO_KEY << "." << RATING_KEY << "=? WHERE "
        << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";

    request.SetStatement(sqlStream.str());

    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(rating));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));

    request.SetParameters(attributes);
}

```

```

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
    request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to update a movie: "
            << outcome.GetError().GetMessage();
        return false;
    }
}

```

DELETE 문을 사용하여 항목을 삭제합니다.

```

// 6. Delete the movie using a "Delete" statement. (ExecuteStatement)
{
    Aws::DynamoDB::Model::ExecuteStatementRequest request;
    std::stringstream sqlStream;
    sqlStream << "DELETE FROM \" << MOVIE_TABLE_NAME << "\" WHERE \"
        << TITLE_KEY << "\"=? and \" << YEAR_KEY << "\"=?\";

    request.SetStatement(sqlStream.str());

    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
    request.SetParameters(attributes);

    Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
    request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to delete the movie: "
            << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ExecuteStatement](#)를 참조하세요.

## GetItem

다음 코드 예시는 GetItem의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

///
//! Get an item from an Amazon DynamoDB table.
/*!
    \sa getItem()
    \param tableName: The table name.
    \param partitionKey: The partition key.
    \param partitionValue: The value for the partition key.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/

bool AwsDoc::DynamoDB::getItem(const Aws::String &tableName,
                               const Aws::String &partitionKey,
                               const Aws::String &partitionValue,
                               const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
    Aws::DynamoDB::Model::GetItemRequest request;

    // Set up the request.
    request.SetTableName(tableName);
    request.AddKey(partitionKey,
                   Aws::DynamoDB::Model::AttributeValue().SetS(partitionValue));

    // Retrieve the item's fields and values.
    const Aws::DynamoDB::Model::GetItemOutcome &outcome =
dynamoClient.GetItem(request);
    if (outcome.IsSuccess()) {
        // Reference the retrieved fields/values.
        const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> &item =
outcome.GetResult().GetItem();
        if (!item.empty()) {


```

```

        // Output each retrieved field and its value.
        for (const auto &i: item)
            std::cout << "Values: " << i.first << ": " << i.second.GetS()
                << std::endl;
    }
    else {
        std::cout << "No item found with the key " << partitionKey << std::endl;
    }
}
else {
    std::cerr << "Failed to get item: " << outcome.GetError().GetMessage();
}

return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetItem](#)을 참조하세요.

## ListTables

다음 코드 예시는 ListTables의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

/*! List the Amazon DynamoDB tables for the current AWS account.
 *!
 * \sa listTables()
 * \param clientConfiguration: AWS client configuration.
 * \return bool: Function succeeded.
 */

bool AwsDoc::DynamoDB::listTables(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
}

```

```

    Aws::DynamoDB::Model::ListTablesRequest listTablesRequest;
    listTablesRequest.SetLimit(50);
    do {
        const Aws::DynamoDB::Model::ListTablesOutcome &outcome =
dynamoClient.ListTables(
            listTablesRequest);
        if (!outcome.IsSuccess()) {
            std::cout << "Error: " << outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        for (const auto &tableName: outcome.GetResult().GetTableNames())
            std::cout << tableName << std::endl;
        listTablesRequest.SetExclusiveStartTableName(
            outcome.GetResult().GetLastEvaluatedTableName());

    } while (!listTablesRequest.GetExclusiveStartTableName().empty());

    return true;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ListTables](#)를 참조하세요.

## PutItem

다음 코드 예시는 PutItem의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Put an item in an Amazon DynamoDB table.
/*!
    \sa putItem()
    \param tableName: The table name.
    \param artistKey: The artist key. This is the partition key for the table.
    \param artistValue: The artist value.

```

```

\param albumTitleKey: The album title key.
\param albumTitleValue: The album title value.
\param awardsKey: The awards key.
\param awardsValue: The awards value.
\param songTitleKey: The song title key.
\param songTitleValue: The song title value.
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::putItem(const Aws::String &tableName,
                                const Aws::String &artistKey,
                                const Aws::String &artistValue,
                                const Aws::String &albumTitleKey,
                                const Aws::String &albumTitleValue,
                                const Aws::String &awardsKey,
                                const Aws::String &awardsValue,
                                const Aws::String &songTitleKey,
                                const Aws::String &songTitleValue,
                                const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::PutItemRequest putItemRequest;
    putItemRequest.SetTableName(tableName);

    putItemRequest.AddItem(artistKey, Aws::DynamoDB::Model::AttributeValue().SetS(
        artistValue)); // This is the hash key.
    putItemRequest.AddItem(albumTitleKey,
        Aws::DynamoDB::Model::AttributeValue().SetS(
            albumTitleValue));
    putItemRequest.AddItem(awardsKey,

        Aws::DynamoDB::Model::AttributeValue().SetS(awardsValue));
    putItemRequest.AddItem(songTitleKey,

        Aws::DynamoDB::Model::AttributeValue().SetS(songTitleValue));

    const Aws::DynamoDB::Model::PutItemOutcome outcome = dynamoClient.PutItem(
        putItemRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully added Item!" << std::endl;
    }
    else {
        std::cerr << outcome.GetError().GetMessage() << std::endl;
    }
}

```

```

        return false;
    }

    return waitTableActive(tableName, dynamoClient);
}

```

테이블이 활성화될 때까지 대기하는 코드입니다.

```

//! Query a newly created DynamoDB table until it is active.
/*!
    \sa waitTableActive()
    \param waitTableActive: The DynamoDB table's name.
    \param dynamoClient: A DynamoDB client.
    \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                         const Aws::DynamoDB::DynamoDBClient
                                         &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
dynamoClient.DescribeTable(
            request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            }
    }
    else {

```

```

        std::cerr << "Error DynamoDB::waitTableActive "
                    << result.GetError().GetMessage() << std::endl;
        return false;
    }
    count++;
}
return false;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [PutItem](#)을 참조하세요.

## Query

다음 코드 예시는 Query의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Perform a query on an Amazon DynamoDB Table and retrieve items.
/*!
    \sa queryItem()
    \param tableName: The table name.
    \param partitionKey: The partition key.
    \param partitionValue: The value for the partition key.
    \param projectionExpression: The projections expression, which is ignored if
empty.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/

/*
 * The partition key attribute is searched with the specified value. By default, all
fields and values
 * contained in the item are returned. If an optional projection expression is
 * specified on the command line, only the specified fields and values are
 * returned.

```

```

*/

bool AwsDoc::DynamoDB::queryItems(const Aws::String &tableName,
                                   const Aws::String &partitionKey,
                                   const Aws::String &partitionValue,
                                   const Aws::String &projectionExpression,
                                   const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
    Aws::DynamoDB::Model::QueryRequest request;

    request.SetTableName(tableName);

    if (!projectionExpression.empty()) {
        request.SetProjectionExpression(projectionExpression);
    }

    // Set query key condition expression.
    request.SetKeyConditionExpression(partitionKey + "= :valueToMatch");

    // Set Expression AttributeValues.
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> attributeValues;
    attributeValues.emplace(":valueToMatch", partitionValue);

    request.SetExpressionAttributeValues(attributeValues);

    bool result = true;

    // "exclusiveStartKey" is used for pagination.
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue> exclusiveStartKey;
    do {
        if (!exclusiveStartKey.empty()) {
            request.SetExclusiveStartKey(exclusiveStartKey);
            exclusiveStartKey.clear();
        }
        // Perform Query operation.
        const Aws::DynamoDB::Model::QueryOutcome &outcome =
dynamoClient.Query(request);
        if (outcome.IsSuccess()) {
            // Reference the retrieved items.
            const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = outcome.GetResult().GetItems();
            if (!items.empty()) {

```

```

        std::cout << "Number of items retrieved from Query: " <<
items.size()
                << std::endl;
        // Iterate each item and print.
        for (const auto &item: items) {
            std::cout
                <<
"*****"
                << std::endl;
            // Output each retrieved field and its value.
            for (const auto &i: item)
                std::cout << i.first << ": " << i.second.GetS() <<
std::endl;
        }
    }
    else {
        std::cout << "No item found in table: " << tableName << std::endl;
    }

    exclusiveStartKey = outcome.GetResult().GetLastEvaluatedKey();
}
else {
    std::cerr << "Failed to Query items: " <<
outcome.GetError().GetMessage();
    result = false;
    break;
}
} while (!exclusiveStartKey.empty());

return result;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [Query](#)를 참조하세요.

## Scan

다음 코드 예시는 Scan의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

///! Scan an Amazon DynamoDB table.
/*!
    \sa scanTable()
    \param tableName: Name for the DynamoDB table.
    \param projectionExpression: An optional projection expression, ignored if empty.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/

bool AwsDoc::DynamoDB::scanTable(const Aws::String &tableName,
                                const Aws::String &projectionExpression,
                                const Aws::Client::ClientConfiguration
                                &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);
    Aws::DynamoDB::Model::ScanRequest request;
    request.SetTableName(tableName);

    if (!projectionExpression.empty())
        request.SetProjectionExpression(projectionExpression);

    Aws::Vector<Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>>
all_items;
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
last_evaluated_key; // Used for pagination;
    do {
        if (!last_evaluated_key.empty()) {
            request.SetExclusiveStartKey(last_evaluated_key);
        }
        const Aws::DynamoDB::Model::ScanOutcome &outcome =
dynamoClient.Scan(request);
        if (outcome.IsSuccess()) {
            // Reference the retrieved items.
            const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = outcome.GetResult().GetItems();

```

```

        all_items.insert(all_items.end(), items.begin(), items.end());

        last_evaluated_key = outcome.GetResult().GetLastEvaluatedKey();
    }
    else {
        std::cerr << "Failed to Scan items: " << outcome.GetError().GetMessage()
                    << std::endl;
        return false;
    }

} while (!last_evaluated_key.empty());

if (!all_items.empty()) {
    std::cout << "Number of items retrieved from scan: " << all_items.size()
                << std::endl;
    // Iterate each item and print.
    for (const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
&itemMap: all_items) {
        std::cout << "*****"
                    << std::endl;
        // Output each retrieved field and its value.
        for (const auto &itemEntry: itemMap)
            std::cout << itemEntry.first << ": " << itemEntry.second.GetS()
                        << std::endl;
    }
}

else {
    std::cout << "No items found in table: " << tableName << std::endl;
}

return true;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [Scan](#)을 참조하세요.

## UpdateItem

다음 코드 예시는 UpdateItem의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

///
//! Update an Amazon DynamoDB table item.
/*!
    \sa updateItem()
    \param tableName: The table name.
    \param partitionKey: The partition key.
    \param partitionValue: The value for the partition key.
    \param attributeKey: The key for the attribute to be updated.
    \param attributeValue: The value for the attribute to be updated.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/

/*
 * The example code only sets/updates an attribute value. It processes
 * the attribute value as a string, even if the value could be interpreted
 * as a number. Also, the example code does not remove an existing attribute
 * from the key value.
 */

bool AwsDoc::DynamoDB::updateItem(const Aws::String &tableName,
                                   const Aws::String &partitionKey,
                                   const Aws::String &partitionValue,
                                   const Aws::String &attributeKey,
                                   const Aws::String &attributeValue,
                                   const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    // *** Define UpdateItem request arguments.
    // Define TableName argument.
    Aws::DynamoDB::Model::UpdateItemRequest request;
    request.SetTableName(tableName);

    // Define KeyName argument.


```

```

    Aws::DynamoDB::Model::AttributeValue attribValue;
    attribValue.SetS(partitionValue);
    request.AddKey(partitionKey, attribValue);

    // Construct the SET update expression argument.
    Aws::String update_expression("SET #a = :valueA");
    request.SetUpdateExpression(update_expression);

    // Construct attribute name argument.
    Aws::Map<Aws::String, Aws::String> expressionAttributeNames;
    expressionAttributeNames["#a"] = attributeKey;
    request.SetExpressionAttributeNames(expressionAttributeNames);

    // Construct attribute value argument.
    Aws::DynamoDB::Model::AttributeValue attributeUpdatedValue;
    attributeUpdatedValue.SetS(attributeValue);
    Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
expressionAttributeValues;
    expressionAttributeValues[":valueA"] = attributeUpdatedValue;
    request.SetExpressionAttributeValues(expressionAttributeValues);

    // Update the item.
    const Aws::DynamoDB::Model::UpdateItemOutcome &outcome =
dynamoClient.UpdateItem(
        request);
    if (outcome.IsSuccess()) {
        std::cout << "Item was updated" << std::endl;
    } else {
        std::cerr << outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    return waitTableActive(tableName, dynamoClient);
}

```

테이블이 활성화될 때까지 대기하는 코드입니다.

```

//! Query a newly created DynamoDB table until it is active.
/*!
    \sa waitTableActive()
    \param waitTableActive: The DynamoDB table's name.
    \param dynamoClient: A DynamoDB client.

```

```

    \return bool: Function succeeded.
    */
    bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                           const Aws::DynamoDB::DynamoDBClient
                                           &dynamoClient) {

        // Repeatedly call DescribeTable until table is ACTIVE.
        const int MAX_QUERIES = 20;
        Aws::DynamoDB::Model::DescribeTableRequest request;
        request.SetTableName(tableName);

        int count = 0;
        while (count < MAX_QUERIES) {
            const Aws::DynamoDB::Model::DescribeTableOutcome &result =
dynamoClient.DescribeTable(
                request);
            if (result.IsSuccess()) {
                Aws::DynamoDB::Model::TableStatus status =
result.GetResult().GetTable().GetTableStatus();

                if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                    std::this_thread::sleep_for(std::chrono::seconds(1));
                }
                else {
                    return true;
                }
            }
            else {
                std::cerr << "Error DynamoDB::waitTableActive "
                    << result.GetError().GetMessage() << std::endl;
                return false;
            }
            count++;
        }
        return false;
    }
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [UpdateItem](#)을 참조하세요.

## UpdateTable

다음 코드 예시는 UpdateTable의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Update a DynamoDB table.
/*!
    \sa updateTable()
    \param tableName: Name for the DynamoDB table.
    \param readCapacity: Provisioned read capacity.
    \param writeCapacity: Provisioned write capacity.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::updateTable(const Aws::String &tableName,
                                    long long readCapacity, long long writeCapacity,
                                    const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    std::cout << "Updating " << tableName << " with new provisioned throughput
values"
                << std::endl;
    std::cout << "Read capacity : " << readCapacity << std::endl;
    std::cout << "Write capacity: " << writeCapacity << std::endl;

    Aws::DynamoDB::Model::UpdateTableRequest request;
    Aws::DynamoDB::Model::ProvisionedThroughput provisionedThroughput;

    provisionedThroughput.WithReadCapacityUnits(readCapacity).WithWriteCapacityUnits(
        writeCapacity);

    request.WithProvisionedThroughput(provisionedThroughput).WithTableName(tableName);

    const Aws::DynamoDB::Model::UpdateTableOutcome &outcome =
    dynamoClient.UpdateTable(
        request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated the table." << std::endl;
    }
}

```

```

    } else {
        const Aws::DynamoDB::DynamoDBError &error = outcome.GetError();
        if (error.GetErrorType() == Aws::DynamoDB::DynamoDBErrors::VALIDATION &&
            error.GetMessage().find("The provisioned throughput for the table will
not change") != std::string::npos) {
            std::cout << "The provisioned throughput for the table will not change."
<< std::endl;
        } else {
            std::cerr << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }

    return waitTableActive(tableName, dynamoClient);
}

```

테이블이 활성화될 때까지 대기하는 코드입니다.

```

//! Query a newly created DynamoDB table until it is active.
/*!
    \sa waitTableActive()
    \param waitTableActive: The DynamoDB table's name.
    \param dynamoClient: A DynamoDB client.
    \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                        const Aws::DynamoDB::DynamoDBClient
&dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
dynamoClient.DescribeTable(
            request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
result.GetResult().GetTable().GetTableStatus();

```

```

        if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
            std::this_thread::sleep_for(std::chrono::seconds(1));
        }
        else {
            return true;
        }
    }
    else {
        std::cerr << "Error DynamoDB::waitTableActive "
                    << result.GetError().GetMessage() << std::endl;
        return false;
    }
    count++;
}
return false;
}

```

- API에 대한 세부 정보는 AWS SDK for C++ API 참조의 [UpdateTable](#)을 참조하세요.

## 시나리오

### 사진을 관리하기 위한 서버리스 애플리케이션 만들기

다음 코드 예시에서는 사용자가 레이블을 사용하여 사진을 관리할 수 있는 서버리스 애플리케이션을 생성하는 방법을 보여줍니다.

#### SDK for C++

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 [AWS 커뮤니티](#)의 게시물을 참조하세요.

이 예제에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda

- Amazon Rekognition
- Amazon S3
- Amazon SNS

## 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 C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// 1. Create a table. (CreateTable)
if (AwsDoc::DynamoDB::createMoviesDynamoDBTable(clientConfig)) {

    AwsDoc::DynamoDB::partiqlBatchExecuteScenario(clientConfig);

    // 7. Delete the table. (DeleteTable)
    AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(clientConfig);
}

//! Scenario to modify and query a DynamoDB table using PartiQL batch statements.
/*!
    \sa partiqlBatchExecuteScenario()
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::partiqlBatchExecuteScenario(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
```

```

// 2. Add multiple movies using "Insert" statements. (BatchExecuteStatement)
Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

std::vector<Aws::String> titles;
std::vector<float> ratings;
std::vector<int> years;
std::vector<Aws::String> plots;
Aws::String doAgain = "n";
do {
    Aws::String aTitle = askQuestion(
        "Enter the title of a movie you want to add to the table: ");
    titles.push_back(aTitle);
    int aYear = askQuestionForInt("What year was it released? ");
    years.push_back(aYear);
    float aRating = askQuestionForFloatRange(
        "On a scale of 1 - 10, how do you rate it? ",
        1, 10);
    ratings.push_back(aRating);
    Aws::String aPlot = askQuestion("Summarize the plot for me: ");
    plots.push_back(aPlot);

    doAgain = askQuestion(Aws::String("Would you like to add more movies? (y/n)
"));
} while (doAgain == "y");

std::cout << "Adding " << titles.size()
    << (titles.size() == 1 ? " movie " : " movies ")
    << "to the table using a batch \"INSERT\" statement." << std::endl;

{
    Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
        titles.size());

    std::stringstream sqlStream;
    sqlStream << "INSERT INTO \"" << MOVIE_TABLE_NAME << "\" VALUE {'"
        << TITLE_KEY << "': ?, '" << YEAR_KEY << "': ?, '"
        << INFO_KEY << "': ?}";

    std::string sql(sqlStream.str());

    for (size_t i = 0; i < statements.size(); ++i) {
        statements[i].SetStatement(sql);
    }
}

```

```

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));

        // Create attribute for the info map.
        Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
        Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
            ALLOCATION_TAG.c_str());
        ratingAttribute->SetN(ratings[i]);
        infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
        Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
            ALLOCATION_TAG.c_str());
        plotAttribute->SetS(plots[i]);
        infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
        attributes.push_back(infoMapAttribute);
        statements[i].SetParameters(attributes);
    }

    Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

    request.SetStatements(statements);

    Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
    dynamoClient.BatchExecuteStatement(
        request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to add the movies: " <<
        outcome.GetError().GetMessage()
            << std::endl;
        return false;
    }
}

std::cout << "Retrieving the movie data with a batch \"SELECT\" statement."
    << std::endl;

// 3. Get the data for multiple movies using "Select" statements.
(BatchExecuteStatement)

```

```

{
    Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
        titles.size());
    std::stringstream sqlStream;
    sqlStream << "SELECT * FROM \"" << MOVIE_TABLE_NAME << "\" WHERE "
        << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

    std::string sql(sqlStream.str());

    for (size_t i = 0; i < statements.size(); ++i) {
        statements[i].SetStatement(sql);
        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
        statements[i].SetParameters(attributes);
    }

    Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

    request.SetStatements(statements);

    Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
    dynamoClient.BatchExecuteStatement(
        request);
    if (outcome.IsSuccess()) {
        const Aws::DynamoDB::Model::BatchExecuteStatementResult &result =
        outcome.GetResult();

        const Aws::Vector<Aws::DynamoDB::Model::BatchStatementResponse>
        &responses = result.GetResponses();

        for (const Aws::DynamoDB::Model::BatchStatementResponse &response:
        responses) {
            const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
            &item = response.GetItem();

            printMovieInfo(item);
        }
    }
    else {
        std::cerr << "Failed to retrieve the movie information: "
            << outcome.GetError().GetMessage() << std::endl;
    }
}

```

```

        return false;
    }
}

// 4. Update the data for multiple movies using "Update" statements.
(BatchExecuteStatement)

for (size_t i = 0; i < titles.size(); ++i) {
    ratings[i] = askQuestionForFloatRange(
        Aws::String("\nLet's update your the movie, \"") + titles[i] +
        ".\nYou rated it " + std::to_string(ratings[i])
        + ", what new rating would you give it? ", 1, 10);
}

std::cout << "Updating the movie with a batch \"UPDATE\" statement." <<
std::endl;

{
    Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
        titles.size());

    std::stringstream sqlStream;
    sqlStream << "UPDATE \"" << MOVIE_TABLE_NAME << "\" SET "
        << INFO_KEY << "." << RATING_KEY << "=? WHERE "
        << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";

    std::string sql(sqlStream.str());

    for (size_t i = 0; i < statements.size(); ++i) {
        statements[i].SetStatement(sql);

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetN(ratings[i]));
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));

        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
        statements[i].SetParameters(attributes);
    }

    Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

```

```

        request.SetStatements(statements);
        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
            request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to update movie information: "
                << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }

    std::cout << "Retrieving the updated movie data with a batch \"SELECT\"
statement."
        << std::endl;

    // 5. Get the updated data for multiple movies using "Select" statements.
    (BatchExecuteStatement)
    {
        Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
            titles.size());
        std::stringstream sqlStream;
        sqlStream << "SELECT * FROM \"" << MOVIE_TABLE_NAME << "\" WHERE "
            << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

        std::string sql(sqlStream.str());

        for (size_t i = 0; i < statements.size(); ++i) {
            statements[i].SetStatement(sql);
            Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
            attributes.push_back(
                Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));
            attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
            statements[i].SetParameters(attributes);
        }

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);

        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
            request);
        if (outcome.IsSuccess()) {

```

```

        const Aws::DynamoDB::Model::BatchExecuteStatementResult &result =
outcome.GetResult();

        const Aws::Vector<Aws::DynamoDB::Model::BatchStatementResponse>
&responses = result.GetResponses();

        for (const Aws::DynamoDB::Model::BatchStatementResponse &response:
responses) {
            const Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
&item = response.GetItem();

            printMovieInfo(item);
        }
    }
    else {
        std::cerr << "Failed to retrieve the movies information: "
            << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
}

std::cout << "Deleting the movie data with a batch \"DELETE\" statement."
    << std::endl;

// 6. Delete multiple movies using "Delete" statements. (BatchExecuteStatement)
{
    Aws::Vector<Aws::DynamoDB::Model::BatchStatementRequest> statements(
        titles.size());
    std::stringstream sqlStream;
    sqlStream << "DELETE FROM \"" << MOVIE_TABLE_NAME << "\" WHERE "
        << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

    std::string sql(sqlStream.str());

    for (size_t i = 0; i < statements.size(); ++i) {
        statements[i].SetStatement(sql);
        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(
            Aws::DynamoDB::Model::AttributeValue().SetS(titles[i]));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(years[i]));
        statements[i].SetParameters(attributes);
    }
}

```

```

        Aws::DynamoDB::Model::BatchExecuteStatementRequest request;

        request.SetStatements(statements);

        Aws::DynamoDB::Model::BatchExecuteStatementOutcome outcome =
dynamoClient.BatchExecuteStatement(
            request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to delete the movies: "
                << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    }

    return true;
}

//! Create a DynamoDB table to be used in sample code scenarios.
/*!
    \sa createMoviesDynamoDBTable()
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::createMoviesDynamoDBTable(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    bool movieTableAlreadyExisted = false;

    {
        Aws::DynamoDB::Model::CreateTableRequest request;

        Aws::DynamoDB::Model::AttributeDefinition yearAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(YEAR_KEY);
        yearAttributeDefinition.SetAttributeType(
            Aws::DynamoDB::Model::ScalarAttributeType::N);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::AttributeDefinition titleAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(TITLE_KEY);
        yearAttributeDefinition.SetAttributeType(
            Aws::DynamoDB::Model::ScalarAttributeType::S);
        request.AddAttributeDefinitions(yearAttributeDefinition);
    }
}

```

```

    Aws::DynamoDB::Model::KeySchemaElement yearKeySchema;
    yearKeySchema.WithAttributeName(YEAR_KEY).WithKeyType(
        Aws::DynamoDB::Model::KeyType::HASH);
    request.AddKeySchema(yearKeySchema);

    Aws::DynamoDB::Model::KeySchemaElement titleKeySchema;
    yearKeySchema.WithAttributeName(TITLE_KEY).WithKeyType(
        Aws::DynamoDB::Model::KeyType::RANGE);
    request.AddKeySchema(yearKeySchema);

    Aws::DynamoDB::Model::ProvisionedThroughput throughput;
    throughput.WithReadCapacityUnits(
        PROVISIONED_THROUGHPUT_UNITS).WithWriteCapacityUnits(
        PROVISIONED_THROUGHPUT_UNITS);
    request.SetProvisionedThroughput(throughput);
    request.SetTableName(MOVIE_TABLE_NAME);

    std::cout << "Creating table '" << MOVIE_TABLE_NAME << "'..." << std::endl;
    const Aws::DynamoDB::Model::CreateTableOutcome &result =
    dynamoClient.CreateTable(
        request);
    if (!result.IsSuccess()) {
        if (result.GetError().GetErrorType() ==
            Aws::DynamoDB::DynamoDBErrors::RESOURCE_IN_USE) {
            std::cout << "Table already exists." << std::endl;
            movieTableAlreadyExisted = true;
        }
        else {
            std::cerr << "Failed to create table: "
                << result.GetError().GetMessage();
            return false;
        }
    }
}

// Wait for table to become active.
if (!movieTableAlreadyExisted) {
    std::cout << "Waiting for table '" << MOVIE_TABLE_NAME
        << "' to become active...." << std::endl;
    if (!AwsDoc::DynamoDB::waitTableActive(MOVIE_TABLE_NAME,
        clientConfiguration)) {
        return false;
    }
}

```

```

        std::cout << "Table '" << MOVIE_TABLE_NAME << "' created and active."
            << std::endl;
    }

    return true;
}

//! Delete the DynamoDB table used for sample code scenarios.
/*!
    \sa deleteMoviesDynamoDBTable()
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(MOVIE_TABLE_NAME);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result =
dynamoClient.DeleteTable(
    request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
            << result.GetResult().GetTableDescription().GetTableName()
            << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
            << std::endl;
    }

    return result.IsSuccess();
}

//! Query a newly created DynamoDB table until it is active.
/*!
    \sa waitTableActive()
    \param waitTableActive: The DynamoDB table's name.
    \param dynamoClient: A DynamoDB client.
    \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
```

```

const Aws::DynamoDB::DynamoDBClient
&dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
dynamoClient.DescribeTable(
            request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [BatchExecuteStatement](#)를 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 C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// 1. Create a table. (CreateTable)
if (AwsDoc::DynamoDB::createMoviesDynamoDBTable(clientConfig)) {

    AwsDoc::DynamoDB::partiqlExecuteScenario(clientConfig);

    // 7. Delete the table. (DeleteTable)
    AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(clientConfig);
}

//! Scenario to modify and query a DynamoDB table using single PartiQL statements.
/*!
    \sa partiqlExecuteScenario()
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool
AwsDoc::DynamoDB::partiqlExecuteScenario(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    // 2. Add a new movie using an "Insert" statement. (ExecuteStatement)
    Aws::String title;
    float rating;
    int year;
    Aws::String plot;
    {
        title = askQuestion(
            "Enter the title of a movie you want to add to the table: ");
        year = askQuestionForInt("What year was it released? ");
    }
}
```

```

        rating = askQuestionForFloatRange("On a scale of 1 - 10, how do you rate it?",
                                          1, 10);
        plot = askQuestion("Summarize the plot for me: ");

        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "INSERT INTO \"" << MOVIE_TABLE_NAME << "\" VALUE {'"
                  << TITLE_KEY << "': ?, '" << YEAR_KEY << "': ?, '"
                  << INFO_KEY << "': ?}";

        request.SetStatement(sqlStream.str());

        // Create the parameter attributes.
        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));

        Aws::DynamoDB::Model::AttributeValue infoMapAttribute;

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> ratingAttribute =
        Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
            ALLOCATION_TAG.c_str());
        ratingAttribute->SetN(rating);
        infoMapAttribute.AddMEntry(RATING_KEY, ratingAttribute);

        std::shared_ptr<Aws::DynamoDB::Model::AttributeValue> plotAttribute =
        Aws::MakeShared<Aws::DynamoDB::Model::AttributeValue>(
            ALLOCATION_TAG.c_str());
        plotAttribute->SetS(plot);
        infoMapAttribute.AddMEntry(PLOT_KEY, plotAttribute);
        attributes.push_back(infoMapAttribute);
        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
        dynamoClient.ExecuteStatement(
            request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to add a movie: " <<
            outcome.GetError().GetMessage()
                << std::endl;
            return false;
        }
    }

```

```

}

std::cout << "\nAdded '" << title << "' to '" << MOVIE_TABLE_NAME << "'."
    << std::endl;

// 3. Get the data for the movie using a "Select" statement. (ExecuteStatement)
{
    Aws::DynamoDB::Model::ExecuteStatementRequest request;
    std::stringstream sqlStream;
    sqlStream << "SELECT * FROM \"" << MOVIE_TABLE_NAME << "\" WHERE "
        << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

    request.SetStatement(sqlStream.str());

    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
    request.SetParameters(attributes);

    Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
    request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to retrieve movie information: "
            << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
    else {
        // Print the retrieved movie information.
        const Aws::DynamoDB::Model::ExecuteStatementResult &result =
outcome.GetResult();

        const Aws::Vector<Aws::Map<Aws::String,
Aws::DynamoDB::Model::AttributeValue>> &items = result.GetItems();

        if (items.size() == 1) {
            printMovieInfo(items[0]);
        }
        else {
            std::cerr << "Error: " << items.size() << " movies were retrieved. "
                << " There should be only one movie." << std::endl;
        }
    }
}

```

```

    }

    // 4. Update the data for the movie using an "Update" statement.
    (ExecuteStatement)
    {
        rating = askQuestionForFloatRange(
            Aws::String("\nLet's update your movie.\nYou rated it  ") +
            std::to_string(rating)
            + ", what new rating would you give it? ", 1, 10);

        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "UPDATE \"" << MOVIE_TABLE_NAME << "\" SET "
            << INFO_KEY << "." << RATING_KEY << "=? WHERE "
            << TITLE_KEY << "=? AND " << YEAR_KEY << "=?";

        request.SetStatement(sqlStream.str());

        Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(rating));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
        attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));

        request.SetParameters(attributes);

        Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
        dynamoClient.ExecuteStatement(
            request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to update a movie: "
                << outcome.GetError().GetMessage();
            return false;
        }
    }

    std::cout << "\nUpdated '" << title << "' with new attributes:" << std::endl;

    // 5. Get the updated data for the movie using a "Select" statement.
    (ExecuteStatement)
    {
        Aws::DynamoDB::Model::ExecuteStatementRequest request;
        std::stringstream sqlStream;
        sqlStream << "SELECT * FROM \"" << MOVIE_TABLE_NAME << "\" WHERE "

```

```

        << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

    request.SetStatement(sqlStream.str());

    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
    request.SetParameters(attributes);

    Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
    dynamoClient.ExecuteStatement(
        request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to retrieve the movie information: "
            << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
    else {
        const Aws::DynamoDB::Model::ExecuteStatementResult &result =
        outcome.GetResult();

        const Aws::Vector<Aws::Map<Aws::String,
        Aws::DynamoDB::Model::AttributeValue>> &items = result.GetItems();

        if (items.size() == 1) {
            printMovieInfo(items[0]);
        }
        else {
            std::cerr << "Error: " << items.size() << " movies were retrieved. "
                << " There should be only one movie." << std::endl;
        }
    }
}

std::cout << "Deleting the movie" << std::endl;

// 6. Delete the movie using a "Delete" statement. (ExecuteStatement)
{
    Aws::DynamoDB::Model::ExecuteStatementRequest request;
    std::stringstream sqlStream;
    sqlStream << "DELETE FROM \"" << MOVIE_TABLE_NAME << "\" WHERE "
        << TITLE_KEY << "=? and " << YEAR_KEY << "=?";

    request.SetStatement(sqlStream.str());

```

```

    Aws::Vector<Aws::DynamoDB::Model::AttributeValue> attributes;
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetS(title));
    attributes.push_back(Aws::DynamoDB::Model::AttributeValue().SetN(year));
    request.SetParameters(attributes);

    Aws::DynamoDB::Model::ExecuteStatementOutcome outcome =
dynamoClient.ExecuteStatement(
    request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to delete the movie: "
            << outcome.GetError().GetMessage() << std::endl;
        return false;
    }
}

std::cout << "Movie successfully deleted." << std::endl;
return true;
}

//! Create a DynamoDB table to be used in sample code scenarios.
/*!
    \sa createMoviesDynamoDBTable()
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::createMoviesDynamoDBTable(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    bool movieTableAlreadyExisted = false;

    {
        Aws::DynamoDB::Model::CreateTableRequest request;

        Aws::DynamoDB::Model::AttributeDefinition yearAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(YEAR_KEY);
        yearAttributeDefinition.SetAttributeType(
            Aws::DynamoDB::Model::ScalarAttributeType::N);
        request.AddAttributeDefinitions(yearAttributeDefinition);

        Aws::DynamoDB::Model::AttributeDefinition titleAttributeDefinition;
        yearAttributeDefinition.SetAttributeName(TITLE_KEY);
        yearAttributeDefinition.SetAttributeType(

```

```

        Aws::DynamoDB::Model::ScalarAttributeType::S);
    request.AddAttributeDefinitions(yearAttributeDefinition);

    Aws::DynamoDB::Model::KeySchemaElement yearKeySchema;
    yearKeySchema.WithAttributeName(YEAR_KEY).WithKeyType(
        Aws::DynamoDB::Model::KeyType::HASH);
    request.AddKeySchema(yearKeySchema);

    Aws::DynamoDB::Model::KeySchemaElement titleKeySchema;
    yearKeySchema.WithAttributeName(TITLE_KEY).WithKeyType(
        Aws::DynamoDB::Model::KeyType::RANGE);
    request.AddKeySchema(yearKeySchema);

    Aws::DynamoDB::Model::ProvisionedThroughput throughput;
    throughput.WithReadCapacityUnits(
        PROVISIONED_THROUGHPUT_UNITS).WithWriteCapacityUnits(
        PROVISIONED_THROUGHPUT_UNITS);
    request.SetProvisionedThroughput(throughput);
    request.SetTableName(MOVIE_TABLE_NAME);

    std::cout << "Creating table '" << MOVIE_TABLE_NAME << "'..." << std::endl;
    const Aws::DynamoDB::Model::CreateTableOutcome &result =
dynamoClient.CreateTable(
    request);
    if (!result.IsSuccess()) {
        if (result.GetError().GetErrorType() ==
            Aws::DynamoDB::DynamoDBErrors::RESOURCE_IN_USE) {
            std::cout << "Table already exists." << std::endl;
            movieTableAlreadyExisted = true;
        }
        else {
            std::cerr << "Failed to create table: "
                << result.GetError().GetMessage();
            return false;
        }
    }
}

// Wait for table to become active.
if (!movieTableAlreadyExisted) {
    std::cout << "Waiting for table '" << MOVIE_TABLE_NAME
        << "' to become active...." << std::endl;
    if (!AwsDoc::DynamoDB::waitTableActive(MOVIE_TABLE_NAME,
clientConfiguration)) {

```

```

        return false;
    }
    std::cout << "Table '" << MOVIE_TABLE_NAME << "' created and active."
        << std::endl;
}

return true;
}

//! Delete the DynamoDB table used for sample code scenarios.
/*!
    \sa deleteMoviesDynamoDBTable()
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::deleteMoviesDynamoDBTable(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::DeleteTableRequest request;
    request.SetTableName(MOVIE_TABLE_NAME);

    const Aws::DynamoDB::Model::DeleteTableOutcome &result =
    dynamoClient.DeleteTable(
        request);
    if (result.IsSuccess()) {
        std::cout << "Your table \""
            << result.GetResult().GetTableDescription().GetTableName()
            << " was deleted.\n";
    }
    else {
        std::cerr << "Failed to delete table: " << result.GetError().GetMessage()
            << std::endl;
    }

    return result.IsSuccess();
}

//! Query a newly created DynamoDB table until it is active.
/*!
    \sa waitTableActive()
    \param waitTableActive: The DynamoDB table's name.
    \param dynamoClient: A DynamoDB client.
    \return bool: Function succeeded.

```

```

*/
bool AwsDoc::DynamoDB::waitTableActive(const Aws::String &tableName,
                                         const Aws::DynamoDB::DynamoDBClient
                                         &dynamoClient) {

    // Repeatedly call DescribeTable until table is ACTIVE.
    const int MAX_QUERIES = 20;
    Aws::DynamoDB::Model::DescribeTableRequest request;
    request.SetTableName(tableName);

    int count = 0;
    while (count < MAX_QUERIES) {
        const Aws::DynamoDB::Model::DescribeTableOutcome &result =
dynamoClient.DescribeTable(
            request);
        if (result.IsSuccess()) {
            Aws::DynamoDB::Model::TableStatus status =
result.GetResult().GetTable().GetTableStatus();

            if (Aws::DynamoDB::Model::TableStatus::ACTIVE != status) {
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                return true;
            }
        }
        else {
            std::cerr << "Error DynamoDB::waitTableActive "
                << result.GetError().GetMessage() << std::endl;
            return false;
        }
        count++;
    }
    return false;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ExecuteStatement](#)를 참조하세요.

# SDK for C++를 사용한 Amazon EC2 예제

다음 코드 예제에서는 Amazon EC2와 AWS SDK for C++ 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 C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

CMakeLists.txt CMake 파일의 코드입니다.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS ec2)

# Set this project's name.
```

```

project("hello_ec2")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
  for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
      "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
  endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
  # Copy relevant AWS SDK for C++ libraries into the current binary directory for
  running and debugging.

  # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
  may need to uncomment this

  # and set the proper subdirectory to the
  executables' location.

  AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
    ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
  hello_ec2.cpp)

target_link_libraries(${PROJECT_NAME}
  ${AWSSDK_LINK_LIBRARIES})

```

hello\_ec2.cpp 소스 파일의 코드입니다.

```

#include <aws/core/Aws.h>
#include <aws/ec2/EC2Client.h>

```

```

#include <aws/ec2/model/DescribeInstancesRequest.h>
#include <iomanip>
#include <iostream>

/*
 * A "Hello EC2" starter application which initializes an Amazon Elastic Compute
 * Cloud (Amazon EC2) client and describes
 * the Amazon EC2 instances.
 *
 * main function
 *
 * Usage: 'hello_ec2'
 *
 */

int main(int argc, char **argv) {
    (void)argc;
    (void)argv;

    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::EC2::EC2Client ec2Client(clientConfig);
        Aws::EC2::Model::DescribeInstancesRequest request;
        bool header = false;
        bool done = false;
        while (!done) {
            Aws::EC2::Model::DescribeInstancesOutcome outcome =
ec2Client.DescribeInstances(request);
            if (outcome.IsSuccess()) {
                if (!header) {
                    std::cout << std::left <<
                        std::setw(48) << "Name" <<
                        std::setw(20) << "ID" <<
                        std::setw(25) << "Ami" <<
                        std::setw(15) << "Type" <<
                        std::setw(15) << "State" <<

```

```

        std::setw(15) << "Monitoring" << std::endl;
        header = true;
    }

    const std::vector<Aws::EC2::Model::Reservation> &reservations =
        outcome.GetResult().GetReservations();

    for (const auto &reservation: reservations) {
        const std::vector<Aws::EC2::Model::Instance> &instances =
            reservation.GetInstances();
        for (const auto &instance: instances) {
            Aws::String instanceStateString =

Aws::EC2::Model::InstanceStateNameMapper::GetNameForInstanceStateName(
                instance.GetState().GetName());

            Aws::String typeString =

Aws::EC2::Model::InstanceTypeMapper::GetNameForInstanceType(
                instance.GetInstanceType());

            Aws::String monitorString =

Aws::EC2::Model::MonitoringStateMapper::GetNameForMonitoringState(
                instance.GetMonitoring().GetState());
            Aws::String name = "Unknown";

            const std::vector<Aws::EC2::Model::Tag> &tags =
instance.GetTags();
            auto nameIter = std::find_if(tags.cbegin(), tags.cend(),
                [](const Aws::EC2::Model::Tag
&tag) {
                    return tag.GetKey() ==

                    "Name";

                });
            if (nameIter != tags.cend()) {
                name = nameIter->GetValue();
            }
            std::cout <<
                std::setw(48) << name <<
                std::setw(20) << instance.GetInstanceId() <<
                std::setw(25) << instance.GetImageId() <<
                std::setw(15) << typeString <<
                std::setw(15) << instanceStateString <<

```

```

        std::setw(15) << monitorString << std::endl;
    }
}

if (!outcome.GetResult().GetNextToken().empty()) {
    request.SetNextToken(outcome.GetResult().GetNextToken());
} else {
    done = true;
}
} else {
    std::cerr << "Failed to describe EC2 instances:" <<
        outcome.GetError().GetMessage() << std::endl;
    result = 1;
    break;
}
}
}

Aws::ShutdownAPI(options); // Should only be called once.
return result;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DescribeSecurityGroups](#)를 참조하세요.

## 작업

### AllocateAddress

다음 코드 예시는 AllocateAddress의 사용 방법을 보여줍니다.

SDK for C++

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Allocate an Elastic IP address and associate it with an Amazon Elastic Compute
    Cloud
//! (Amazon EC2) instance.
//!
    \param instanceID: An EC2 instance ID.
    \param[out] publicIPAddress: String to return the public IP address.
    \param[out] allocationID: String to return the allocation ID.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */
bool AwsDoc::EC2::allocateAndAssociateAddress(const Aws::String &instanceId,
    Aws::String &publicIPAddress,
                                           Aws::String &allocationID,
                                           const Aws::Client::ClientConfiguration
    &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::AllocateAddressRequest request;
    request.SetDomain(Aws::EC2::Model::DomainType::vpc);

    const Aws::EC2::Model::AllocateAddressOutcome outcome =
        ec2Client.AllocateAddress(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to allocate Elastic IP address:" <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }
    const Aws::EC2::Model::AllocateAddressResponse &response = outcome.GetResult();
    allocationID = response.GetAllocationId();
    publicIPAddress = response.GetPublicIp();

    return true;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [AllocateAddress](#)를 참조하세요.

## AssociateAddress

다음 코드 예시는 AssociateAddress의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    /// Associate an Elastic IP address with an EC2 instance.
    /*!
    \param instanceId: An EC2 instance ID.
    \param allocationId: An Elastic IP allocation ID.
    \param[out] associationID: String to receive the association ID.
    \param clientConfiguration: AWS client configuration.
    \return bool: True if the address was associated with the instance; otherwise,
    false.
    */
    bool AwsDoc::EC2::associateAddress(const Aws::String &instanceId, const Aws::String
    &allocationId,
                                     Aws::String &associationID,
                                     const Aws::Client::ClientConfiguration
    &clientConfiguration) {
        Aws::EC2::EC2Client ec2Client(clientConfiguration);

        Aws::EC2::Model::AssociateAddressRequest request;
        request.SetInstanceId(instanceId);
        request.SetAllocationId(allocationId);

        Aws::EC2::Model::AssociateAddressOutcome outcome =
        ec2Client.AssociateAddress(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to associate address " << allocationId <<
                " with instance " << instanceId << ": " <<
                outcome.GetError().GetMessage() << std::endl;
        } else {
            std::cout << "Successfully associated address " << allocationId <<
                " with instance " << instanceId << std::endl;
            associationID = outcome.GetResult().GetAssociationId();
        }
    }

```

```
    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [AssociateAddress](#)를 참조하세요.

## AuthorizeSecurityGroupIngress

다음 코드 예시는 AuthorizeSecurityGroupIngress의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
//! Authorize ingress to an Amazon Elastic Compute Cloud (Amazon EC2) group.
/*!
    \param groupID: The EC2 group ID.
    \param clientConfiguration: The ClientConfiguration object.
    \return bool: True if the operation was successful, false otherwise.
 */
bool
AwsDoc::EC2::authorizeSecurityGroupIngress(const Aws::String &groupID,
                                           const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::AuthorizeSecurityGroupIngressRequest
authorizeSecurityGroupIngressRequest;
    authorizeSecurityGroupIngressRequest.SetGroupId(groupID);
    buildSampleIngressRule(authorizeSecurityGroupIngressRequest);

    Aws::EC2::Model::AuthorizeSecurityGroupIngressOutcome
authorizeSecurityGroupIngressOutcome =

    ec2Client.AuthorizeSecurityGroupIngress(authorizeSecurityGroupIngressRequest);

    if (authorizeSecurityGroupIngressOutcome.IsSuccess()) {
        std::cout << "Successfully authorized security group ingress." << std::endl;
```

```

    } else {
        std::cerr << "Error authorizing security group ingress: "
                  << authorizeSecurityGroupIngressOutcome.GetError().GetMessage() <<
std::endl;
    }

    return authorizeSecurityGroupIngressOutcome.IsSuccess();
}

```

수신 규칙을 빌드하는 유틸리티 함수입니다.

```

//! Build a sample ingress rule.
/*!
\param authorize_request: An 'AuthorizeSecurityGroupIngressRequest' instance.
\return void:
*/
void buildSampleIngressRule(
    Aws::EC2::Model::AuthorizeSecurityGroupIngressRequest &authorize_request) {
    Aws::String ingressIPRange = "203.0.113.0/24"; // Configure this for your
allowed IP range.
    Aws::EC2::Model::IpRange ip_range;
    ip_range.SetCidrIp(ingressIPRange);

    Aws::EC2::Model::IpPermission permission1;
    permission1.SetIpProtocol("tcp");
    permission1.SetToPort(80);
    permission1.SetFromPort(80);
    permission1.AddIpRanges(ip_range);

    authorize_request.AddIpPermissions(permission1);

    Aws::EC2::Model::IpPermission permission2;
    permission2.SetIpProtocol("tcp");
    permission2.SetToPort(22);
    permission2.SetFromPort(22);
    permission2.AddIpRanges(ip_range);

    authorize_request.AddIpPermissions(permission2);
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [AuthorizeSecurityGroupIngress](#)를 참조하세요.

## CreateKeyPair

다음 코드 예시는 CreateKeyPair의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Create an Amazon Elastic Compute Cloud (Amazon EC2) instance key pair.
/*!
    \param keyPairName: A name for a key pair.
    \param keyFilePath: File path where the credentials are stored. Ignored if it is
    an empty string;
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::EC2::createKeyPair(const Aws::String &keyPairName, const Aws::String
&keyFilePath,
                                const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::CreateKeyPairRequest request;
    request.SetKeyName(keyPairName);

    Aws::EC2::Model::CreateKeyPairOutcome outcome =
    ec2Client.CreateKeyPair(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to create key pair - " << keyPairName << ". " <<
        outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully created key pair named " <<
        keyPairName << std::endl;
        if (!keyFilePath.empty()) {
            std::ofstream keyFile(keyFilePath.c_str());
            keyFile << outcome.GetResult().GetKeyMaterial();
            keyFile.close();
            std::cout << "Keys written to the file " <<
            keyFilePath << std::endl;
        }
    }
}

```

```

    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CreateKeyPair](#)를 참조하세요.

## CreateSecurityGroup

다음 코드 예시는 CreateSecurityGroup의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Create a security group.
/*!
    \param groupName: A security group name.
    \param description: A description.
    \param vpcID: A virtual private cloud (VPC) ID.
    \param[out] groupIDResult: A string to receive the group ID.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::EC2::createSecurityGroup(const Aws::String &groupName,
                                      const Aws::String &description,
                                      const Aws::String &vpcID,
                                      Aws::String &groupIDResult,
                                      const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::CreateSecurityGroupRequest request;

    request.SetGroupName(groupName);

```

```

request.SetDescription(description);
request.SetVpcId(vpcID);

const Aws::EC2::Model::CreateSecurityGroupOutcome outcome =
    ec2Client.CreateSecurityGroup(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Failed to create security group:" <<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}

std::cout << "Successfully created security group named " << groupName <<
    std::endl;

groupIDResult = outcome.GetResult().GetGroupId();

return true;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CreateSecurityGroup](#)을 참조하세요.

## CreateTags

다음 코드 예시는 CreateTags의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Add or overwrite only the specified tags for the specified Amazon Elastic
    Compute Cloud (Amazon EC2) resource or resources.
/*!
    \param resources: The resources for the tags.
    \param tags: Vector of tags.
    \param clientConfiguration: AWS client configuration.

```

```

    \return bool: Function succeeded.
    */
bool AwsDoc::EC2::createTags(const Aws::Vector<Aws::String> &resources,
                             const Aws::Vector<Aws::EC2::Model::Tag> &tags,
                             const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::CreateTagsRequest createTagsRequest;
    createTagsRequest.SetResources(resources);
    createTagsRequest.SetTags(tags);

    Aws::EC2::Model::CreateTagsOutcome outcome =
    ec2Client.CreateTags(createTagsRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully created tags for resources" << std::endl;
    } else {
        std::cerr << "Failed to create tags for resources, " <<
    outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CreateTags](#)를 참조하세요.

## DeleteKeyPair

다음 코드 예시는 DeleteKeyPair의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Delete an Amazon Elastic Compute Cloud (Amazon EC2) instance key pair.
/*!

```

```

\param keyPairName: A name for a key pair.
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/

bool AwsDoc::EC2::deleteKeyPair(const Aws::String &keyPairName,
                                const Aws::Client::ClientConfiguration
                                &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DeleteKeyPairRequest request;

    request.SetKeyName(keyPairName);
    const Aws::EC2::Model::DeleteKeyPairOutcome outcome = ec2Client.DeleteKeyPair(
        request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to delete key pair " << keyPairName <<
            ":" << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully deleted key pair named " << keyPairName <<
            std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteKeyPair](#)를 참조하세요.

## DeleteSecurityGroup

다음 코드 예시는 DeleteSecurityGroup의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Delete a security group.

```

```

/ * !
  \param securityGroupID: A security group ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
  */
bool AwsDoc::EC2::deleteSecurityGroup(const Aws::String &securityGroupID,
                                      const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DeleteSecurityGroupRequest request;

    request.SetGroupId(securityGroupID);
    Aws::EC2::Model::DeleteSecurityGroupOutcome outcome =
    ec2Client.DeleteSecurityGroup(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to delete security group " << securityGroupID <<
            ":" << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully deleted security group " << securityGroupID <<
            std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteSecurityGroup](#)을 참조하세요.

## DescribeAddresses

다음 코드 예시는 DescribeAddresses의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

///! Describe all Elastic IP addresses.

```

```

/ * !
\ param clientConfiguration: AWS client configuration.
\ return bool: Function succeeded.
*/
bool AwsDoc::EC2::describeAddresses(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeAddressesRequest request;
    Aws::EC2::Model::DescribeAddressesOutcome outcome =
    ec2Client.DescribeAddresses(request);
    if (outcome.IsSuccess()) {
        std::cout << std::left << std::setw(20) << "InstanceId" <<
            std::setw(15) << "Public IP" << std::setw(10) << "Domain" <<
            std::setw(30) << "Allocation ID" << std::setw(25) <<
            "NIC ID" << std::endl;

        const Aws::Vector<Aws::EC2::Model::Address> &addresses =
        outcome.GetResult().GetAddresses();
        for (const auto &address: addresses) {
            Aws::String domainString =
                Aws::EC2::Model::DomainTypeMapper::GetNameForDomainType(
                    address.GetDomain());

            std::cout << std::left << std::setw(20) <<
                address.GetInstanceId() << std::setw(15) <<
                address.GetPublicIp() << std::setw(10) << domainString <<
                std::setw(30) << address.GetAllocationId() << std::setw(25)
                << address.GetNetworkInterfaceId() << std::endl;
        }
    } else {
        std::cerr << "Failed to describe Elastic IP addresses:" <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DescribeAddresses](#) 참조하세요.

## DescribeAvailabilityZones

다음 코드 예시는 DescribeAvailabilityZones의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

///DescribeAvailabilityZones
/*!
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
*/
int AwsDoc::EC2::describeAvailabilityZones(const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeAvailabilityZonesRequest request;
    Aws::EC2::Model::DescribeAvailabilityZonesOutcome outcome =
    ec2Client.DescribeAvailabilityZones(request);

    if (outcome.IsSuccess()) {
        std::cout << std::left <<
                std::setw(32) << "ZoneName" <<
                std::setw(20) << "State" <<
                std::setw(32) << "Region" << std::endl;

        const auto &zones =
            outcome.GetResult().GetAvailabilityZones();

        for (const auto &zone: zones) {
            Aws::String stateString =

            Aws::EC2::Model::AvailabilityZoneStateMapper::GetNameForAvailabilityZoneState(
                zone.GetState());
            std::cout << std::left <<
                    std::setw(32) << zone.GetZoneName() <<
                    std::setw(20) << stateString <<
                    std::setw(32) << zone.GetRegionName() << std::endl;
        }
    } else {
        std::cerr << "Failed to describe availability zones:" <<

```

```

        outcome.GetError().GetMessage() << std::endl;

    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DescribeAvailabilityZones](#)를 참조하세요.

## DescribeInstances

다음 코드 예시는 DescribeInstances의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Describe all Amazon Elastic Compute Cloud (Amazon EC2) instances associated with
an account.
/*!
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::EC2::describeInstances(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeInstancesRequest request;
    bool header = false;
    bool done = false;
    while (!done) {
        Aws::EC2::Model::DescribeInstancesOutcome outcome =
ec2Client.DescribeInstances(request);
        if (outcome.IsSuccess()) {
            if (!header) {
                std::cout << std::left <<
                    std::setw(48) << "Name" <<
                    std::setw(20) << "ID" <<

```

```

        std::setw(25) << "Ami" <<
        std::setw(15) << "Type" <<
        std::setw(15) << "State" <<
        std::setw(15) << "Monitoring" << std::endl;
    header = true;
}

const std::vector<Aws::EC2::Model::Reservation> &reservations =
    outcome.GetResult().GetReservations();

for (const auto &reservation: reservations) {
    const std::vector<Aws::EC2::Model::Instance> &instances =
        reservation.GetInstances();
    for (const auto &instance: instances) {
        Aws::String instanceStateString =

Aws::EC2::Model::InstanceStateNameMapper::GetNameForInstanceStateName(
            instance.GetState().GetName());

        Aws::String typeString =

Aws::EC2::Model::InstanceTypeMapper::GetNameForInstanceType(
            instance.GetInstanceType());

        Aws::String monitorString =

Aws::EC2::Model::MonitoringStateMapper::GetNameForMonitoringState(
            instance.GetMonitoring().GetState());
        Aws::String name = "Unknown";

        const std::vector<Aws::EC2::Model::Tag> &tags =
instance.GetTags();
        auto nameIter = std::find_if(tags.cbegin(), tags.cend(),
            [](const Aws::EC2::Model::Tag &tag)
{
                return tag.GetKey() == "Name";
            });
        if (nameIter != tags.cend()) {
            name = nameIter->GetValue();
        }
        std::cout <<
            std::setw(48) << name <<
            std::setw(20) << instance.GetInstanceId() <<
            std::setw(25) << instance.GetImageId() <<

```

```

        std::setw(15) << typeString <<
        std::setw(15) << instanceStateString <<
        std::setw(15) << monitorString << std::endl;
    }
}

if (!outcome.GetResult().GetNextToken().empty()) {
    request.SetNextToken(outcome.GetResult().GetNextToken());
} else {
    done = true;
}
} else {
    std::cerr << "Failed to describe EC2 instances:" <<
               outcome.GetError().GetMessage() << std::endl;
    return false;
}
}

return true;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DescribeInstances](#) 참조하세요.

## DescribeKeyPairs

다음 코드 예시는 DescribeKeyPairs의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Describe all Amazon Elastic Compute Cloud (Amazon EC2) instance key pairs.
/*!
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
 */
bool AwsDoc::EC2::describeKeyPairs(

```

```

    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeKeyPairsRequest request;

    Aws::EC2::Model::DescribeKeyPairsOutcome outcome =
    ec2Client.DescribeKeyPairs(request);
    if (outcome.IsSuccess()) {
        std::cout << std::left <<
            std::setw(32) << "Name" <<
            std::setw(64) << "Fingerprint" << std::endl;

        const std::vector<Aws::EC2::Model::KeyPairInfo> &key_pairs =
            outcome.GetResult().GetKeyPairs();
        for (const auto &key_pair: key_pairs) {
            std::cout << std::left <<
                std::setw(32) << key_pair.GetKeyName() <<
                std::setw(64) << key_pair.GetKeyFingerprint() << std::endl;
        }
    } else {
        std::cerr << "Failed to describe key pairs:" <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DescribeKeyPairs](#)를 참조하세요.

## DescribeRegions

다음 코드 예시는 DescribeRegions의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Describe all Amazon Elastic Compute Cloud (Amazon EC2) Regions.

```

```

    /*!
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */
bool AwsDoc::EC2::describeRegions(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::DescribeRegionsRequest request;
    Aws::EC2::Model::DescribeRegionsOutcome outcome =
    ec2Client.DescribeRegions(request);
    if (outcome.IsSuccess()) {
        std::cout << std::left <<
            std::setw(32) << "RegionName" <<
            std::setw(64) << "Endpoint" << std::endl;

        const auto &regions = outcome.GetResult().GetRegions();
        for (const auto &region: regions) {
            std::cout << std::left <<
                std::setw(32) << region.GetRegionName() <<
                std::setw(64) << region.GetEndpoint() << std::endl;
        }
    } else {
        std::cerr << "Failed to describe regions:" <<
            outcome.GetError().GetMessage() << std::endl;
    }

    std::cout << std::endl;

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DescribeRegions](#)를 참조하세요.

## DescribeSecurityGroups

다음 코드 예시는 DescribeSecurityGroups의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Describe all Amazon Elastic Compute Cloud (Amazon EC2) security groups, or a
specific group.
/*!
\param groupId: A group ID, ignored if empty.
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::EC2::describeSecurityGroups(const Aws::String &groupId,
                                         const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::DescribeSecurityGroupsRequest request;

    if (!groupId.empty()) {
        request.AddGroupIds(groupId);
    }

    Aws::String nextToken;
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::EC2::Model::DescribeSecurityGroupsOutcome outcome =
ec2Client.DescribeSecurityGroups(request);
        if (outcome.IsSuccess()) {
            std::cout << std::left <<
                std::setw(32) << "Name" <<
                std::setw(30) << "GroupId" <<
                std::setw(30) << "VpcId" <<
                std::setw(64) << "Description" << std::endl;

            const std::vector<Aws::EC2::Model::SecurityGroup> &securityGroups =
                outcome.GetResult().GetSecurityGroups();

```

```

        for (const auto &securityGroup: securityGroups) {
            std::cout << std::left <<
                std::setw(32) << securityGroup.GetGroupName() <<
                std::setw(30) << securityGroup.GetGroupId() <<
                std::setw(30) << securityGroup.GetVpcId() <<
                std::setw(64) << securityGroup.GetDescription() <<
                std::endl;
        }
    } else {
        std::cerr << "Failed to describe security groups:" <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    nextToken = outcome.GetResult().GetNextToken();
} while (!nextToken.empty());

return true;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DescribeSecurityGroups](#)를 참조하세요.

## MonitorInstances

다음 코드 예시는 MonitorInstances의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

/*! Enable detailed monitoring for an Amazon Elastic Compute Cloud (Amazon EC2)
instance.
*/
\param instanceId: An EC2 instance ID.
\param clientConfiguration: AWS client configuration.

```

```

    \return bool: Function succeeded.
    */
bool AwsDoc::EC2::enableMonitoring(const Aws::String &instanceId,
                                   const Aws::Client::ClientConfiguration
                                   &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::MonitorInstancesRequest request;
    request.AddInstanceIds(instanceId);
    request.SetDryRun(true);

    Aws::EC2::Model::MonitorInstancesOutcome dryRunOutcome =
ec2Client.MonitorInstances(request);
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
            << "Failed dry run to enable monitoring on instance. A dry run
should trigger an error."
            <<
            std::endl;
        return false;
    } else if (dryRunOutcome.GetError().GetErrorType()
               != Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cerr << "Failed dry run to enable monitoring on instance " <<
            instanceId << ": " << dryRunOutcome.GetError().GetMessage() <<
            std::endl;
        return false;
    }

    request.SetDryRun(false);
    Aws::EC2::Model::MonitorInstancesOutcome monitorInstancesOutcome =
ec2Client.MonitorInstances(request);
    if (!monitorInstancesOutcome.IsSuccess()) {
        std::cerr << "Failed to enable monitoring on instance " <<
            instanceId << ": " <<
            monitorInstancesOutcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully enabled monitoring on instance " <<
            instanceId << std::endl;
    }

    return monitorInstancesOutcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [MonitorInstances](#)를 참조하세요.

## RebootInstances

다음 코드 예시는 RebootInstances의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Reboot an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
    \param instanceID: An EC2 instance ID.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
 */
bool AwsDoc::EC2::rebootInstance(const Aws::String &instanceId,
                                const Aws::Client::ClientConfiguration
                                &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::RebootInstancesRequest request;
    request.AddInstanceIds(instanceId);
    request.SetDryRun(true);

    Aws::EC2::Model::RebootInstancesOutcome dry_run_outcome =
    ec2Client.RebootInstances(request);
    if (dry_run_outcome.IsSuccess()) {
        std::cerr
            << "Failed dry run to reboot on instance. A dry run should trigger
an error."
            <<
            std::endl;
        return false;
    } else if (dry_run_outcome.GetError().GetErrorType()
               != Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to reboot instance " << instanceId << ": "
            << dry_run_outcome.GetError().GetMessage() << std::endl;
        return false;
    }
}

```

```

    request.SetDryRun(false);
    Aws::EC2::Model::RebootInstancesOutcome outcome =
    ec2Client.RebootInstances(request);
    if (!outcome.IsSuccess()) {
        std::cout << "Failed to reboot instance " << instanceId << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully rebooted instance " << instanceId <<
            std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [RebootInstances](#)를 참조하세요.

## ReleaseAddress

다음 코드 예시는 ReleaseAddress의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Release an Elastic IP address.
/*!
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::EC2::releaseAddress(const Aws::String &allocationID,
                                const Aws::Client::ClientConfiguration
                                &clientConfiguration) {
    Aws::EC2::EC2Client ec2(clientConfiguration);

    Aws::EC2::Model::ReleaseAddressRequest request;
    request.SetAllocationId(allocationID);
}

```

```

    Aws::EC2::Model::ReleaseAddressOutcome outcome = ec2.ReleaseAddress(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to release Elastic IP address " <<
            allocationID << ":" << outcome.GetError().GetMessage() <<
            std::endl;
    } else {
        std::cout << "Successfully released Elastic IP address " <<
            allocationID << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ReleaseAddress](#)를 참조하세요.

## RunInstances

다음 코드 예시는 RunInstances의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Launch an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
    \param instanceName: A name for the EC2 instance.
    \param amiId: An Amazon Machine Image (AMI) identifier.
    \param[out] instanceID: String to return the instance ID.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::EC2::runInstance(const Aws::String &instanceName,
                              const Aws::String &amiId,
                              Aws::String &instanceID,
                              const Aws::Client::ClientConfiguration
                              &clientConfiguration) {

```

```

    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::RunInstancesRequest runRequest;
    runRequest.SetImageId(amiId);
    runRequest.SetInstanceType(Aws::EC2::Model::InstanceType::t1_micro);
    runRequest.SetMinCount(1);
    runRequest.SetMaxCount(1);

    Aws::EC2::Model::RunInstancesOutcome runOutcome = ec2Client.RunInstances(
        runRequest);
    if (!runOutcome.IsSuccess()) {
        std::cerr << "Failed to launch EC2 instance " << instanceName <<
            " based on ami " << amiId << ":" <<
            runOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    const Aws::Vector<Aws::EC2::Model::Instance> &instances =
runOutcome.GetResult().GetInstances();
    if (instances.empty()) {
        std::cerr << "Failed to launch EC2 instance " << instanceName <<
            " based on ami " << amiId << ":" <<
            runOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    instanceID = instances[0].GetInstanceId();

    return true;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [RunInstances](#)를 참조하세요.

## StartInstances

다음 코드 예시는 StartInstances의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Start an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
    \param instanceID: An EC2 instance ID.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::EC2::startInstance(const Aws::String &instanceId,
                                const Aws::Client::ClientConfiguration
                                &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::StartInstancesRequest startRequest;
    startRequest.AddInstanceIds(instanceId);
    startRequest.SetDryRun(true);

    Aws::EC2::Model::StartInstancesOutcome dryRunOutcome =
    ec2Client.StartInstances(startRequest);
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
            << "Failed dry run to start instance. A dry run should trigger an
error."
            << std::endl;
        return false;
    } else if (dryRunOutcome.GetError().GetErrorType() !=
        Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to start instance " << instanceId << ": "
            << dryRunOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    startRequest.SetDryRun(false);
    Aws::EC2::Model::StartInstancesOutcome startInstancesOutcome =
    ec2Client.StartInstances(startRequest);

```

```

    if (!startInstancesOutcome.IsSuccess()) {
        std::cout << "Failed to start instance " << instanceId << ": " <<
            startInstancesOutcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully started instance " << instanceId <<
            std::endl;
    }

    return startInstancesOutcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [StartInstances](#)를 참조하세요.

## StopInstances

다음 코드 예시는 StopInstances의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Stop an EC2 instance.
/*!
    \param instanceID: An EC2 instance ID.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
 */
bool AwsDoc::EC2::stopInstance(const Aws::String &instanceId,
                               const Aws::Client::ClientConfiguration
    &clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::StopInstancesRequest request;
    request.AddInstanceIds(instanceId);
    request.SetDryRun(true);

    Aws::EC2::Model::StopInstancesOutcome dryRunOutcome =
    ec2Client.StopInstances(request);
}

```

```

    if (dryRunOutcome.IsSuccess()) {
        std::cerr
            << "Failed dry run to stop instance. A dry run should trigger an
error."
            << std::endl;
        return false;
    } else if (dryRunOutcome.GetError().GetErrorType() !=
        Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to stop instance " << instanceId << ": "
            << dryRunOutcome.GetError().GetMessage() << std::endl;
        return false;
    }

    request.SetDryRun(false);
    Aws::EC2::Model::StopInstancesOutcome outcome =
ec2Client.StopInstances(request);
    if (!outcome.IsSuccess()) {
        std::cout << "Failed to stop instance " << instanceId << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Successfully stopped instance " << instanceId <<
            std::endl;
    }

    return outcome.IsSuccess();
}

void PrintUsage() {
    std::cout << "Usage: run_start_stop_instance <instance_id> <start|stop>" <<
        std::endl;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [StopInstances](#)를 참조하세요.

## TerminateInstances

다음 코드 예시는 TerminateInstances의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
#!/ Terminate an Amazon Elastic Compute Cloud (Amazon EC2) instance.
/*!
  \param instanceID: An EC2 instance ID.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::EC2::terminateInstances(const Aws::String &instanceID,
                                     const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);

    Aws::EC2::Model::TerminateInstancesRequest request;
    request.SetInstanceIds({instanceID});

    Aws::EC2::Model::TerminateInstancesOutcome outcome =
        ec2Client.TerminateInstances(request);
    if (outcome.IsSuccess()) {
        std::cout << "Ec2 instance '" << instanceID <<
            "' was terminated." << std::endl;
    } else {
        std::cerr << "Failed to terminate ec2 instance " << instanceID <<
            ", " <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [TerminateInstances](#)를 참조하세요.

## UnmonitorInstances

다음 코드 예시는 UnmonitorInstances의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Disable monitoring for an EC2 instance.
/*!
    \param instanceId: An EC2 instance ID.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::EC2::disableMonitoring(const Aws::String &instanceId,
                                    const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::EC2::EC2Client ec2Client(clientConfiguration);
    Aws::EC2::Model::UnmonitorInstancesRequest unrequest;
    unrequest.AddInstanceIds(instanceId);
    unrequest.SetDryRun(true);

    Aws::EC2::Model::UnmonitorInstancesOutcome dryRunOutcome =
    ec2Client.UnmonitorInstances(unrequest);
    if (dryRunOutcome.IsSuccess()) {
        std::cerr
            << "Failed dry run to disable monitoring on instance. A dry run
should trigger an error."
            <<
            std::endl;
        return false;
    } else if (dryRunOutcome.GetError().GetErrorType() !=
        Aws::EC2::EC2Errors::DRY_RUN_OPERATION) {
        std::cout << "Failed dry run to disable monitoring on instance " <<
            instanceId << ": " << dryRunOutcome.GetError().GetMessage() <<
            std::endl;
        return false;
    }
}

```

```

    unrequest.SetDryRun(false);
    Aws::EC2::Model::UnmonitorInstancesOutcome unmonitorInstancesOutcome =
    ec2Client.UnmonitorInstances(unrequest);
    if (!unmonitorInstancesOutcome.IsSuccess()) {
        std::cout << "Failed to disable monitoring on instance " << instanceId
            << ": " << unmonitorInstancesOutcome.GetError().GetMessage() <<
            std::endl;
    } else {
        std::cout << "Successfully disable monitoring on instance " <<
            instanceId << std::endl;
    }

    return unmonitorInstancesOutcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [UnmonitorInstances](#)를 참조하세요.

## SDK for C++를 사용한 EventBridge 예제

다음 코드 예제에서는 EventBridge와 AWS SDK for C++ 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

#### PutEvents

다음 코드 예시는 PutEvents의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

필수 파일을 포함합니다.

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutEventsRequest.h>
#include <aws/events/model/PutEventsResult.h>
#include <aws/core/Utils/Outcome.h>
#include <iostream>
```

이벤트를 전송합니다.

```
Aws::CloudWatchEvents::EventBridgeClient cwe;

Aws::CloudWatchEvents::Model::PutEventsRequestEntry event_entry;
event_entry.SetDetail(MakeDetails(event_key, event_value));
event_entry.SetDetailType("sampleSubmitted");
event_entry.AddResources(resource_arn);
event_entry.SetSource("aws-sdk-cpp-cloudwatch-example");

Aws::CloudWatchEvents::Model::PutEventsRequest request;
request.AddEntries(event_entry);

auto outcome = cwe.PutEvents(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to post CloudWatch event: " <<
        outcome.GetError().GetMessage() << std::endl;
}
else
{
    std::cout << "Successfully posted CloudWatch event" << std::endl;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [PutEvents](#)를 참조하세요.

## PutRule

다음 코드 예시는 PutRule의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

필수 파일을 포함합니다.

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutRuleRequest.h>
#include <aws/events/model/PutRuleResult.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

규칙을 생성합니다.

```
Aws::CloudWatchEvents::EventBridgeClient cwe;
Aws::CloudWatchEvents::Model::PutRuleRequest request;
request.SetName(rule_name);
request.SetRoleArn(role_arn);
request.SetScheduleExpression("rate(5 minutes)");
request.SetState(Aws::CloudWatchEvents::Model::RuleState::ENABLED);

auto outcome = cwe.PutRule(request);
if (!outcome.IsSuccess())
{
    std::cout << "Failed to create CloudWatch events rule " <<
        rule_name << ": " << outcome.GetError().GetMessage() <<
        std::endl;
}
else
```

```
{
    std::cout << "Successfully created CloudWatch events rule " <<
        rule_name << " with resulting Arn " <<
        outcome.GetResult().GetRuleArn() << std::endl;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [PutRule](#)을 참조하세요.

## PutTargets

다음 코드 예시는 PutTargets의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

필수 파일을 포함합니다.

```
#include <aws/core/Aws.h>
#include <aws/events/EventBridgeClient.h>
#include <aws/events/model/PutTargetsRequest.h>
#include <aws/events/model/PutTargetsResult.h>
#include <aws/core/Utils/Outcome.h>
#include <iostream>
```

대상을 추가합니다.

```
Aws::CloudWatchEvents::EventBridgeClient cwe;

Aws::CloudWatchEvents::Model::Target target;
target.SetArn(lambda_arn);
target.SetId(target_id);

Aws::CloudWatchEvents::Model::PutTargetsRequest request;
```

```

request.SetRule(rule_name);
request.AddTargets(target);

auto putTargetsOutcome = cwe.PutTargets(request);
if (!putTargetsOutcome.IsSuccess())
{
    std::cout << "Failed to create CloudWatch events target for rule "
                << rule_name << ": " <<
                putTargetsOutcome.GetError().GetMessage() << std::endl;
}
else
{
    std::cout <<
        "Successfully created CloudWatch events target for rule "
        << rule_name << std::endl;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [PutTargets](#)를 참조하세요.

## AWS Glue SDK for C++를 사용한 예제

다음 코드 예제에서는 Glue와 AWS SDK for C++ 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](#)
- [기본 사항](#)
- [작업](#)

# 시작하기

안녕하세요 AWS Glue

다음 코드 예제에서는 AWS Glue를 사용하여 시작하는 방법을 보여 줍니다.

SDK for C++

## Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

CMakeLists.txt CMake 파일의 코드입니다.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS glue)

# Set this project's name.
project("hello_glue")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
  for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
      "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
  endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})
```

```

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory for
    running and debugging.

    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you may
    need to uncomment this

                                # and set the proper subdirectory to the
    executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
    ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
    hello_glue.cpp)

target_link_libraries(${PROJECT_NAME}
    ${AWSSDK_LINK_LIBRARIES})

```

hello\_glue.cpp 소스 파일의 코드입니다.

```

#include <aws/core/Aws.h>
#include <aws/glue/GlueClient.h>
#include <aws/glue/model/ListJobsRequest.h>
#include <iostream>

/*
 * A "Hello Glue" starter application which initializes an AWS Glue client and
 * lists the
 * AWS Glue job definitions.
 *
 * main function
 *
 * Usage: 'hello_glue'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
}

```

```

int result = 0;
{
    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient glueClient(clientConfig);

    std::vector<Aws::String> jobs;

    Aws::String nextToken; // Used for pagination.
    do {
        Aws::Glue::Model::ListJobsRequest listJobsRequest;
        if (!nextToken.empty()) {
            listJobsRequest.SetNextToken(nextToken);
        }

        Aws::Glue::Model::ListJobsOutcome listRunsOutcome = glueClient.ListJobs(
            listJobsRequest);

        if (listRunsOutcome.IsSuccess()) {
            const std::vector<Aws::String> &jobNames =
listRunsOutcome.GetResult().GetJobNames();
            jobs.insert(jobs.end(), jobNames.begin(), jobNames.end());

            nextToken = listRunsOutcome.GetResult().GetNextToken();
        } else {
            std::cerr << "Error listing jobs. "
                << listRunsOutcome.GetError().GetMessage()
                << std::endl;
            result = 1;
            break;
        }
    } while (!nextToken.empty());

    std::cout << "Your account has " << jobs.size() << " jobs."
        << std::endl;
    for (size_t i = 0; i < jobs.size(); ++i) {
        std::cout << "    " << i + 1 << ". " << jobs[i] << std::endl;
    }
}
Aws::ShutdownAPI(options); // Should only be called once.
return result;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ListJobs](#)를 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 C++

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예제 리포지토리](#)에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
//! Scenario which demonstrates using AWS Glue to add a crawler and run a job.
/*!
  \sa runGettingStartedWithGlueScenario()
  \param bucketName: An S3 bucket created in the setup.
  \param roleName: An AWS Identity and Access Management (IAM) role created in the
  setup.
  \param clientConfig: AWS client configuration.
  \return bool: Successful completion.
 */

bool AwsDoc::Glue::runGettingStartedWithGlueScenario(const Aws::String &bucketName,
                                                    const Aws::String &roleName,
```

```

const
Aws::Client::ClientConfiguration &clientConfig) {
    Aws::Glue::GlueClient client(clientConfig);

    Aws::String roleArn;
    if (!getRoleArn(roleName, roleArn, clientConfig)) {
        std::cerr << "Error getting role ARN for role." << std::endl;
        return false;
    }

    // 1. Upload the job script to the S3 bucket.
    {
        std::cout << "Uploading the job script '"
                    << AwsDoc::Glue::PYTHON_SCRIPT
                    << "'." << std::endl;

        if (!AwsDoc::Glue::uploadFile(bucketName,
                                       AwsDoc::Glue::PYTHON_SCRIPT_PATH,
                                       AwsDoc::Glue::PYTHON_SCRIPT,
                                       clientConfig)) {
            std::cerr << "Error uploading the job file." << std::endl;
            return false;
        }
    }

    // 2. Create a crawler.
    {
        Aws::Glue::Model::S3Target s3Target;
        s3Target.SetPath("s3://crawler-public-us-east-1/flight/2016/csv");
        Aws::Glue::Model::CrawlerTargets crawlerTargets;
        crawlerTargets.AddS3Targets(s3Target);

        Aws::Glue::Model::CreateCrawlerRequest request;
        request.SetTargets(crawlerTargets);
        request.SetName(CRAWLER_NAME);
        request.SetDatabaseName(CRAWLER_DATABASE_NAME);
        request.SetTablePrefix(CRAWLER_DATABASE_PREFIX);
        request.SetRole(roleArn);

        Aws::Glue::Model::CreateCrawlerOutcome outcome =
        client.CreateCrawler(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully created the crawler." << std::endl;
        }
    }
}

```

```

    }
    else {
        std::cerr << "Error creating a crawler. " <<
outcome.GetError().GetMessage()
        << std::endl;
        deleteAssets("", CRAWLER_DATABASE_NAME, "", bucketName, clientConfig);
        return false;
    }
}

// 3. Get a crawler.
{
    Aws::Glue::Model::GetCrawlerRequest request;
    request.SetName(CRAWLER_NAME);

    Aws::Glue::Model::GetCrawlerOutcome outcome = client.GetCrawler(request);

    if (outcome.IsSuccess()) {
        Aws::Glue::Model::CrawlerState crawlerState =
outcome.GetResult().GetCrawler().GetState();
        std::cout << "Retrieved crawler with state " <<
            Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                crawlerState)
            << "." << std::endl;
    }
    else {
        std::cerr << "Error retrieving a crawler. "
            << outcome.GetError().GetMessage() << std::endl;
        deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
            clientConfig);
        return false;
    }
}

// 4. Start a crawler.
{
    Aws::Glue::Model::StartCrawlerRequest request;
    request.SetName(CRAWLER_NAME);

    Aws::Glue::Model::StartCrawlerOutcome outcome =
client.StartCrawler(request);

    if (outcome.IsSuccess() || (Aws::Glue::GlueErrors::CRAWLER_RUNNING ==

```

```

        outcome.GetError().GetErrorType())) {
    if (!outcome.IsSuccess()) {
        std::cout << "Crawler was already started." << std::endl;
    }
    else {
        std::cout << "Successfully started crawler." << std::endl;
    }

    std::cout << "This may take a while to run." << std::endl;

    Aws::Glue::Model::CrawlerState crawlerState =
    Aws::Glue::Model::CrawlerState::NOT_SET;
    int iterations = 0;
    while (Aws::Glue::Model::CrawlerState::READY != crawlerState) {
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++iterations;
        if ((iterations % 10) == 0) { // Log status every 10 seconds.
            std::cout << "Crawler status " <<

    Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
        crawlerState)
        << ". After " << iterations
        << " seconds elapsed."
        << std::endl;
        }
        Aws::Glue::Model::GetCrawlerRequest getCrawlerRequest;
        getCrawlerRequest.SetName(CRAWLER_NAME);

        Aws::Glue::Model::GetCrawlerOutcome getCrawlerOutcome =
    client.GetCrawler(
        getCrawlerRequest);

        if (getCrawlerOutcome.IsSuccess()) {
            crawlerState =
    getCrawlerOutcome.GetResult().GetCrawler().GetState();
        }
        else {
            std::cerr << "Error getting crawler.  "
                << getCrawlerOutcome.GetError().GetMessage() <<
    std::endl;
            break;
        }
    }
}

```

```

        if (Aws::Glue::Model::CrawlerState::READY == crawlerState) {
            std::cout << "Crawler finished running after " << iterations
                << " seconds."
                << std::endl;
        }
    }
    else {
        std::cerr << "Error starting a crawler. "
            << outcome.GetError().GetMessage()
            << std::endl;

        deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
            clientConfig);
        return false;
    }
}

// 5. Get a database.
{
    Aws::Glue::Model::GetDatabaseRequest request;
    request.SetName(CRAWLER_DATABASE_NAME);

    Aws::Glue::Model::GetDatabaseOutcome outcome = client.GetDatabase(request);

    if (outcome.IsSuccess()) {
        const Aws::Glue::Model::Database &database =
outcome.GetResult().GetDatabase();

        std::cout << "Successfully retrieve the database\n" <<
            database.Jsonize().View().WriteReadable() << "'. " <<
std::endl;
    }
    else {
        std::cerr << "Error getting the database. "
            << outcome.GetError().GetMessage() << std::endl;
        deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
            clientConfig);
        return false;
    }
}

// 6. Get tables.
Aws::String tableName;
{

```

```

    Aws::Glue::Model::GetTablesRequest request;
    request.SetDatabaseName(CRAWLER_DATABASE_NAME);
    std::vector<Aws::Glue::Model::Table> all_tables;
    Aws::String nextToken; // Used for pagination.
    do {
        Aws::Glue::Model::GetTablesOutcome outcome = client.GetTables(request);

        if (outcome.IsSuccess()) {
            const std::vector<Aws::Glue::Model::Table> &tables =
outcome.GetResult().GetTableList();
            all_tables.insert(all_tables.end(), tables.begin(), tables.end());
            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cerr << "Error getting the tables. "
                << outcome.GetError().GetMessage()
                << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                clientConfig);
            return false;
        }
    } while (!nextToken.empty());

    std::cout << "The database contains " << all_tables.size()
        << (all_tables.size() == 1 ?
            " table." : "tables.") << std::endl;
    std::cout << "Here is a list of the tables in the database.";
    for (size_t index = 0; index < all_tables.size(); ++index) {
        std::cout << "    " << index + 1 << ": " << all_tables[index].GetName()
            << std::endl;
    }

    if (!all_tables.empty()) {
        int tableIndex = askQuestionForIntRange(
            "Enter an index to display the database detail ",
            1, static_cast<int>(all_tables.size()));
        std::cout << all_tables[tableIndex - 1].Jsonize().View().WriteReadable()
            << std::endl;

        tableName = all_tables[tableIndex - 1].GetName();
    }
}

// 7. Create a job.

```

```

{
    Aws::Glue::Model::CreateJobRequest request;
    request.SetName(JOB_NAME);
    request.SetRole(roleArn);
    request.SetGlueVersion(GLUE_VERSION);

    Aws::Glue::Model::JobCommand command;
    command.SetName(JOB_COMMAND_NAME);
    command.SetPythonVersion(JOB_PYTHON_VERSION);
    command.SetScriptLocation(
        Aws::String("s3://") + bucketName + "/" + PYTHON_SCRIPT);
    request.SetCommand(command);

    Aws::Glue::Model::CreateJobOutcome outcome = client.CreateJob(request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully created the job." << std::endl;
    }
    else {
        std::cerr << "Error creating the job. " <<
outcome.GetError().GetMessage()
        << std::endl;
        deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
            clientConfig);
        return false;
    }
}

// 8. Start a job run.
{
    Aws::Glue::Model::StartJobRunRequest request;
    request.SetJobName(JOB_NAME);

    Aws::Map<Aws::String, Aws::String> arguments;
    arguments["--input_database"] = CRAWLER_DATABASE_NAME;
    arguments["--input_table"] = tableName;
    arguments["--output_bucket_url"] = Aws::String("s3://") + bucketName + "/";
    request.SetArguments(arguments);

    Aws::Glue::Model::StartJobRunOutcome outcome = client.StartJobRun(request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully started the job." << std::endl;
    }
}

```

```

    Aws::String jobRunId = outcome.GetResult().GetJobRunId();

    int iterator = 0;
    bool done = false;
    while (!done) {
        ++iterator;
        std::this_thread::sleep_for(std::chrono::seconds(1));
        Aws::Glue::Model::GetJobRunRequest jobRunRequest;
        jobRunRequest.SetJobName(JOB_NAME);
        jobRunRequest.SetRunId(jobRunId);

        Aws::Glue::Model::GetJobRunOutcome jobRunOutcome = client.GetJobRun(
            jobRunRequest);

        if (jobRunOutcome.IsSuccess()) {
            const Aws::Glue::Model::JobRun &jobRun =
jobRunOutcome.GetResult().GetJobRun();
            Aws::Glue::Model::JobRunState jobRunState =
jobRun.GetJobRunState();

            if ((jobRunState == Aws::Glue::Model::JobRunState::STOPPED) ||
                (jobRunState == Aws::Glue::Model::JobRunState::FAILED) ||
                (jobRunState == Aws::Glue::Model::JobRunState::TIMEOUT)) {
                std::cerr << "Error running job. "
                    << jobRun.GetErrorMessage()
                    << std::endl;
                deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
                    bucketName,
                    clientConfig);
                return false;
            }
            else if (jobRunState ==
                Aws::Glue::Model::JobRunState::SUCCEEDED) {
                std::cout << "Job run succeeded after " << iterator <<
                    " seconds elapsed." << std::endl;
                done = true;
            }
            else if ((iterator % 10) == 0) { // Log status every 10 seconds.
                std::cout << "Job run status " <<

Aws::Glue::Model::JobRunStateMapper::GetNameForJobRunState(
                    jobRunState) <<
                ". " << iterator <<
                " seconds elapsed." << std::endl;
            }
        }
    }

```

```

        }
    }
    else {
        std::cerr << "Error retrieving job run state. "
                  << jobRunOutcome.GetError().GetMessage()
                  << std::endl;
        deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
                     bucketName, clientConfig);
        return false;
    }
}
}
else {
    std::cerr << "Error starting a job. " << outcome.GetError().GetMessage()
              << std::endl;
    deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME, bucketName,
                 clientConfig);
    return false;
}
}

// 9. List the output data stored in the S3 bucket.
{
    Aws::S3::S3Client s3Client;
    Aws::S3::Model::ListObjectsV2Request request;
    request.SetBucket(bucketName);
    request.SetPrefix(OUTPUT_FILE_PREFIX);

    Aws::String continuationToken; // Used for pagination.
    std::vector<Aws::S3::Model::Object> allObjects;
    do {
        if (!continuationToken.empty()) {
            request.SetContinuationToken(continuationToken);
        }
        Aws::S3::Model::ListObjectsV2Outcome outcome = s3Client.ListObjectsV2(
            request);

        if (outcome.IsSuccess()) {
            const std::vector<Aws::S3::Model::Object> &objects =
                outcome.GetResult().GetContents();
            allObjects.insert(allObjects.end(), objects.begin(), objects.end());
            continuationToken = outcome.GetResult().GetNextContinuationToken();
        }
        else {

```

```

        std::cerr << "Error listing objects. "
                    << outcome.GetError().GetMessage()
                    << std::endl;

        break;
    }
} while (!continuationToken.empty());

std::cout << "Data from your job is in " << allObjects.size() <<
           " files in the S3 bucket, " << bucketName << "." << std::endl;

for (size_t i = 0; i < allObjects.size(); ++i) {
    std::cout << "      " << i + 1 << ". " << allObjects[i].GetKey()
               << std::endl;
}

int objectIndex = askQuestionForIntRange(
    std::string(
        "Enter the number of a block to download it and see the
first ") +
    std::to_string(LINES_OF_RUN_FILE_TO_DISPLAY) +
    " lines of JSON output in the block: ", 1,
    static_cast<int>(allObjects.size()));

Aws::String objectKey = allObjects[objectIndex - 1].GetKey();

std::stringstream stringStream;
if (getObjectFromBucket(bucketName, objectKey, stringStream,
    clientConfig)) {
    for (int i = 0; i < LINES_OF_RUN_FILE_TO_DISPLAY && stringStream; ++i) {
        std::string line;
        std::getline(stringStream, line);
        std::cout << "      " << line << std::endl;
    }
}
else {
    deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME, bucketName,
        clientConfig);
    return false;
}
}

// 10. List all the jobs.
Aws::String jobName;
{

```

```

    Aws::Glue::Model::ListJobsRequest listJobsRequest;

    Aws::String nextToken;
    std::vector<Aws::String> allJobNames;

    do {
        if (!nextToken.empty()) {
            listJobsRequest.SetNextToken(nextToken);
        }
        Aws::Glue::Model::ListJobsOutcome listRunsOutcome = client.ListJobs(
            listJobsRequest);

        if (listRunsOutcome.IsSuccess()) {
            const std::vector<Aws::String> &jobNames =
listRunsOutcome.GetResult().GetJobNames();
            allJobNames.insert(allJobNames.end(), jobNames.begin(),
jobNames.end());
            nextToken = listRunsOutcome.GetResult().GetNextToken();
        }
        else {
            std::cerr << "Error listing jobs. "
                << listRunsOutcome.GetError().GetMessage()
                << std::endl;
        }
    } while (!nextToken.empty());
    std::cout << "Your account has " << allJobNames.size() << " jobs."
        << std::endl;
    for (size_t i = 0; i < allJobNames.size(); ++i) {
        std::cout << "    " << i + 1 << ". " << allJobNames[i] << std::endl;
    }
    int jobIndex = askQuestionForIntRange(
        Aws::String("Enter a number between 1 and ") +
        std::to_string(allJobNames.size()) +
        " to see the list of runs for a job: ",
        1, static_cast<int>(allJobNames.size()));

    jobName = allJobNames[jobIndex - 1];
}

// 11. Get the job runs for a job.
Aws::String jobRunID;
if (!jobName.empty()) {
    Aws::Glue::Model::GetJobRunsRequest getJobRunsRequest;
    getJobRunsRequest.SetJobName(jobName);
}

```

```

    Aws::String nextToken; // Used for pagination.
    std::vector<Aws::Glue::Model::JobRun> allJobRuns;
    do {
        if (!nextToken.empty()) {
            getJobRunsRequest.SetNextToken(nextToken);
        }
        Aws::Glue::Model::GetJobRunsOutcome jobRunsOutcome = client.GetJobRuns(
            getJobRunsRequest);

        if (jobRunsOutcome.IsSuccess()) {
            const std::vector<Aws::Glue::Model::JobRun> &jobRuns =
jobRunsOutcome.GetResult().GetJobRuns();
            allJobRuns.insert(allJobRuns.end(), jobRuns.begin(), jobRuns.end());

            nextToken = jobRunsOutcome.GetResult().GetNextToken();
        }
        else {
            std::cerr << "Error getting job runs. "
                << jobRunsOutcome.GetError().GetMessage()
                << std::endl;

            break;
        }
    } while (!nextToken.empty());

    std::cout << "There are " << allJobRuns.size() << " runs in the job '"
        <<
        jobName << "'." << std::endl;

    for (size_t i = 0; i < allJobRuns.size(); ++i) {
        std::cout << "    " << i + 1 << ". " << allJobRuns[i].GetJobName()
            << std::endl;
    }

    int runIndex = askQuestionForIntRange(
        Aws::String("Enter a number between 1 and ") +
        std::to_string(allJobRuns.size()) +
        " to see details for a run: ",
        1, static_cast<int>(allJobRuns.size()));
    jobRunID = allJobRuns[runIndex - 1].GetId();
}

// 12. Get a single job run.
if (!jobRunID.empty()) {

```

```

        Aws::Glue::Model::GetJobRunRequest jobRunRequest;
        jobRunRequest.SetJobName(jobName);
        jobRunRequest.SetRunId(jobRunID);

        Aws::Glue::Model::GetJobRunOutcome jobRunOutcome = client.GetJobRun(
            jobRunRequest);

        if (jobRunOutcome.IsSuccess()) {
            std::cout << "Displaying the job run JSON description." << std::endl;
            std::cout
                <<
jobRunOutcome.GetResult().GetJobRun().Jsonize().View().WriteReadable()
                << std::endl;
        }
        else {
            std::cerr << "Error get a job run. "
                << jobRunOutcome.GetError().GetMessage()
                << std::endl;
        }
    }

    return deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME, bucketName,
        clientConfig);
}

//! Cleanup routine to delete created assets.
/*!
    \sa deleteAssets()
    \param crawler: Name of an AWS Glue crawler.
    \param database: The name of an AWS Glue database.
    \param job: The name of an AWS Glue job.
    \param bucketName: The name of an S3 bucket.
    \param clientConfig: AWS client configuration.
    \return bool: Successful completion.
*/
bool AwsDoc::Glue::deleteAssets(const Aws::String &crawler, const Aws::String
    &database,
                                const Aws::String &job, const Aws::String
    &bucketName,
                                const Aws::Client::ClientConfiguration
    &clientConfig) {
    const Aws::Glue::GlueClient client(clientConfig);
    bool result = true;

```

```
// 13. Delete a job.
if (!job.empty()) {
    Aws::Glue::Model::DeleteJobRequest request;
    request.SetJobName(job);

    Aws::Glue::Model::DeleteJobOutcome outcome = client.DeleteJob(request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted the job." << std::endl;
    }
    else {
        std::cerr << "Error deleting the job. " <<
outcome.GetError().GetMessage()
        << std::endl;
        result = false;
    }
}

// 14. Delete a database.
if (!database.empty()) {
    Aws::Glue::Model::DeleteDatabaseRequest request;
    request.SetName(database);

    Aws::Glue::Model::DeleteDatabaseOutcome outcome = client.DeleteDatabase(
        request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted the database." << std::endl;
    }
    else {
        std::cerr << "Error deleting database. " <<
outcome.GetError().GetMessage()
        << std::endl;
        result = false;
    }
}

// 15. Delete a crawler.
if (!crawler.empty()) {
    Aws::Glue::Model::DeleteCrawlerRequest request;
    request.SetName(crawler);
```

```

        Aws::Glue::Model::DeleteCrawlerOutcome outcome =
client.DeleteCrawler(request);

        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted the crawler." << std::endl;
        }
        else {
            std::cerr << "Error deleting the crawler. "
                << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
    }

    // 16. Delete the job script and run data from the S3 bucket.
    result &= AwsDoc::Glue::deleteAllObjectsInS3Bucket(bucketName,
                                                        clientConfig);

    return result;
}

//! Routine which uploads a file to an S3 bucket.
/*!
\\sa uploadFile()
\param bucketName: An S3 bucket created in the setup.
\param filePath: The path of the file to upload.
\param fileName The name for the uploaded file.
\param clientConfig: AWS client configuration.
\return bool: Successful completion.
*/
bool
AwsDoc::Glue::uploadFile(const Aws::String &bucketName,
                        const Aws::String &filePath,
                        const Aws::String &fileName,
                        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3_client(clientConfig);

    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(fileName);

    std::shared_ptr<Aws::IOStream> inputData =
        Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                       filePath.c_str(),
                                       std::ios_base::in |
std::ios_base::binary);

```

```

    if (!*inputData) {
        std::cerr << "Error unable to read file " << filePath << std::endl;
        return false;
    }

    request.SetBody(inputData);

    Aws::S3::Model::PutObjectOutcome outcome =
        s3_client.PutObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutObject: " <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Added object '" << filePath << "' to bucket '"
            << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which deletes all objects in an S3 bucket.
/*!
    \sa deleteAllObjectsInS3Bucket()
    \param bucketName: The S3 bucket name.
    \param clientConfig: AWS client configuration.
    \return bool: Successful completion.
    */
bool AwsDoc::Glue::deleteAllObjectsInS3Bucket(const Aws::String &bucketName,
                                              const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::ListObjectsV2Request listObjectsRequest;
    listObjectsRequest.SetBucket(bucketName);

    Aws::String continuationToken; // Used for pagination.
    bool result = true;
    do {
        if (!continuationToken.empty()) {
            listObjectsRequest.SetContinuationToken(continuationToken);
        }
    }

```

```

        Aws::S3::Model::ListObjectsV2Outcome listObjectsOutcome =
client.ListObjectsV2(
    listObjectsRequest);

    if (listObjectsOutcome.IsSuccess()) {
        const std::vector<Aws::S3::Model::Object> &objects =
listObjectsOutcome.GetResult().GetContents();
        if (!objects.empty()) {
            Aws::S3::Model::DeleteObjectsRequest deleteObjectsRequest;
            deleteObjectsRequest.SetBucket(bucketName);

            std::vector<Aws::S3::Model::ObjectIdentifier> objectIdentifiers;
            for (const Aws::S3::Model::Object &object: objects) {
                objectIdentifiers.push_back(
                    Aws::S3::Model::ObjectIdentifier().WithKey(
                        object.GetKey()));
            }
            Aws::S3::Model::Delete objectsDelete;
            objectsDelete.SetObjects(objectIdentifiers);
            objectsDelete.SetQuiet(true);
            deleteObjectsRequest.SetDelete(objectsDelete);

            Aws::S3::Model::DeleteObjectsOutcome deleteObjectsOutcome =
                client.DeleteObjects(deleteObjectsRequest);

            if (!deleteObjectsOutcome.IsSuccess()) {
                std::cerr << "Error deleting objects. " <<
                    deleteObjectsOutcome.GetError().GetMessage() <<
std::endl;

                result = false;
                break;
            }
            else {
                std::cout << "Successfully deleted the objects." << std::endl;
            }
        }
    }
    else {
        std::cout << "No objects to delete in '" << bucketName << "'."
            << std::endl;
    }

    continuationToken =
listObjectsOutcome.GetResult().GetNextContinuationToken();

```

```

    }
    else {
        std::cerr << "Error listing objects. "
                  << listObjectsOutcome.GetError().GetMessage() << std::endl;
        result = false;
        break;
    }
} while (!continuationToken.empty());

return result;
}

//! Routine which retrieves an object from an S3 bucket.
/*!
  \sa getObjectFromBucket()
  \param bucketName: The S3 bucket name.
  \param objectKey: The object's name.
  \param objectStream: A stream to receive the retrieved data.
  \param clientConfig: AWS client configuration.
  \return bool: Successful completion.
  */
bool AwsDoc::Glue::getObjectFromBucket(const Aws::String &bucketName,
                                       const Aws::String &objectKey,
                                       std::ostream &objectStream,
                                       const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectOutcome outcome = client.GetObject(request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully retrieved '" << objectKey << "'." << std::endl;
        auto &body = outcome.GetResult().GetBody();
        objectStream << body.rdbuf();
    }
    else {
        std::cerr << "Error retrieving object. " << outcome.GetError().GetMessage()
                  << std::endl;
    }
}

```

```
    return outcome.IsSuccess();  
}
```

- API 세부 정보는 AWS SDK for C++ 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 CreateCrawler의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
// clientConfig.region = "us-east-1";

Aws::Glue::GlueClient client(clientConfig);

Aws::Glue::Model::S3Target s3Target;
s3Target.SetPath("s3://crawler-public-us-east-1/flight/2016/csv");
Aws::Glue::Model::CrawlerTargets crawlerTargets;
crawlerTargets.AddS3Targets(s3Target);

Aws::Glue::Model::CreateCrawlerRequest request;
request.SetTargets(crawlerTargets);
request.SetName(CRAWLER_NAME);
request.SetDatabaseName(CRAWLER_DATABASE_NAME);
request.SetTablePrefix(CRAWLER_DATABASE_PREFIX);
request.SetRole(roleArn);

Aws::Glue::Model::CreateCrawlerOutcome outcome =
client.CreateCrawler(request);

if (outcome.IsSuccess()) {
    std::cout << "Successfully created the crawler." << std::endl;
}
else {
    std::cerr << "Error creating a crawler. " <<
outcome.GetError().GetMessage()
    << std::endl;
    deleteAssets("", CRAWLER_DATABASE_NAME, "", bucketName, clientConfig);
    return false;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CreateCrawler](#)를 참조하세요.

## CreateJob

다음 코드 예시는 CreateJob의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
// clientConfig.region = "us-east-1";

Aws::Glue::GlueClient client(clientConfig);

Aws::Glue::Model::CreateJobRequest request;
request.SetName(JOB_NAME);
request.SetRole(roleArn);
request.SetGlueVersion(GLUE_VERSION);

Aws::Glue::Model::JobCommand command;
command.SetName(JOB_COMMAND_NAME);
command.SetPythonVersion(JOB_PYTHON_VERSION);
command.SetScriptLocation(
    Aws::String("s3://" + bucketName + "/" + PYTHON_SCRIPT);
request.SetCommand(command);

Aws::Glue::Model::CreateJobOutcome outcome = client.CreateJob(request);

if (outcome.IsSuccess()) {
    std::cout << "Successfully created the job." << std::endl;
}
else {
    std::cerr << "Error creating the job. " <<
outcome.GetError().GetMessage()
    << std::endl;
```

```

        deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                     clientConfig);
    return false;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CreateJob](#)을 참조하세요.

## DeleteCrawler

다음 코드 예시는 DeleteCrawler의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region in which the bucket was created
    (overrides config file).
    // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

    Aws::Glue::Model::DeleteCrawlerRequest request;
    request.SetName(crawler);

    Aws::Glue::Model::DeleteCrawlerOutcome outcome =
    client.DeleteCrawler(request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted the crawler." << std::endl;
    }
    else {
        std::cerr << "Error deleting the crawler. "
                  << outcome.GetError().GetMessage() << std::endl;
        result = false;
    }
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteCrawler](#)를 참조하세요.

## DeleteDatabase

다음 코드 예시는 DeleteDatabase의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
// clientConfig.region = "us-east-1";

Aws::Glue::GlueClient client(clientConfig);

Aws::Glue::Model::DeleteDatabaseRequest request;
request.SetName(database);

Aws::Glue::Model::DeleteDatabaseOutcome outcome = client.DeleteDatabase(
    request);

if (outcome.IsSuccess()) {
    std::cout << "Successfully deleted the database." << std::endl;
}
else {
    std::cerr << "Error deleting database. " <<
outcome.GetError().GetMessage()
    << std::endl;
    result = false;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteDatabase](#)를 참조하세요.

## DeleteJob

다음 코드 예시는 DeleteJob의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
// clientConfig.region = "us-east-1";

Aws::Glue::GlueClient client(clientConfig);

Aws::Glue::Model::DeleteJobRequest request;
request.SetJobName(job);

Aws::Glue::Model::DeleteJobOutcome outcome = client.DeleteJob(request);

if (outcome.IsSuccess()) {
    std::cout << "Successfully deleted the job." << std::endl;
}
else {
    std::cerr << "Error deleting the job. " <<
outcome.GetError().GetMessage()
    << std::endl;
    result = false;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteJob](#)을 참조하세요.

## GetCrawler

다음 코드 예시는 GetCrawler의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region in which the bucket was created
    (overrides config file).
    // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

    Aws::Glue::Model::GetCrawlerRequest request;
    request.SetName(CRAWLER_NAME);

    Aws::Glue::Model::GetCrawlerOutcome outcome = client.GetCrawler(request);

    if (outcome.IsSuccess()) {
        Aws::Glue::Model::CrawlerState crawlerState =
outcome.GetResult().GetCrawler().GetState();
        std::cout << "Retrieved crawler with state " <<
            Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
                crawlerState)
            << "." << std::endl;
    }
    else {
        std::cerr << "Error retrieving a crawler. "
            << outcome.GetError().GetMessage() << std::endl;
        deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
            clientConfig);
        return false;
    }

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetCrawler](#)를 참조하세요.

## GetDatabase

다음 코드 예시는 GetDatabase의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
// clientConfig.region = "us-east-1";

Aws::Glue::GlueClient client(clientConfig);

Aws::Glue::Model::GetDatabaseRequest request;
request.SetName(CRAWLER_DATABASE_NAME);

Aws::Glue::Model::GetDatabaseOutcome outcome = client.GetDatabase(request);

if (outcome.IsSuccess()) {
    const Aws::Glue::Model::Database &database =
outcome.GetResult().GetDatabase();

    std::cout << "Successfully retrieve the database\n" <<
        database.Jsonize().View().WriteReadable() << "." <<
std::endl;
}
else {
    std::cerr << "Error getting the database. "
        << outcome.GetError().GetMessage() << std::endl;
    deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
        clientConfig);
    return false;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetDatabase](#)를 참조하세요.

## GetJobRun

다음 코드 예시는 GetJobRun의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
// clientConfig.region = "us-east-1";

Aws::Glue::GlueClient client(clientConfig);

Aws::Glue::Model::GetJobRunRequest jobRunRequest;
jobRunRequest.SetJobName(jobName);
jobRunRequest.SetRunId(jobRunID);

Aws::Glue::Model::GetJobRunOutcome jobRunOutcome = client.GetJobRun(
    jobRunRequest);

if (jobRunOutcome.IsSuccess()) {
    std::cout << "Displaying the job run JSON description." << std::endl;
    std::cout
        <<
jobRunOutcome.GetResult().GetJobRun().Jsonize().View().WriteReadable()
        << std::endl;
}
else {
    std::cerr << "Error get a job run. "
        << jobRunOutcome.GetError().GetMessage()
        << std::endl;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetJobRun](#)을 참조하세요.

## GetJobRuns

다음 코드 예시는 GetJobRuns의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
// clientConfig.region = "us-east-1";

Aws::Glue::GlueClient client(clientConfig);

Aws::Glue::Model::GetJobRunsRequest getJobRunsRequest;
getJobRunsRequest.SetJobName(jobName);

Aws::String nextToken; // Used for pagination.
std::vector<Aws::Glue::Model::JobRun> allJobRuns;
do {
    if (!nextToken.empty()) {
        getJobRunsRequest.SetNextToken(nextToken);
    }
    Aws::Glue::Model::GetJobRunsOutcome jobRunsOutcome = client.GetJobRuns(
        getJobRunsRequest);

    if (jobRunsOutcome.IsSuccess()) {
        const std::vector<Aws::Glue::Model::JobRun> &jobRuns =
jobRunsOutcome.GetResult().GetJobRuns();
        allJobRuns.insert(allJobRuns.end(), jobRuns.begin(), jobRuns.end());

        nextToken = jobRunsOutcome.GetResult().GetNextToken();
    }
    else {
        std::cerr << "Error getting job runs. "
            << jobRunsOutcome.GetError().GetMessage()
            << std::endl;

        break;
    }
}
```

```
    }
    } while (!nextToken.empty());
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetJobRuns](#)를 참조하세요.

## GetTables

다음 코드 예시는 GetTables의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
// clientConfig.region = "us-east-1";

Aws::Glue::GlueClient client(clientConfig);

Aws::Glue::Model::GetTablesRequest request;
request.SetDatabaseName(CRAWLER_DATABASE_NAME);
std::vector<Aws::Glue::Model::Table> all_tables;
Aws::String nextToken; // Used for pagination.
do {
    Aws::Glue::Model::GetTablesOutcome outcome = client.GetTables(request);

    if (outcome.IsSuccess()) {
        const std::vector<Aws::Glue::Model::Table> &tables =
outcome.GetResult().GetTableList();
        all_tables.insert(all_tables.end(), tables.begin(), tables.end());
        nextToken = outcome.GetResult().GetNextToken();
    }
    else {
        std::cerr << "Error getting the tables. "
                    << outcome.GetError().GetMessage()
                    << std::endl;
```

```

        deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
                     clientConfig);
        return false;
    }
} while (!nextToken.empty());

std::cout << "The database contains " << all_tables.size()
           << (all_tables.size() == 1 ?
               " table." : "tables.") << std::endl;
std::cout << "Here is a list of the tables in the database.";
for (size_t index = 0; index < all_tables.size(); ++index) {
    std::cout << "    " << index + 1 << ": " << all_tables[index].GetName()
               << std::endl;
}

if (!all_tables.empty()) {
    int tableIndex = askQuestionForIntRange(
        "Enter an index to display the database detail ",
        1, static_cast<int>(all_tables.size()));
    std::cout << all_tables[tableIndex - 1].Jsonize().View().WriteReadable()
               << std::endl;

    tableName = all_tables[tableIndex - 1].GetName();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetTables](#)를 참조하세요.

## ListJobs

다음 코드 예시는 ListJobs의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
```

```

    // Optional: Set to the AWS Region in which the bucket was created
    (overrides config file).
    // clientConfig.region = "us-east-1";

    Aws::Glue::GlueClient client(clientConfig);

    Aws::Glue::Model::ListJobsRequest listJobsRequest;

    Aws::String nextToken;
    std::vector<Aws::String> allJobNames;

    do {
        if (!nextToken.empty()) {
            listJobsRequest.SetNextToken(nextToken);
        }
        Aws::Glue::Model::ListJobsOutcome listRunsOutcome = client.ListJobs(
            listJobsRequest);

        if (listRunsOutcome.IsSuccess()) {
            const std::vector<Aws::String> &jobNames =
listRunsOutcome.GetResult().GetJobNames();
            allJobNames.insert(allJobNames.end(), jobNames.begin(),
jobNames.end());
            nextToken = listRunsOutcome.GetResult().GetNextToken();
        }
        else {
            std::cerr << "Error listing jobs. "
                << listRunsOutcome.GetError().GetMessage()
                << std::endl;
        }
    } while (!nextToken.empty());

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ListJobs](#)를 참조하세요.

## StartCrawler

다음 코드 예시는 StartCrawler의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
// clientConfig.region = "us-east-1";

Aws::Glue::GlueClient client(clientConfig);

Aws::Glue::Model::StartCrawlerRequest request;
request.SetName(CRAWLER_NAME);

Aws::Glue::Model::StartCrawlerOutcome outcome =
client.StartCrawler(request);

if (outcome.IsSuccess() || (Aws::Glue::GlueErrors::CRAWLER_RUNNING ==
                           outcome.GetError().GetErrorType())) {
    if (!outcome.IsSuccess()) {
        std::cout << "Crawler was already started." << std::endl;
    }
    else {
        std::cout << "Successfully started crawler." << std::endl;
    }

    std::cout << "This may take a while to run." << std::endl;

    Aws::Glue::Model::CrawlerState crawlerState =
    Aws::Glue::Model::CrawlerState::NOT_SET;
    int iterations = 0;
    while (Aws::Glue::Model::CrawlerState::READY != crawlerState) {
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++iterations;
        if ((iterations % 10) == 0) { // Log status every 10 seconds.
            std::cout << "Crawler status " <<
```

```

Aws::Glue::Model::CrawlerStateMapper::GetNameForCrawlerState(
    crawlerState)
    << ". After " << iterations
    << " seconds elapsed."
    << std::endl;
}
Aws::Glue::Model::GetCrawlerRequest getCrawlerRequest;
getCrawlerRequest.SetName(CRAWLER_NAME);

Aws::Glue::Model::GetCrawlerOutcome getCrawlerOutcome =
client.GetCrawler(
    getCrawlerRequest);

if (getCrawlerOutcome.IsSuccess()) {
    crawlerState =
getCrawlerOutcome.GetResult().GetCrawler().GetState();
}
else {
    std::cerr << "Error getting crawler.  "
    << getCrawlerOutcome.GetError().GetMessage() <<
std::endl;
    break;
}

if (Aws::Glue::Model::CrawlerState::READY == crawlerState) {
    std::cout << "Crawler finished running after " << iterations
    << " seconds."
    << std::endl;
}
}
else {
    std::cerr << "Error starting a crawler.  "
    << outcome.GetError().GetMessage()
    << std::endl;

    deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, "", bucketName,
        clientConfig);
    return false;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [StartCrawler](#)를 참조하세요.

## StartJobRun

다음 코드 예시는 StartJobRun의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
// clientConfig.region = "us-east-1";

Aws::Glue::GlueClient client(clientConfig);

Aws::Glue::Model::StartJobRunRequest request;
request.SetJobName(JOB_NAME);

Aws::Map<Aws::String, Aws::String> arguments;
arguments["--input_database"] = CRAWLER_DATABASE_NAME;
arguments["--input_table"] = tableName;
arguments["--output_bucket_url"] = Aws::String("s3://") + bucketName + "/";
request.SetArguments(arguments);

Aws::Glue::Model::StartJobRunOutcome outcome = client.StartJobRun(request);

if (outcome.IsSuccess()) {
    std::cout << "Successfully started the job." << std::endl;

    Aws::String jobRunId = outcome.GetResult().GetJobRunId();

    int iterator = 0;
    bool done = false;
    while (!done) {
        ++iterator;
        std::this_thread::sleep_for(std::chrono::seconds(1));
        Aws::Glue::Model::GetJobRunRequest jobRunRequest;
        jobRunRequest.SetJobName(JOB_NAME);
        jobRunRequest.SetRunId(jobRunId);
```

```

        Aws::Glue::Model::GetJobRunOutcome jobRunOutcome = client.GetJobRun(
            jobRunRequest);

        if (jobRunOutcome.IsSuccess()) {
            const Aws::Glue::Model::JobRun &jobRun =
jobRunOutcome.GetResult().GetJobRun();
            Aws::Glue::Model::JobRunState jobRunState =
jobRun.GetJobRunState();

            if ((jobRunState == Aws::Glue::Model::JobRunState::STOPPED) ||
                (jobRunState == Aws::Glue::Model::JobRunState::FAILED) ||
                (jobRunState == Aws::Glue::Model::JobRunState::TIMEOUT)) {
                std::cerr << "Error running job. "
                    << jobRun.GetErrorMessage()
                    << std::endl;
                deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
                    bucketName,
                    clientConfig);
                return false;
            }
            else if (jobRunState ==
                Aws::Glue::Model::JobRunState::SUCCEEDED) {
                std::cout << "Job run succeeded after " << iterator <<
                    " seconds elapsed." << std::endl;
                done = true;
            }
            else if ((iterator % 10) == 0) { // Log status every 10 seconds.
                std::cout << "Job run status " <<

Aws::Glue::Model::JobRunStateMapper::GetNameForJobRunState(
                    jobRunState) <<
                    ". " << iterator <<
                    " seconds elapsed." << std::endl;
            }
        }
        else {
            std::cerr << "Error retrieving job run state. "
                << jobRunOutcome.GetError().GetMessage()
                << std::endl;
            deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME,
                bucketName, clientConfig);
            return false;
        }
    }
}

```

```

    }
}
else {
    std::cerr << "Error starting a job. " << outcome.GetError().GetMessage()
               << std::endl;
    deleteAssets(CRAWLER_NAME, CRAWLER_DATABASE_NAME, JOB_NAME, bucketName,
                  clientConfig);
    return false;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [StartJobRun](#)을 참조하세요.

## SDK for C++를 사용한 HealthImaging 예시

다음 코드 예제에서는 HealthImaging과 AWS SDK for C++ 함께를 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

- [시작하기](#)
- [작업](#)
- [시나리오](#)

## 시작하기

### HealthImaging 시작

다음 코드 예제에서는 HealthImaging을 사용하여 시작하는 방법을 보여줍니다.

## SDK for C++

CMakeLists.txt CMake 파일의 코드입니다.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS medical-imaging)

# Set this project's name.
project("hello_health-imaging")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
  for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
      "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
  endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
  # Copy relevant AWS SDK for C++ libraries into the current binary directory for
  running and debugging.

  # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you may
  need to uncomment this
  # and set the proper subdirectory to the executable location.

  AWSSDK_COPY_DYN_LIBS(SERVICE_COMPONENTS ""
    ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
  hello_health_imaging.cpp)
```

```
target_link_libraries(${PROJECT_NAME}
    ${AWSSDK_LINK_LIBRARIES})
```

## hello\_health\_imaging.cpp 소스 파일의 코드

```
#include <aws/core/Aws.h>
#include <aws/medical-imaging/MedicalImagingClient.h>
#include <aws/medical-imaging/model/ListDatastoresRequest.h>

#include <iostream>

/*
 * A "Hello HealthImaging" starter application which initializes an AWS
 * HealthImaging (HealthImaging) client
 * and lists the HealthImaging data stores in the current account.
 *
 * main function
 *
 * Usage: 'hello_health-imaging'
 */
#include <aws/core/auth/AWSCredentialsProviderChain.h>
#include <aws/core/platform/Environment.h>

int main(int argc, char **argv) {
    (void) argc;
    (void) argv;
    Aws::SDKOptions options;
    // Optional: change the log level for debugging.
    // options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;

    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::MedicalImaging::MedicalImagingClient
        medicalImagingClient(clientConfig);
        Aws::MedicalImaging::Model::ListDatastoresRequest listDatastoresRequest;
```

```

    Aws::Vector<Aws::MedicalImaging::Model::DatastoreSummary>
allDataStoreSummaries;
    Aws::String nextToken; // Used for paginated results.
    do {
        if (!nextToken.empty()) {
            listDatastoresRequest.SetNextToken(nextToken);
        }
        Aws::MedicalImaging::Model::ListDatastoresOutcome listDatastoresOutcome
=
            medicalImagingClient.ListDatastores(listDatastoresRequest);
        if (listDatastoresOutcome.IsSuccess()) {
            const Aws::Vector<Aws::MedicalImaging::Model::DatastoreSummary>
&dataStoreSummaries =
                listDatastoresOutcome.GetResult().GetDatastoreSummaries();
            allDataStoreSummaries.insert(allDataStoreSummaries.cend(),
                                         datastoreSummaries.cbegin(),
                                         datastoreSummaries.cend());
            nextToken = listDatastoresOutcome.GetResult().GetNextToken();
        }
        else {
            std::cerr << "ListDatastores error: "
                << listDatastoresOutcome.GetError().GetMessage() <<
std::endl;
            break;
        }
    } while (!nextToken.empty());

    std::cout << allDataStoreSummaries.size() << " HealthImaging data "
        << ((allDataStoreSummaries.size() == 1) ?
            "store was retrieved." : "stores were retrieved.") <<
std::endl;

    for (auto const &dataStoreSummary: allDataStoreSummaries) {
        std::cout << " Datastore: " << dataStoreSummary.GetDatastoreName()
            << std::endl;
        std::cout << " Datastore ID: " << dataStoreSummary.GetDatastoreId()
            << std::endl;
    }
}

Aws::ShutdownAPI(options); // Should only be called once.
return 0;
}

```

- API에 대한 세부 정보는 AWS SDK for C++ API 참조의 [ListDatastores](#)를 참조하세요.

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

## 작업

### DeleteImageSet

다음 코드 예시는 DeleteImageSet의 사용 방법을 보여줍니다.

SDK for C++

```

//! Routine which deletes an AWS HealthImaging image set.
/*!
    \param datastoreID: The HealthImaging data store ID.
    \param imageSetID: The image set ID.
    \param clientConfig: Aws client configuration.
    \return bool: Function succeeded.
    */
bool AwsDoc::Medical_Imaging::deleteImageSet(
    const Aws::String &dataStoreID, const Aws::String &imageSetID,
    const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::DeleteImageSetRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);
    Aws::MedicalImaging::Model::DeleteImageSetOutcome outcome =
    client.DeleteImageSet(
        request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted image set " << imageSetID
            << " from data store " << dataStoreID << std::endl;
    }
    else {
        std::cerr << "Error deleting image set " << imageSetID << " from data store
    "

```

```

        << datastoreID << ": " <<
        outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API에 대한 세부 정보는 AWS SDK for C++ API 참조의 [DeleteImageSet](#)를 참조하세요.

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

## GetDICOMImportJob

다음 코드 예시는 GetDICOMImportJob의 사용 방법을 보여줍니다.

SDK for C++

```

//! Routine which gets a HealthImaging DICOM import job's properties.
/*!
    \param datastoreID: The HealthImaging data store ID.
    \param importJobID: The DICOM import job ID
    \param clientConfig: Aws client configuration.
    \return GetDICOMImportJobOutcome: The import job outcome.
*/
Aws::MedicalImaging::Model::GetDICOMImportJobOutcome
AwsDoc::Medical_Imaging::getDICOMImportJob(const Aws::String &datastoreId,
                                             const Aws::String &importJobID,
                                             const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::GetDICOMImportJobRequest request;
    request.SetDatastoreId(datastoreId);
    request.SetJobId(importJobID);
    Aws::MedicalImaging::Model::GetDICOMImportJobOutcome outcome =
    client.GetDICOMImportJob(
        request);
    if (!outcome.IsSuccess()) {
        std::cerr << "GetDICOMImportJob error: "

```

```

        << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome;
}

```

- API에 대한 세부 정보는 AWS SDK for C++ API 참조의 [GetDICOMImportJob](#)을 참조하세요.

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

## GetImageFrame

다음 코드 예시는 GetImageFrame의 사용 방법을 보여줍니다.

SDK for C++

```

//! Routine which downloads an AWS HealthImaging image frame.
/*!
    \param datastoreID: The HealthImaging data store ID.
    \param imageSetID: The image set ID.
    \param frameID: The image frame ID.
    \param jphFile: File to store the downloaded frame.
    \param clientConfig: Aws client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::Medical_Imaging::getImageFrame(const Aws::String &dataStoreID,
                                             const Aws::String &imageSetID,
                                             const Aws::String &frameID,
                                             const Aws::String &jphFile,
                                             const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);

    Aws::MedicalImaging::Model::GetImageFrameRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);

    Aws::MedicalImaging::Model::ImageFrameInformation imageFrameInformation;

```

```

imageFrameInformation.SetImageFrameId(frameID);
request.SetImageFrameInformation(imageFrameInformation);

Aws::MedicalImaging::Model::GetImageFrameOutcome outcome = client.GetImageFrame(
    request);

if (outcome.IsSuccess()) {
    std::cout << "Successfully retrieved image frame." << std::endl;
    auto &buffer = outcome.GetResult().GetImageFrameBlob();

    std::ofstream outfile(jphFile, std::ios::binary);
    outfile << buffer.rdbuf();
}
else {
    std::cout << "Error retrieving image frame." <<
outcome.GetError().GetMessage()
    << std::endl;
}

return outcome.IsSuccess();
}

```

- API에 대한 세부 정보는 AWS SDK for C++ API 참조의 [GetImageFrame](#)을 참조하세요.

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

## GetImageSetMetadata

다음 코드 예시는 GetImageSetMetadata의 사용 방법을 보여줍니다.

SDK for C++

이미지 세트 메타데이터를 가져오는 유틸리티 함수입니다.

```

//! Routine which gets a HealthImaging image set's metadata.
/*!
    \param dataStoreID: The HealthImaging data store ID.

```

```

\param imageSetID: The HealthImaging image set ID.
\param versionID: The HealthImaging image set version ID, ignored if empty.
\param outputFilePath: The path where the metadata will be stored as gzipped json.
\param clientConfig: Aws client configuration.
\\return bool: Function succeeded.
*/
bool AwsDoc::Medical_Imaging::getImageSetMetadata(const Aws::String &dataStoreID,
                                                    const Aws::String &imageSetID,
                                                    const Aws::String &versionID,
                                                    const Aws::String &outputFilePath,
                                                    const
                                                    Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::Model::GetImageSetMetadataRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);
    if (!versionID.empty()) {
        request.SetVersionId(versionID);
    }
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::GetImageSetMetadataOutcome outcome =
    client.GetImageSetMetadata(
        request);
    if (outcome.IsSuccess()) {
        std::ofstream file(outputFilePath, std::ios::binary);
        auto &metadata = outcome.GetResult().GetImageSetMetadataBlob();
        file << metadata.rdbuf();
    }
    else {
        std::cerr << "Failed to get image set metadata: "
                    << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

버전 없이 이미지 세트 메타데이터를 가져옵니다.

```

    if (AwsDoc::Medical_Imaging::getImageSetMetadata(dataStoreID, imageSetID,
    "", outputFilePath, clientConfig))
    {
        std::cout << "Successfully retrieved image set metadata." << std::endl;
        std::cout << "Metadata stored in: " << outputFilePath << std::endl;
    }

```

```
}
```

버전과 함께 이미지 세트 메타데이터를 가져옵니다.

```
if (AwsDoc::Medical_Imaging::getImageSetMetadata(dataStoreID, imageSetID,
versionID, outputFilePath, clientConfig))
{
    std::cout << "Successfully retrieved image set metadata." << std::endl;
    std::cout << "Metadata stored in: " << outputFilePath << std::endl;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetImageSetMetadata](#)를 참조하세요.

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

## SearchImageSets

다음 코드 예시는 SearchImageSets의 사용 방법을 보여줍니다.

SDK for C++

이미지 세트 검색을 위한 유틸리티 함수.

```
//! Routine which searches for image sets based on defined input attributes.
/*!
    \param dataStoreID: The HealthImaging data store ID.
    \param searchCriteria: A search criteria instance.
    \param imageSetResults: Vector to receive the image set IDs.
    \param clientConfig: Aws client configuration.
    \return bool: Function succeeded.
    */
bool AwsDoc::Medical_Imaging::searchImageSets(const Aws::String &dataStoreID,
                                              const
                                              Aws::MedicalImaging::Model::SearchCriteria &searchCriteria,
                                              Aws::Vector<Aws::String>
                                              &imageSetResults,
```

```

const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::SearchImageSetsRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetSearchCriteria(searchCriteria);

    Aws::String nextToken; // Used for paginated results.
    bool result = true;
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::MedicalImaging::Model::SearchImageSetsOutcome outcome =
client.SearchImageSets(
    request);
        if (outcome.IsSuccess()) {
            for (auto &imageSetMetadataSummary:
outcome.GetResult().GetImageSetsMetadataSummaries()) {
                imageSetResults.push_back(imageSetMetadataSummary.GetImageSetId());
            }

            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cout << "Error: " << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
    } while (!nextToken.empty());

    return result;
}

```

### 사용 사례 #1: EQUAL 연산자.

```

    Aws::Vector<Aws::String> imageIDsForPatientID;
    Aws::MedicalImaging::Model::SearchCriteria searchCriteriaEqualsPatientID;
    Aws::Vector<Aws::MedicalImaging::Model::SearchFilter> patientIDSearchFilters
= {

    Aws::MedicalImaging::Model::SearchFilter().WithOperator(Aws::MedicalImaging::Model::Operator

```

```

.WithValues({Aws::MedicalImaging::Model::SearchByAttributeValue().WithDICOMPatientId(patientID)});

searchCriteriaEqualsPatientID.SetFilters(patientIDSearchFilters);
bool result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,

searchCriteriaEqualsPatientID,

imageIDsForPatientID,
clientConfig);

if (result) {
    std::cout << imageIDsForPatientID.size() << " image sets found for the
patient with ID '"
    << patientID << "'." << std::endl;
    for (auto &imageSetResult : imageIDsForPatientID) {
        std::cout << " Image set with ID '" << imageSetResult << std::endl;
    }
}

```

사용 사례 #2: DICOMStudyDate 및 DICOMStudyTime을 사용한 BETWEEN 연산자.

```

    Aws::MedicalImaging::Model::SearchByAttributeValue useCase2StartDate;

useCase2StartDate.SetDICOMStudyDateAndTime(Aws::MedicalImaging::Model::DICOMStudyDateAndTime(
    .WithDICOMStudyDate("19990101")
    .WithDICOMStudyTime("000000.000")));

    Aws::MedicalImaging::Model::SearchByAttributeValue useCase2EndDate;

useCase2EndDate.SetDICOMStudyDateAndTime(Aws::MedicalImaging::Model::DICOMStudyDateAndTime(
    .WithDICOMStudyDate(Aws::Utils::DateTime(std::chrono::system_clock::now()).ToLocalTimeStrin
"%m%d"))
    .WithDICOMStudyTime("000000.000"));

    Aws::MedicalImaging::Model::SearchFilter useCase2SearchFilter;
    useCase2SearchFilter.SetValues({useCase2StartDate, useCase2EndDate});

    useCase2SearchFilter.SetOperator(Aws::MedicalImaging::Model::Operator::BETWEEN);

    Aws::MedicalImaging::Model::SearchCriteria useCase2SearchCriteria;

```

```

useCase2SearchCriteria.SetFilters({useCase2SearchFilter});

Aws::Vector<Aws::String> usesCase2Results;
result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,
                                                    useCase2SearchCriteria,
                                                    usesCase2Results,
                                                    clientConfig);

if (result) {
    std::cout << usesCase2Results.size() << " image sets found for between
1999/01/01 and present."
               << std::endl;
    for (auto &imageSetResult : usesCase2Results) {
        std::cout << " Image set with ID '" << imageSetResult << std::endl;
    }
}

```

사용 사례 #3: createdAt을 사용한 BETWEEN 연산자. 시간 연구가 이전에 지속되었습니다.

```

Aws::MedicalImaging::Model::SearchByAttributeValue useCase3StartDate;
useCase3StartDate.SetCreatedAt(Aws::Utils::DateTime("20231130T000000000Z", Aws::Utils::DateF

Aws::MedicalImaging::Model::SearchByAttributeValue useCase3EndDate;
useCase3EndDate.SetCreatedAt(Aws::Utils::DateTime(std::chrono::system_clock::now()));

Aws::MedicalImaging::Model::SearchFilter useCase3SearchFilter;
useCase3SearchFilter.SetValues({useCase3StartDate, useCase3EndDate});

useCase3SearchFilter.SetOperator(Aws::MedicalImaging::Model::Operator::BETWEEN);

Aws::MedicalImaging::Model::SearchCriteria useCase3SearchCriteria;
useCase3SearchCriteria.SetFilters({useCase3SearchFilter});

Aws::Vector<Aws::String> usesCase3Results;
result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,
                                                    useCase3SearchCriteria,
                                                    usesCase3Results,
                                                    clientConfig);

if (result) {
    std::cout << usesCase3Results.size() << " image sets found for created
between 2023/11/30 and present."

```

```

        << std::endl;
    for (auto &imageSetResult : usesCase3Results) {
        std::cout << "    Image set with ID '" << imageSetResult << std::endl;
    }
}

```

사용 사례 4: DICOMSeriesInstanceUID 필드에는 EQUAL 연산자를 적용하고 updatedAt 필드에는 BETWEEN 연산자를 적용하며, 응답은 updatedAt 필드를 기준으로 오름차순(ASC) 정렬합니다.

```

    Aws::MedicalImaging::Model::SearchByAttributeValue useCase4StartDate;

    useCase4StartDate.SetUpdatedAt(Aws::Utils::DateTime("20231130T000000000Z", Aws::Utils::DateF

    Aws::MedicalImaging::Model::SearchByAttributeValue useCase4EndDate;

    useCase4EndDate.SetUpdatedAt(Aws::Utils::DateTime(std::chrono::system_clock::now()));

    Aws::MedicalImaging::Model::SearchFilter useCase4SearchFilterBetween;
    useCase4SearchFilterBetween.SetValues({useCase4StartDate, useCase4EndDate});

    useCase4SearchFilterBetween.SetOperator(Aws::MedicalImaging::Model::Operator::BETWEEN);

    Aws::MedicalImaging::Model::SearchByAttributeValue seriesInstanceUID;
    seriesInstanceUID.SetDICOMSeriesInstanceUID(dicomSeriesInstanceUID);

    Aws::MedicalImaging::Model::SearchFilter useCase4SearchFilterEqual;
    useCase4SearchFilterEqual.SetValues({seriesInstanceUID});

    useCase4SearchFilterEqual.SetOperator(Aws::MedicalImaging::Model::Operator::EQUAL);

    Aws::MedicalImaging::Model::SearchCriteria useCase4SearchCriteria;
    useCase4SearchCriteria.SetFilters({useCase4SearchFilterBetween,
    useCase4SearchFilterEqual});

    Aws::MedicalImaging::Model::Sort useCase4Sort;
    useCase4Sort.SetSortField(Aws::MedicalImaging::Model::SortField::updatedAt);
    useCase4Sort.SetSortOrder(Aws::MedicalImaging::Model::SortOrder::ASC);

    useCase4SearchCriteria.SetSort(useCase4Sort);

    Aws::Vector<Aws::String> usesCase4Results;
    result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,

```

```

        useCase4SearchCriteria,
        usesCase4Results,
        clientConfig);

    if (result) {
        std::cout << usesCase4Results.size() << " image sets found for EQUAL
operator "
        << "on DICOMSeriesInstanceUID and BETWEEN on updatedAt and sort response
\n"
        << "in ASC order on updatedAt field." << std::endl;
        for (auto &imageSetResult : usesCase4Results) {
            std::cout << " Image set with ID '" << imageSetResult << std::endl;
        }
    }
}

```

- API에 대한 세부 정보는 AWS SDK for C++ API 참조의 [SearchImageSets](#)를 참조하세요.

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

## StartDICOMImportJob

다음 코드 예시는 StartDICOMImportJob의 사용 방법을 보여줍니다.

SDK for C++

```

//! Routine which starts a HealthImaging import job.
/*!
    \param dataStoreID: The HealthImaging data store ID.
    \param inputBucketName: The name of the Amazon S3 bucket containing the DICOM
files.
    \param inputDirectory: The directory in the S3 bucket containing the DICOM files.
    \param outputBucketName: The name of the S3 bucket for the output.
    \param outputDirectory: The directory in the S3 bucket to store the output.
    \param roleArn: The ARN of the IAM role with permissions for the import.
    \param importJobId: A string to receive the import job ID.
    \param clientConfig: Aws client configuration.
    \return bool: Function succeeded.
*/

```

```

bool AwsDoc::Medical_Imaging::startDICOMImportJob(
    const Aws::String &dataStoreID, const Aws::String &inputBucketName,
    const Aws::String &inputDirectory, const Aws::String &outputBucketName,
    const Aws::String &outputDirectory, const Aws::String &roleArn,
    Aws::String &importJobId,
    const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient medicalImagingClient(clientConfig);
    Aws::String inputURI = "s3://" + inputBucketName + "/" + inputDirectory + "/";
    Aws::String outputURI = "s3://" + outputBucketName + "/" + outputDirectory +
        "/";
    Aws::MedicalImaging::Model::StartDICOMImportJobRequest
startDICOMImportJobRequest;
    startDICOMImportJobRequest.SetDatastoreId(dataStoreID);
    startDICOMImportJobRequest.SetDataAccessRoleArn(roleArn);
    startDICOMImportJobRequest.SetInputS3Uri(inputURI);
    startDICOMImportJobRequest.SetOutputS3Uri(outputURI);

    Aws::MedicalImaging::Model::StartDICOMImportJobOutcome
startDICOMImportJobOutcome = medicalImagingClient.StartDICOMImportJob(
    startDICOMImportJobRequest);

    if (startDICOMImportJobOutcome.IsSuccess()) {
        importJobId = startDICOMImportJobOutcome.GetResult().GetJobId();
    }
    else {
        std::cerr << "Failed to start DICOM import job because "
            << startDICOMImportJobOutcome.GetError().GetMessage() <<
std::endl;
    }

    return startDICOMImportJobOutcome.IsSuccess();
}

```

- API에 대한 세부 정보는 AWS SDK for C++ API 참조의 [StartDICOMImportJob](#)을 참조하세요.

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

## 시나리오

### 이미지 세트 및 이미지 프레임 시작하기

다음 코드 예제에서는 HealthImaging에서 DICOM 파일을 가져오고 이미지 프레임을 다운로드하는 방법을 보여줍니다.

구현은 명령줄 애플리케이션으로 구성됩니다.

- DICOM 가져오기의 리소스를 설정합니다.
- 데이터 스토어로 DICOM 파일을 가져옵니다.
- 가져오기 작업의 이미지 세트 ID를 검색합니다.
- 이미지 세트의 이미지 프레임 ID를 검색합니다.
- 이미지 프레임을 다운로드, 디코딩 및 확인합니다.
- 리소스를 정리합니다.

### SDK for C++

필요한 리소스를 사용하여 CloudFormation 스택을 생성합니다.

```
Aws::String inputBucketName;
Aws::String outputBucketName;
Aws::String dataStoreId;
Aws::String roleArn;
Aws::String stackName;

if (askYesNoQuestion(
    "Would you like to let this workflow create the resources for you? (y/n)
")) {
    stackName = askQuestion(
        "Enter a name for the AWS CloudFormation stack to create. ");
    Aws::String dataStoreName = askQuestion(
        "Enter a name for the HealthImaging datastore to create. ");

    Aws::Map<Aws::String, Aws::String> outputs = createCloudFormationStack(
        stackName,
        dataStoreName,
        clientConfiguration);
```

```

        if (!retrieveOutputs(outputs, dataStoreId, inputBucketName,
            outputBucketName,
                                roleArn)) {
            return false;
        }

        std::cout << "The following resources have been created." << std::endl;
        std::cout << "A HealthImaging datastore with ID: " << dataStoreId << "."
            << std::endl;
        std::cout << "An Amazon S3 input bucket named: " << inputBucketName << "."
            << std::endl;
        std::cout << "An Amazon S3 output bucket named: " << outputBucketName << "."
            << std::endl;
        std::cout << "An IAM role with the ARN: " << roleArn << "." << std::endl;
        askQuestion("Enter return to continue.", alwaysTrueTest);
    }
    else {
        std::cout << "You have chosen to use preexisting resources:" << std::endl;
        dataStoreId = askQuestion(
            "Enter the data store ID of the HealthImaging datastore you wish to
use: ");
        inputBucketName = askQuestion(
            "Enter the name of the S3 input bucket you wish to use: ");
        outputBucketName = askQuestion(
            "Enter the name of the S3 output bucket you wish to use: ");
        roleArn = askQuestion(
            "Enter the ARN for the IAM role with the proper permissions to
import a DICOM series: ");
    }
}

```

Amazon S3 가져오기 버킷에 DICOM 파일을 복사합니다.

```

std::cout
    << "This workflow uses DICOM files from the National Cancer Institute
Imaging Data\n"
    << "Commons (IDC) Collections." << std::endl;
std::cout << "Here is the link to their website." << std::endl;
std::cout << "https://registry.opendata.aws/nci-imaging-data-commons/" <<
std::endl;
std::cout << "We will use DICOM files stored in an S3 bucket managed by the
IDC."
    << std::endl;

```

```

std::cout
    << "First one of the DICOM folders in the IDC collection must be copied
to your\n"
    "input S3 bucket."
    << std::endl;
std::cout << "You have the choice of one of the following "
    << IDC_ImageChoices.size() << " folders to copy." << std::endl;

int index = 1;
for (auto &idcChoice: IDC_ImageChoices) {
    std::cout << index << " - " << idcChoice.mDescription << std::endl;
    index++;
}
int choice = askQuestionForIntRange("Choose DICOM files to import: ", 1, 4);

Aws::String fromDirectory = IDC_ImageChoices[choice - 1].mDirectory;
Aws::String inputDirectory = "input";

std::cout << "The files in the directory '" << fromDirectory << "' in the bucket
'"
    << IDC_S3_BucketName << "' will be copied " << std::endl;
std::cout << "to the folder '" << inputDirectory << "/" << fromDirectory
    << "' in the bucket '" << inputBucketName << "'." << std::endl;
askQuestion("Enter return to start the copy.", alwaysTrueTest);

if (!AwsDoc::Medical_Imaging::copySeriesBetweenBuckets(
    IDC_S3_BucketName,
    fromDirectory,
    inputBucketName,
    inputDirectory, clientConfiguration)) {
    std::cerr << "This workflow will exit because of an error." << std::endl;
    cleanup(stackName, dataStoreId, clientConfiguration);
    return false;
}

```

Amazon S3 데이터 스토어로 DICOM 파일을 가져옵니다.

```

bool AwsDoc::Medical_Imaging::startDicomImport(const Aws::String &dataStoreID,
                                                const Aws::String &inputBucketName,
                                                const Aws::String &inputDirectory,
                                                const Aws::String &outputBucketName,
                                                const Aws::String &outputDirectory,

```

```

        const Aws::String &roleArn,
        Aws::String &importJobId,
        const
    Aws::Client::ClientConfiguration &clientConfiguration) {
        bool result = false;
        if (startDICOMImportJob(dataStoreID, inputBucketName, inputDirectory,
                                outputBucketName, outputDirectory, roleArn, importJobId,
                                clientConfiguration)) {
            std::cout << "DICOM import job started with job ID " << importJobId << "."
                << std::endl;
            result = waitImportJobCompleted(dataStoreID, importJobId,
            clientConfiguration);
            if (result) {
                std::cout << "DICOM import job completed." << std::endl;
            }
        }

        return result;
    }

    //! Routine which starts a HealthImaging import job.
    /*!
    \param datastoreID: The HealthImaging data store ID.
    \param inputBucketName: The name of the Amazon S3 bucket containing the DICOM
    files.
    \param inputDirectory: The directory in the S3 bucket containing the DICOM files.
    \param outputBucketName: The name of the S3 bucket for the output.
    \param outputDirectory: The directory in the S3 bucket to store the output.
    \param roleArn: The ARN of the IAM role with permissions for the import.
    \param importJobId: A string to receive the import job ID.
    \param clientConfig: Aws client configuration.
    \return bool: Function succeeded.
    */
    bool AwsDoc::Medical_Imaging::startDICOMImportJob(
        const Aws::String &dataStoreID, const Aws::String &inputBucketName,
        const Aws::String &inputDirectory, const Aws::String &outputBucketName,
        const Aws::String &outputDirectory, const Aws::String &roleArn,
        Aws::String &importJobId,
        const Aws::Client::ClientConfiguration &clientConfig) {
        Aws::MedicalImaging::MedicalImagingClient medicalImagingClient(clientConfig);
        Aws::String inputURI = "s3://" + inputBucketName + "/" + inputDirectory + "/";
        Aws::String outputURI = "s3://" + outputBucketName + "/" + outputDirectory +
        "/";

```

```

    Aws::MedicalImaging::Model::StartDICOMImportJobRequest
startDICOMImportJobRequest;
    startDICOMImportJobRequest.SetDatastoreId(dataStoreID);
    startDICOMImportJobRequest.SetDataAccessRoleArn(roleArn);
    startDICOMImportJobRequest.SetInputS3Uri(inputURI);
    startDICOMImportJobRequest.SetOutputS3Uri(outputURI);

    Aws::MedicalImaging::Model::StartDICOMImportJobOutcome
startDICOMImportJobOutcome = medicalImagingClient.StartDICOMImportJob(
    startDICOMImportJobRequest);

    if (startDICOMImportJobOutcome.IsSuccess()) {
        importJobId = startDICOMImportJobOutcome.GetResult().GetJobId();
    }
    else {
        std::cerr << "Failed to start DICOM import job because "
        << startDICOMImportJobOutcome.GetError().GetMessage() <<
std::endl;
    }

    return startDICOMImportJobOutcome.IsSuccess();
}

//! Routine which waits for a DICOM import job to complete.
/*!
 * @param datastoreID: The HealthImaging data store ID.
 * @param importJobId: The import job ID.
 * @param clientConfiguration : Aws client configuration.
 * @return bool: Function succeeded.
 */
bool AwsDoc::Medical_Imaging::waitImportJobCompleted(const Aws::String &datastoreID,
                                                    const Aws::String &importJobId,
                                                    const
    Aws::Client::ClientConfiguration &clientConfiguration) {

    Aws::MedicalImaging::Model::JobStatus jobStatus =
    Aws::MedicalImaging::Model::JobStatus::IN_PROGRESS;
    while (jobStatus == Aws::MedicalImaging::Model::JobStatus::IN_PROGRESS) {
        std::this_thread::sleep_for(std::chrono::seconds(1));

        Aws::MedicalImaging::Model::GetDICOMImportJobOutcome
getDicomImportJobOutcome = getDICOMImportJob(
    datastoreID, importJobId,

```

```

        clientConfiguration);

    if (getDicomImportJobOutcome.IsSuccess()) {
        jobStatus =
getDicomImportJobOutcome.GetResult().GetJobProperties().GetJobStatus();

        std::cout << "DICOM import job status: " <<

Aws::MedicalImaging::Model::JobStatusMapper::GetNameForJobStatus(
        jobStatus) << std::endl;
    }
    else {
        std::cerr << "Failed to get import job status because "
        << getDicomImportJobOutcome.GetError().GetMessage() <<
std::endl;
        return false;
    }
}

return jobStatus == Aws::MedicalImaging::Model::JobStatus::COMPLETED;
}

//! Routine which gets a HealthImaging DICOM import job's properties.
/*!
\param dataStoreID: The HealthImaging data store ID.
\param importJobID: The DICOM import job ID
\param clientConfig: Aws client configuration.
\return GetDICOMImportJobOutcome: The import job outcome.
*/
Aws::MedicalImaging::Model::GetDICOMImportJobOutcome
AwsDoc::Medical_Imaging::getDICOMImportJob(const Aws::String &dataStoreID,
                                           const Aws::String &importJobID,
                                           const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::GetDICOMImportJobRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetJobId(importJobID);
    Aws::MedicalImaging::Model::GetDICOMImportJobOutcome outcome =
client.GetDICOMImportJob(
    request);
    if (!outcome.IsSuccess()) {
        std::cerr << "GetDICOMImportJob error: "
        << outcome.GetError().GetMessage() << std::endl;
    }
}

```

```

    }

    return outcome;
}

```

DICOM 가져오기 작업으로 생성된 이미지 세트를 가져옵니다.

```

bool
AwsDoc::Medical_Imaging::getImageSetsForDicomImportJob(const Aws::String
&datastoreId,
                                                         const Aws::String
&importJobId,
                                                         Aws::Vector<Aws::String>
&imageSets,
                                                         const
Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::MedicalImaging::Model::GetDICOMImportJobOutcome getDicomImportJobOutcome =
getDICOMImportJob(
    datastoreID, importJobId, clientConfiguration);
    bool result = false;
    if (getDicomImportJobOutcome.IsSuccess()) {
        auto outputURI =
getDicomImportJobOutcome.GetResult().GetJobProperties().GetOutputS3Uri();
        Aws::Http::URI uri(outputURI);
        const Aws::String &bucket = uri.GetAuthority();
        Aws::String key = uri.GetPath();

        Aws::S3::S3Client s3Client(clientConfiguration);
        Aws::S3::Model::GetObjectRequest objectRequest;
        objectRequest.SetBucket(bucket);
        objectRequest.SetKey(key + "/" + IMPORT_JOB_MANIFEST_FILE_NAME);

        auto getObjectOutcome = s3Client.GetObject(objectRequest);
        if (getObjectOutcome.IsSuccess()) {
            auto &data = getObjectOutcome.GetResult().GetBody();

            std::stringstream stringStream;
            stringStream << data.rdbuf();

            try {
                // Use JMESPath to extract the image set IDs.
                // https://jmespath.org/specification.html

```

```

        std::string jmesPathExpression =
"jobSummary.imageSetsSummary[].imageSetId";
        jsoncons::json doc = jsoncons::json::parse(stringStream.str());

        jsoncons::json imageSetsJson = jsoncons::jmespath::search(doc,
jmesPathExpression);\
        for (auto &imageSet: imageSetsJson.array_range()) {
            imageSets.push_back(imageSet.as_string());
        }

        result = true;
    }
    catch (const std::exception &e) {
        std::cerr << e.what() << '\n';
    }

}
else {
    std::cerr << "Failed to get object because "
        << getObjectOutcome.GetError().GetMessage() << std::endl;
}

}
else {
    std::cerr << "Failed to get import job status because "
        << getDicomImportJobOutcome.GetError().GetMessage() << std::endl;
}

return result;
}

```

이미지 세트의 이미지 프레임 정보를 가져옵니다.

```

bool AwsDoc::Medical_Imaging::getImageFramesForImageSet(const Aws::String
&dataStoreID,
                                                         const Aws::String
&imageSetID,
                                                         const Aws::String
&outDirectory,
                                                         Aws::Vector<ImageFrameInfo>
&imageFrames,

```

```

const
Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::String fileName = outDirectory + "/" + imageSetID + "_metadata.json.gzip";
    bool result = false;
    if (getImageSetMetadata(dataStoreID, imageSetID, "", // Empty string for version
ID.
                                fileName, clientConfiguration)) {
        try {
            std::string metadataGZip;
            {
                std::ifstream inFileStream(fileName.c_str(), std::ios::binary);
                if (!inFileStream) {
                    throw std::runtime_error("Failed to open file " + fileName);
                }

                std::stringstream stringStream;
                stringStream << inFileStream.rdbuf();
                metadataGZip = stringStream.str();
            }
            std::string metadataJson = gzip::decompress(metadataGZip.data(),
                                                        metadataGZip.size());

            // Use JMESPath to extract the image set IDs.
            // https://jmespath.org/specification.html
            jsoncons::json doc = jsoncons::json::parse(metadataJson);
            std::string jmesPathExpression = "Study.Series.*.Instances[].[*]";
            jsoncons::json instances = jsoncons::jmespath::search(doc,
jmesPathExpression);
            for (auto &instance: instances.array_range()) {
                jmesPathExpression = "DICOM.RescaleSlope";
                std::string rescaleSlope = jsoncons::jmespath::search(instance,
jmesPathExpression).to_string();
                jmesPathExpression = "DICOM.RescaleIntercept";
                std::string rescaleIntercept = jsoncons::jmespath::search(instance,
jmesPathExpression).to_string();

                jmesPathExpression = "ImageFrames[].[*]";
                jsoncons::json imageFramesJson =
jsoncons::jmespath::search(instance,
jmesPathExpression);

```

```

        for (auto &imageFrame: imageFramesJson.array_range()) {
            ImageFrameInfo imageFrameIDs;
            imageFrameIDs.mImageSetId = imageSetID;
            imageFrameIDs.mImageFrameId = imageFrame.find(
                "ID")->value().as_string();
            imageFrameIDs.mRescaleIntercept = rescaleIntercept;
            imageFrameIDs.mRescaleSlope = rescaleSlope;
            imageFrameIDs.MinPixelValue = imageFrame.find(
                "MinPixelValue")->value().as_string();
            imageFrameIDs.MaxPixelValue = imageFrame.find(
                "MaxPixelValue")->value().as_string();

            jmesPathExpression =
                "max_by(PixelDataChecksumFromBaseToFullResolution, &Width).Checksum";
            jsoncons::json checksumJson =
                jsoncons::jmespath::search(imageFrame,

                jmesPathExpression);
            imageFrameIDs.mFullResolutionChecksum =
                checksumJson.as_integer<uint32_t>();

            imageFrames.emplace_back(imageFrameIDs);
        }
    }

    result = true;
}
catch (const std::exception &e) {
    std::cerr << "getImageFramesForImageSet failed because " << e.what()
        << std::endl;
}

return result;
}

//! Routine which gets a HealthImaging image set's metadata.
/*!
    \param datastoreID: The HealthImaging data store ID.
    \param imageSetID: The HealthImaging image set ID.
    \param versionID: The HealthImaging image set version ID, ignored if empty.
    \param outputPath: The path where the metadata will be stored as gzipped json.
    \param clientConfig: Aws client configuration.
    \return bool: Function succeeded.

```

```

*/
bool AwsDoc::Medical_Imaging::getImageSetMetadata(const Aws::String &dataStoreID,
                                                    const Aws::String &imageSetID,
                                                    const Aws::String &versionID,
                                                    const Aws::String &outputFilePath,
                                                    const
                                                    Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::Model::GetImageSetMetadataRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);
    if (!versionID.empty()) {
        request.SetVersionId(versionID);
    }
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::GetImageSetMetadataOutcome outcome =
    client.GetImageSetMetadata(
        request);
    if (outcome.IsSuccess()) {
        std::ofstream file(outputFilePath, std::ios::binary);
        auto &metadata = outcome.GetResult().GetImageSetMetadataBlob();
        file << metadata.rdbuf();
    }
    else {
        std::cerr << "Failed to get image set metadata: "
                   << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

이미지 프레임을 다운로드하고, 디코딩하고, 확인합니다.

```

bool AwsDoc::Medical_Imaging::downloadDecodeAndCheckImageFrames(
    const Aws::String &dataStoreID,
    const Aws::Vector<ImageFrameInfo> &imageFrames,
    const Aws::String &outDirectory,
    const Aws::Client::ClientConfiguration &clientConfiguration) {

    Aws::Client::ClientConfiguration clientConfiguration1(clientConfiguration);
    clientConfiguration1.executor =
    Aws::MakeShared<Aws::Utils::Threading::PooledThreadExecutor>(
        "executor", 25);
}

```

```

    Aws::MedicalImaging::MedicalImagingClient medicalImagingClient(
        clientConfiguration1);

    Aws::Utils::Threading::Semaphore semaphore(0, 1);
    std::atomic<size_t> count(imageFrames.size());

    bool result = true;
    for (auto &imageFrame: imageFrames) {
        Aws::MedicalImaging::Model::GetImageFrameRequest getImageFrameRequest;
        getImageFrameRequest.SetDatastoreId(dataStoreID);
        getImageFrameRequest.SetImageSetId(imageFrame.mImageSetId);

        Aws::MedicalImaging::Model::ImageFrameInformation imageFrameInformation;
        imageFrameInformation.SetImageFrameId(imageFrame.mImageFrameId);
        getImageFrameRequest.SetImageFrameInformation(imageFrameInformation);

        auto getImageFrameAsyncLambda = [&semaphore, &result, &count, imageFrame,
outDirectory](
            const Aws::MedicalImaging::MedicalImagingClient *client,
            const Aws::MedicalImaging::Model::GetImageFrameRequest &request,
            Aws::MedicalImaging::Model::GetImageFrameOutcome outcome,
            const std::shared_ptr<const Aws::Client::AsyncCallerContext>
&context) {

            if (!handleGetImageFrameResult(outcome, outDirectory, imageFrame)) {
                std::cerr << "Failed to download and convert image frame: "
                    << imageFrame.mImageFrameId << " from image set: "
                    << imageFrame.mImageSetId << std::endl;
                result = false;
            }

            count--;
            if (count <= 0) {

                semaphore.ReleaseAll();
            }
        }; // End of 'getImageFrameAsyncLambda' lambda.

        medicalImagingClient.GetImageFrameAsync(getImageFrameRequest,
                                                getImageFrameAsyncLambda);
    }

    if (count > 0) {
        semaphore.WaitOne();
    }

```

```

    }

    if (result) {
        std::cout << imageFrames.size() << " image files were downloaded."
            << std::endl;
    }

    return result;
}

bool AwsDoc::Medical_Imaging::decodeJPHFileAndValidateWithChecksum(
    const Aws::String &jphFile,
    uint32_t crc32Checksum) {
    opj_image_t *outputImage = jphImageToOpjBitmap(jphFile);
    if (!outputImage) {
        return false;
    }

    bool result = true;
    if (!verifyChecksumForImage(outputImage, crc32Checksum)) {
        std::cerr << "The checksum for the image does not match the expected value."
            << std::endl;
        std::cerr << "File :" << jphFile << std::endl;
        result = false;
    }

    opj_image_destroy(outputImage);

    return result;
}

opj_image *
AwsDoc::Medical_Imaging::jphImageToOpjBitmap(const Aws::String &jphFile) {
    opj_stream_t *inFileStream = nullptr;
    opj_codec_t *decompressorCodec = nullptr;
    opj_image_t *outputImage = nullptr;
    try {
        std::shared_ptr<opj_dparameters> decodeParameters =
std::make_shared<opj_dparameters>();
        memset(decodeParameters.get(), 0, sizeof(opj_dparameters));

        opj_set_default_decoder_parameters(decodeParameters.get());

        decodeParameters->decod_format = 1; // JP2 image format.
    }

```

```

decodeParameters->cod_format = 2; // BMP image format.

std::strncpy(decodeParameters->infile, jphFile.c_str(),
             OPJ_PATH_LEN);

inFileStream = opj_stream_create_default_file_stream(
    decodeParameters->infile, true);
if (!inFileStream) {
    throw std::runtime_error(
        "Unable to create input file stream for file '" + jphFile +
        "'.");
}

decompressorCodec = opj_create_decompress(OPJ_CODEC_JP2);
if (!decompressorCodec) {
    throw std::runtime_error("Failed to create decompression codec.");
}

int decodeMessageLevel = 1;
if (!setupCodecLogging(decompressorCodec, &decodeMessageLevel)) {
    std::cerr << "Failed to setup codec logging." << std::endl;
}

if (!opj_setup_decoder(decompressorCodec, decodeParameters.get())) {
    throw std::runtime_error("Failed to setup decompression codec.");
}
if (!opj_codec_set_threads(decompressorCodec, 4)) {
    throw std::runtime_error("Failed to set decompression codec threads.");
}

if (!opj_read_header(inFileStream, decompressorCodec, &outputImage)) {
    throw std::runtime_error("Failed to read header.");
}

if (!opj_decode(decompressorCodec, inFileStream,
                outputImage)) {
    throw std::runtime_error("Failed to decode.");
}

if (DEBUGGING) {
    std::cout << "image width : " << outputImage->x1 - outputImage->x0
              << std::endl;
    std::cout << "image height : " << outputImage->y1 - outputImage->y0
              << std::endl;
}

```

```

        std::cout << "number of channels: " << outputImage->numcomps
                    << std::endl;
        std::cout << "colorspace : " << outputImage->color_space << std::endl;
    }

    } catch (const std::exception &e) {
        std::cerr << e.what() << std::endl;
        if (outputImage) {
            opj_image_destroy(outputImage);
            outputImage = nullptr;
        }
    }
    if (inFileStream) {
        opj_stream_destroy(inFileStream);
    }
    if (decompressorCodec) {
        opj_destroy_codec(decompressorCodec);
    }

    return outputImage;
}

//! Template function which converts a planar image bitmap to an interleaved image
    bitmap and
    //! then verifies the checksum of the bitmap.
    /*!
    * @param image: The OpenJPEG image struct.
    * @param crc32Checksum: The CRC32 checksum.
    * @return bool: Function succeeded.
    */
template<class myType>
bool verifyChecksumForImageForType(opj_image_t *image, uint32_t crc32Checksum) {
    uint32_t width = image->x1 - image->x0;
    uint32_t height = image->y1 - image->y0;
    uint32_t numOfChannels = image->numcomps;

    // Buffer for interleaved bitmap.
    std::vector<myType> buffer(width * height * numOfChannels);

    // Convert planar bitmap to interleaved bitmap.
    for (uint32_t channel = 0; channel < numOfChannels; channel++) {
        for (uint32_t row = 0; row < height; row++) {
            uint32_t fromRowStart = row / image->comps[channel].dy * width /
                                    image->comps[channel].dx;

```

```

        uint32_t toIndex = (row * width) * numOfChannels + channel;

        for (uint32_t col = 0; col < width; col++) {
            uint32_t fromIndex = fromRowStart + col / image->comps[channel].dx;

            buffer[toIndex] = static_cast<myType>(image-
>comps[channel].data[fromIndex]);

            toIndex += numOfChannels;
        }
    }
}

// Verify checksum.
boost::crc_32_type crc32;
crc32.process_bytes(reinterpret_cast<char *>(buffer.data()),
                    buffer.size() * sizeof(myType));

bool result = crc32.checksum() == crc32Checksum;
if (!result) {
    std::cerr << "verifyChecksumForImage, checksum mismatch, expected - "
                << crc32Checksum << ", actual - " << crc32.checksum()
                << std::endl;
}

return result;
}

//! Routine which verifies the checksum of an OpenJPEG image struct.
/*!
 * @param image: The OpenJPEG image struct.
 * @param crc32Checksum: The CRC32 checksum.
 * @return bool: Function succeeded.
 */
bool AwsDoc::Medical_Imaging::verifyChecksumForImage(opj_image_t *image,
                                                       uint32_t crc32Checksum) {
    uint32_t channels = image->numcomps;
    bool result = false;
    if (0 < channels) {
        // Assume the precision is the same for all channels.
        uint32_t precision = image->comps[0].prec;
        bool signedData = image->comps[0].sgnd;
        uint32_t bytes = (precision + 7) / 8;
    }
}

```

```

        if (signedData) {
            switch (bytes) {
                case 1 :
                    result = verifyChecksumForImageForType<int8_t>(image,
                                                                    crc32Checksum);

                    break;
                case 2 :
                    result = verifyChecksumForImageForType<int16_t>(image,
                                                                    crc32Checksum);

                    break;
                case 4 :
                    result = verifyChecksumForImageForType<int32_t>(image,
                                                                    crc32Checksum);

                    break;
                default:
                    std::cerr
                        << "verifyChecksumForImage, unsupported data type,
signed bytes - "
                        << bytes << std::endl;
                    break;
            }
        }
        else {
            switch (bytes) {
                case 1 :
                    result = verifyChecksumForImageForType<uint8_t>(image,
                                                                    crc32Checksum);

                    break;
                case 2 :
                    result = verifyChecksumForImageForType<uint16_t>(image,
                                                                    crc32Checksum);

                    break;
                case 4 :
                    result = verifyChecksumForImageForType<uint32_t>(image,
                                                                    crc32Checksum);

                    break;
                default:
                    std::cerr
                        << "verifyChecksumForImage, unsupported data type,
unsigned bytes - "
                        << bytes << std::endl;
                    break;
            }
        }
    }
}

```

```

        if (!result) {
            std::cerr << "verifyChecksumForImage, error bytes " << bytes
                << " signed "
                << signedData << std::endl;
        }
    }
    else {
        std::cerr << "'verifyChecksumForImage', no channels in the image."
            << std::endl;
    }
    return result;
}

```

리소스를 정리합니다.

```

bool AwsDoc::Medical_Imaging::cleanup(const Aws::String &stackName,
                                       const Aws::String &dataStoreId,
                                       const Aws::Client::ClientConfiguration
&clientConfiguration) {
    bool result = true;

    if (!stackName.empty() && askYesNoQuestion(
        "Would you like to delete the stack " + stackName + "? (y/n)")) {
        std::cout << "Deleting the image sets in the stack." << std::endl;
        result &= emptyDatastore(dataStoreId, clientConfiguration);
        printAsterisksLine();
        std::cout << "Deleting the stack." << std::endl;
        result &= deleteStack(stackName, clientConfiguration);
    }
    return result;
}

bool AwsDoc::Medical_Imaging::emptyDatastore(const Aws::String &datastoreID,
                                              const Aws::Client::ClientConfiguration
&clientConfiguration) {

    Aws::MedicalImaging::Model::SearchCriteria emptyCriteria;
    Aws::Vector<Aws::String> imageSetIDs;
    bool result = false;
    if (searchImageSets(datastoreID, emptyCriteria, imageSetIDs,
                        clientConfiguration)) {

```

```

    result = true;
    for (auto &imageSetID: imageSetIDs) {
        result &= deleteImageSet(datastoreId, imageSetID, clientConfiguration);
    }

    return result;
}

```

• API 세부 정보는 AWS SDK for C++ API 참조의 다음 주제를 참조하세요.

- [DeleteImageSet](#)
- [GetDICOMImportJob](#)
- [GetImageFrame](#)
- [GetImageSetMetadata](#)
- [SearchImageSets](#)
- [StartDICOMImportJob](#)

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

## SDK for C++를 사용한 IAM 예제

다음 코드 예제에서는 IAM과 AWS SDK for C++ 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 C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

CMakeLists.txt CMake 파일의 코드입니다.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS iam)

# Set this project's name.
project("hello_iam")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
"${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
```

```

endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory for
    # running and debugging.

    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you may
    # need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
        ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
    hello_iam.cpp)

target_link_libraries(${PROJECT_NAME}
    ${AWSSDK_LINK_LIBRARIES})

```

iam.cpp 소스 파일의 코드입니다.

```

#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/ListPoliciesRequest.h>
#include <iostream>
#include <iomanip>

/*
 * A "Hello IAM" starter application which initializes an AWS Identity and Access
 * Management (IAM) client
 * and lists the IAM policies.
 *
 * main function
 *
 * Usage: 'hello_iam'
 */

```

```

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        const Aws::String DATE_FORMAT("%Y-%m-%d");
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::IAM::IAMClient iamClient(clientConfig);
        Aws::IAM::Model::ListPoliciesRequest request;

        bool done = false;
        bool header = false;
        while (!done) {
            auto outcome = iamClient.ListPolicies(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Failed to list iam policies: " <<
                    outcome.GetError().GetMessage() << std::endl;
                result = 1;
                break;
            }

            if (!header) {
                std::cout << std::left << std::setw(55) << "Name" <<
                    std::setw(30) << "ID" << std::setw(80) << "Arn" <<
                    std::setw(64) << "Description" << std::setw(12) <<
                    "CreateDate" << std::endl;
                header = true;
            }

            const auto &policies = outcome.GetResult().GetPolicies();
            for (const auto &policy: policies) {
                std::cout << std::left << std::setw(55) <<
                    policy.GetPolicyName() << std::setw(30) <<
                    policy.GetPolicyId() << std::setw(80) << policy.GetArn()
<<
                    std::setw(64) << policy.GetDescription() << std::setw(12)
<<
                    policy.GetCreateDate().ToGmtString(DATE_FORMAT.c_str()) <<
                    std::endl;
            }
        }
    }
}

```

```

        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        } else {
            done = true;
        }
    }
}

Aws::ShutdownAPI(options); // Should only be called once.
return result;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ListPolicies](#)를 참조하세요.

## 기본 사항

### 기본 사항 알아보기

다음 코드 예제에서는 사용자를 생성하고 역할을 수입하는 방법을 보여줍니다.

#### Warning

보안 위험을 방지하려면 목적별 소프트웨어를 개발하거나 실제 데이터로 작업할 때 IAM 사용자를 인증에 사용하지 마세요. 대신 [AWS IAM Identity Center](#)과 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 수입할 수 있도록 정책을 추가합니다.
- 역할을 수입하고 임시 자격 증명 정보를 사용하여 S3 버킷을 나열한 후 리소스를 정리합니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예제 리포지토리](#)에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
namespace AwsDoc {
    namespace IAM {

        ///! Cleanup by deleting created entities.
        /*!
        \sa DeleteCreatedEntities
        \param client: IAM client.
        \param role: IAM role.
        \param user: IAM user.
        \param policy: IAM policy.
        */
        static bool DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                         const Aws::IAM::Model::Role &role,
                                         const Aws::IAM::Model::User &user,
                                         const Aws::IAM::Model::Policy &policy);

    }

    static const int LIST_BUCKETS_WAIT_SEC = 20;

    static const char ALLOCATION_TAG[] = "example_code";
}

///! Scenario to create an IAM user, create an IAM role, and apply the role to the
    user.
// "IAM access" permissions are needed to run this code.
// "STS assume role" permissions are needed to run this code. (Note: It might be
    necessary to
//     create a custom policy).
/*!
    \sa iamCreateUserAssumeRoleScenario
    \param clientConfig: Aws client configuration.
    \return bool: Successful completion.
*/
bool AwsDoc::IAM::iamCreateUserAssumeRoleScenario(
```

```

    const Aws::Client::ClientConfiguration &clientConfig) {

    Aws::IAM::IAMClient client(clientConfig);
    Aws::IAM::Model::User user;
    Aws::IAM::Model::Role role;
    Aws::IAM::Model::Policy policy;

    // 1. Create a user.
    {
        Aws::IAM::Model::CreateUserRequest request;
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String userName = "iam-demo-user-" +
                               Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetUserName(userName);

        Aws::IAM::Model::CreateUserOutcome outcome = client.CreateUser(request);
        if (!outcome.IsSuccess()) {
            std::cout << "Error creating IAM user " << userName << ":" <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        }
        else {
            std::cout << "Successfully created IAM user " << userName << std::endl;
        }

        user = outcome.GetResult().GetUser();
    }

    // 2. Create a role.
    {
        // Get the IAM user for the current client in order to access its ARN.
        Aws::String iamUserArn;
        {
            Aws::IAM::Model::GetUserRequest request;
            Aws::IAM::Model::GetUserOutcome outcome = client.GetUser(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error getting Iam user. " <<
                    outcome.GetError().GetMessage() << std::endl;

                DeleteCreatedEntities(client, role, user, policy);
                return false;
            }
            else {
                std::cout << "Successfully retrieved Iam user "

```

```

        << outcome.GetResult().GetUser().GetUserName()
        << std::endl;
    }

    iamUserArn = outcome.GetResult().GetUser().GetArn();
}

Aws::IAM::Model::CreateRoleRequest request;

Aws::String uuid = Aws::Utils::UUID::RandomUUID();
Aws::String roleName = "iam-demo-role-" +
    Aws::Utils::StringUtils::ToLower(uuid.c_str());
request.SetRoleName(roleName);

// Build policy document for role.
Aws::Utils::Document jsonStatement;
jsonStatement.WithString("Effect", "Allow");

Aws::Utils::Document jsonPrincipal;
jsonPrincipal.WithString("AWS", iamUserArn);
jsonStatement.WithObject("Principal", jsonPrincipal);
jsonStatement.WithString("Action", "sts:AssumeRole");
jsonStatement.WithObject("Condition", Aws::Utils::Document());

Aws::Utils::Document policyDocument;
policyDocument.WithString("Version", "2012-10-17");

Aws::Utils::Array<Aws::Utils::Document> statements(1);
statements[0] = jsonStatement;
policyDocument.WithArray("Statement", statements);

std::cout << "Setting policy for role\n    "
    << policyDocument.View().WriteCompact() << std::endl;

// Set role policy document as JSON string.
request.SetAssumeRolePolicyDocument(policyDocument.View().WriteCompact());

Aws::IAM::Model::CreateRoleOutcome outcome = client.CreateRole(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Error creating role. " <<
        outcome.GetError().GetMessage() << std::endl;

    DeleteCreatedEntities(client, role, user, policy);
    return false;
}

```

```

    }
    else {
        std::cout << "Successfully created a role with name " << roleName
                    << std::endl;
    }

    role = outcome.GetResult().GetRole();
}

// 3. Create an IAM policy.
{
    Aws::IAM::Model::CreatePolicyRequest request;
    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
    Aws::String policyName = "iam-demo-policy-" +
                             Aws::Utils::StringUtils::ToLower(uuid.c_str());
    request.SetPolicyName(policyName);

    // Build IAM policy document.
    Aws::Utils::Document jsonStatement;
    jsonStatement.WithString("Effect", "Allow");
    jsonStatement.WithString("Action", "s3:ListAllMyBuckets");
    jsonStatement.WithString("Resource", "arn:aws:s3:*");

    Aws::Utils::Document policyDocument;
    policyDocument.WithString("Version", "2012-10-17");

    Aws::Utils::Array<Aws::Utils::Document> statements(1);
    statements[0] = jsonStatement;
    policyDocument.WithArray("Statement", statements);

    std::cout << "Creating a policy.\n    " <<
policyDocument.View().WriteCompact()
                << std::endl;

    // Set IAM policy document as JSON string.
    request.SetPolicyDocument(policyDocument.View().WriteCompact());

    Aws::IAM::Model::CreatePolicyOutcome outcome = client.CreatePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating policy. " <<
outcome.GetError().GetMessage() << std::endl;

        DeleteCreatedEntities(client, role, user, policy);
        return false;
    }
}

```

```

    }
    else {
        std::cout << "Successfully created a policy with name, " << policyName
<<
        "." << std::endl;
    }

    policy = outcome.GetResult().GetPolicy();
}

// 4. Assume the new role using the AWS Security Token Service (STS).
Aws::STS::Model::Credentials credentials;
{
    Aws::STS::STSClient stsClient(clientConfig);

    Aws::STS::Model::AssumeRoleRequest request;
    request.SetRoleArn(role.GetArn());
    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
    Aws::String roleSessionName = "iam-demo-role-session-" +

    Aws::Utils::StringUtils::ToLower(uuid.c_str());
    request.SetRoleSessionName(roleSessionName);

    Aws::STS::Model::AssumeRoleOutcome assumeRoleOutcome;

    // Repeatedly call AssumeRole, because there is often a delay
    // before the role is available to be assumed.
    // Repeat at most 20 times when access is denied.
    int count = 0;
    while (true) {
        assumeRoleOutcome = stsClient.AssumeRole(request);
        if (!assumeRoleOutcome.IsSuccess()) {
            if (count > 20 ||
                assumeRoleOutcome.GetError().GetErrorType() !=
                Aws::STS::STSErrors::ACCESS_DENIED) {
                std::cerr << "Error assuming role after 20 tries. " <<
                    assumeRoleOutcome.GetError().GetMessage() <<
std::endl;

                DeleteCreatedEntities(client, role, user, policy);
                return false;
            }
            std::this_thread::sleep_for(std::chrono::seconds(1));
        }
    }
}

```

```

        else {
            std::cout << "Successfully assumed the role after " << count
                      << " seconds." << std::endl;

            break;
        }
        count++;
    }

    credentials = assumeRoleOutcome.GetResult().GetCredentials();
}

// 5. List objects in the bucket (This should fail).
{
    Aws::S3::S3Client s3Client(
        Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
                                    credentials.GetSecretAccessKey(),
                                    credentials.GetSessionToken()),
        Aws::MakeShared<Aws::S3::S3EndpointProvider>(ALLOCATION_TAG),
        clientConfig);
    Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
s3Client.ListBuckets();
    if (!listBucketsOutcome.IsSuccess()) {
        if (listBucketsOutcome.GetError().GetErrorType() !=
            Aws::S3::S3Errors::ACCESS_DENIED) {
            std::cerr << "Could not lists buckets. " <<
                listBucketsOutcome.GetError().GetMessage() << std::endl;
        }
        else {
            std::cout
                << "Access to list buckets denied because privileges have
not been applied."
                << std::endl;
        }
    }
    else {
        std::cerr
            << "Successfully retrieved bucket lists when this should not
happen."
            << std::endl;
    }
}

// 6. Attach the policy to the role.

```

```

{
    Aws::IAM::Model::AttachRolePolicyRequest request;
    request.SetRoleName(role.GetRoleName());
    request.WithPolicyArn(policy.GetArn());

    Aws::IAM::Model::AttachRolePolicyOutcome outcome = client.AttachRolePolicy(
        request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating policy. " <<
            outcome.GetError().GetMessage() << std::endl;

        DeleteCreatedEntities(client, role, user, policy);
        return false;
    }
    else {
        std::cout << "Successfully attached the policy with name, "
            << policy.GetPolicyName() <<
            ", to the role, " << role.GetRoleName() << "." << std::endl;
    }
}

int count = 0;
// 7. List objects in the bucket (this should succeed).
// Repeatedly call ListBuckets, because there is often a delay
// before the policy with ListBucket permissions has been applied to the role.
// Repeat at most LIST_BUCKETS_WAIT_SEC times when access is denied.
while (true) {
    Aws::S3::S3Client s3Client(
        Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
            credentials.GetSecretAccessKey(),
            credentials.GetSessionToken()),
        Aws::MakeShared<Aws::S3::S3EndpointProvider>(ALLOCATION_TAG),
        clientConfig);
    Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
s3Client.ListBuckets();
    if (!listBucketsOutcome.IsSuccess()) {
        if ((count > LIST_BUCKETS_WAIT_SEC) ||
            listBucketsOutcome.GetError().GetErrorType() !=
            Aws::S3::S3Errors::ACCESS_DENIED) {
            std::cerr << "Could not lists buckets after " <<
LIST_BUCKETS_WAIT_SEC << " seconds. " <<
                listBucketsOutcome.GetError().GetMessage() << std::endl;
            DeleteCreatedEntities(client, role, user, policy);
            return false;

```

```

    }

    std::this_thread::sleep_for(std::chrono::seconds(1));
}
else {

    std::cout << "Successfully retrieved bucket lists after " << count
               << " seconds." << std::endl;

    break;
}
count++;
}

// 8. Delete all the created resources.
return DeleteCreatedEntities(client, role, user, policy);
}

bool AwsDoc::IAM::DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                         const Aws::IAM::Model::Role &role,
                                         const Aws::IAM::Model::User &user,
                                         const Aws::IAM::Model::Policy &policy) {

    bool result = true;
    if (policy.ArnHasBeenSet()) {
        // Detach the policy from the role.
        {
            Aws::IAM::Model::DetachRolePolicyRequest request;
            request.SetPolicyArn(policy.GetArn());
            request.SetRoleName(role.GetRoleName());

            Aws::IAM::Model::DetachRolePolicyOutcome outcome =
client.DetachRolePolicy(
    request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error Detaching policy from roles. " <<
                    outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
            else {
                std::cout << "Successfully detached the policy with arn "
                    << policy.GetArn()
                    << " from role " << role.GetRoleName() << "." <<
std::endl;
            }
        }
    }
}

```

```
// Delete the policy.
{
    Aws::IAM::Model::DeletePolicyRequest request;
    request.WithPolicyArn(policy.GetArn());

    Aws::IAM::Model::DeletePolicyOutcome outcome =
client.DeletePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting policy. " <<
            outcome.GetError().GetMessage() << std::endl;
        result = false;
    }
    else {
        std::cout << "Successfully deleted the policy with arn "
            << policy.GetArn() << std::endl;
    }
}

}

if (role.RoleIdHasBeenSet()) {
    // Delete the role.
    Aws::IAM::Model::DeleteRoleRequest request;
    request.SetRoleName(role.GetRoleName());

    Aws::IAM::Model::DeleteRoleOutcome outcome = client.DeleteRole(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting role. " <<
            outcome.GetError().GetMessage() << std::endl;
        result = false;
    }
    else {
        std::cout << "Successfully deleted the role with name "
            << role.GetRoleName() << std::endl;
    }
}

}

if (user.ArnHasBeenSet()) {
    // Delete the user.
    Aws::IAM::Model::DeleteUserRequest request;
    request.WithUserName(user.GetUserName());

    Aws::IAM::Model::DeleteUserOutcome outcome = client.DeleteUser(request);
```

```
        if (!outcome.IsSuccess()) {
            std::cerr << "Error deleting user. " <<
                outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Successfully deleted the user with name "
                << user.GetUserName() << std::endl;
        }
    }

    return result;
}
```

• API 세부 정보는 AWS SDK for C++ API 참조의 다음 주제를 참조하십시오.

- [AttachRolePolicy](#)
- [CreateAccessKey](#)
- [CreatePolicy](#)
- [CreateRole](#)
- [CreateUser](#)
- [DeleteAccessKey](#)
- [DeletePolicy](#)
- [DeleteRole](#)
- [DeleteUser](#)
- [DeleteUserPolicy](#)
- [DetachRolePolicy](#)
- [PutUserPolicy](#)

## 작업

### AttachRolePolicy

다음 코드 예시는 AttachRolePolicy의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::IAM::attachRolePolicy(const Aws::String &roleName,
                                   const Aws::String &policyArn,
                                   const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::ListAttachedRolePoliciesRequest list_request;
    list_request.SetRoleName(roleName);

    bool done = false;
    while (!done) {
        auto list_outcome = iam.ListAttachedRolePolicies(list_request);
        if (!list_outcome.IsSuccess()) {
            std::cerr << "Failed to list attached policies of role " <<
                roleName << ": " << list_outcome.GetError().GetMessage() <<
                std::endl;
            return false;
        }

        const auto &policies = list_outcome.GetResult().GetAttachedPolicies();
        if (std::any_of(policies.cbegin(), policies.cend(),
            [=](const Aws::IAM::Model::AttachedPolicy &policy) {
                return policy.GetPolicyArn() == policyArn;
            })) {
            std::cout << "Policy " << policyArn <<
                " is already attached to role " << roleName << std::endl;
            return true;
        }

        done = !list_outcome.GetResult().GetIsTruncated();
        list_request.SetMarker(list_outcome.GetResult().GetMarker());
    }

    Aws::IAM::Model::AttachRolePolicyRequest request;
```

```

    request.SetRoleName(roleName);
    request.SetPolicyArn(policyArn);

    Aws::IAM::Model::AttachRolePolicyOutcome outcome =
    iam.AttachRolePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to attach policy " << policyArn << " to role " <<
            roleName << ": " << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully attached policy " << policyArn << " to role " <<
            roleName << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [AttachRolePolicy](#)를 참조하세요.

## CreateAccessKey

다음 코드 예시는 CreateAccessKey의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

Aws::String AwsDoc::IAM::createAccessKey(const Aws::String &userName,
                                          const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::CreateAccessKeyRequest request;
    request.SetUserName(userName);

    Aws::String result;

```

```

    Aws::IAM::Model::CreateAccessKeyOutcome outcome = iam.CreateAccessKey(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating access key for IAM user " << userName
            << ":" << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        const auto &accessKey = outcome.GetResult().GetAccessKey();
        std::cout << "Successfully created access key for IAM user " <<
            userName << std::endl << "  aws_access_key_id = " <<
            accessKey.GetAccessKeyId() << std::endl <<
            "  aws_secret_access_key = " << accessKey.GetSecretAccessKey() <<
            std::endl;
        result = accessKey.GetAccessKeyId();
    }

    return result;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CreateAccessKey](#)를 참조하세요.

## CreateAccountAlias

다음 코드 예시는 CreateAccountAlias의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

bool AwsDoc::IAM::createAccountAlias(const Aws::String &aliasName,
                                     const Aws::Client::ClientConfiguration
    &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::CreateAccountAliasRequest request;
    request.SetAccountAlias(aliasName);

    Aws::IAM::Model::CreateAccountAliasOutcome outcome = iam.CreateAccountAlias(

```

```

        request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating account alias " << aliasName << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully created account alias " << aliasName <<
                  << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CreateAccountAlias](#)를 참조하세요.

## CreatePolicy

다음 코드 예시는 CreatePolicy의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

Aws::String AwsDoc::IAM::createPolicy(const Aws::String &policyName,
                                       const Aws::String &rsrcArn,
                                       const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::CreatePolicyRequest request;
    request.SetPolicyName(policyName);
    request.SetPolicyDocument(BuildSamplePolicyDocument(rsrcArn));

    Aws::IAM::Model::CreatePolicyOutcome outcome = iam.CreatePolicy(request);
    Aws::String result;
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating policy " << policyName << ": " <<

```

```

        outcome.GetError().GetMessage() << std::endl;
    }
    else {
        result = outcome.GetResult().GetPolicy().GetArn();
        std::cout << "Successfully created policy " << policyName <<
            std::endl;
    }

    return result;
}

Aws::String AwsDoc::IAM::BuildSamplePolicyDocument(const Aws::String &rsrc_arn) {
    std::stringstream stringStream;
    stringStream << "{"
        << "  \"Version\": \"2012-10-17\", "
        << "  \"Statement\": ["
        << "    {"
        << "      \"Effect\": \"Allow\", "
        << "      \"Action\": \"logs:CreateLogGroup\", "
        << "      \"Resource\": \""
        << rsrc_arn
        << "\"\"
        << "    }, "
        << "    {"
        << "      \"Effect\": \"Allow\", "
        << "      \"Action\": ["
        << "        \"dynamodb:DeleteItem\", "
        << "        \"dynamodb:GetItem\", "
        << "        \"dynamodb:PutItem\", "
        << "        \"dynamodb:Scan\", "
        << "        \"dynamodb:UpdateItem\"
        << "      ], "
        << "      \"Resource\": \""
        << rsrc_arn
        << "\"\"
        << "    }
        << "  ]"
        << "}";

    return stringStream.str();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CreatePolicy](#)를 참조하세요.

## CreateRole

다음 코드 예시는 CreateRole의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::IAM::createIamRole(
    const Aws::String &roleName,
    const Aws::String &policy,
    const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient client(clientConfig);
    Aws::IAM::Model::CreateRoleRequest request;

    request.SetRoleName(roleName);
    request.SetAssumeRolePolicyDocument(policy);

    Aws::IAM::Model::CreateRoleOutcome outcome = client.CreateRole(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating role. " <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else {
        const Aws::IAM::Model::Role iamRole = outcome.GetResult().GetRole();
        std::cout << "Created role " << iamRole.GetRoleName() << "\n";
        std::cout << "ID: " << iamRole.GetRoleId() << "\n";
        std::cout << "ARN: " << iamRole.GetArn() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 [AWS SDK for C++ API 참조](#)의 CreateRole을 참조하세요.

## CreateUser

다음 코드 예시는 CreateUser의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::IAM::IAMClient iam(clientConfig);

Aws::IAM::Model::CreateUserRequest create_request;
create_request.SetUserName(userName);

auto create_outcome = iam.CreateUser(create_request);
if (!create_outcome.IsSuccess()) {
    std::cerr << "Error creating IAM user " << userName << ":" <<
        create_outcome.GetError().GetMessage() << std::endl;
}
else {
    std::cout << "Successfully created IAM user " << userName << std::endl;
}

return create_outcome.IsSuccess();
```

- API 세부 정보는 [AWS SDK for C++ API 참조](#)의 CreateUser를 참조하세요.

## DeleteAccessKey

다음 코드 예시는 DeleteAccessKey의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::IAM::deleteAccessKey(const Aws::String &userName,
                                   const Aws::String &accessKeyID,
                                   const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DeleteAccessKeyRequest request;
    request.SetUserName(userName);
    request.SetAccessKeyId(accessKeyID);

    auto outcome = iam.DeleteAccessKey(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting access key " << accessKeyID << " from user "
                  << userName << ": " << outcome.GetError().GetMessage() <<
                  std::endl;
    }
    else {
        std::cout << "Successfully deleted access key " << accessKeyID
                  << " for IAM user " << userName << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteAccessKey](#)를 참조하세요.

**DeleteAccountAlias**

다음 코드 예시는 DeleteAccountAlias의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::IAM::deleteAccountAlias(const Aws::String &accountAlias,
                                     const Aws::Client::ClientConfiguration
                                     &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DeleteAccountAliasRequest request;
    request.SetAccountAlias(accountAlias);

    const auto outcome = iam.DeleteAccountAlias(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting account alias " << accountAlias << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully deleted account alias " << accountAlias <<
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteAccountAlias](#)를 참조하세요.

## DeletePolicy

다음 코드 예시는 DeletePolicy의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::IAM::deletePolicy(const Aws::String &policyArn,
                               const Aws::Client::ClientConfiguration &clientConfig)
{
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::DeletePolicyRequest request;
    request.SetPolicyArn(policyArn);

    auto outcome = iam.DeletePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting policy with arn " << policyArn << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully deleted policy with arn " << policyArn
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeletePolicy](#)를 참조하세요.

## DeleteServerCertificate

다음 코드 예시는 DeleteServerCertificate의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::IAM::deleteServerCertificate(const Aws::String &certificateName,
                                           const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::DeleteServerCertificateRequest request;
    request.SetServerCertificateName(certificateName);

    const auto outcome = iam.DeleteServerCertificate(request);
    bool result = true;
    if (!outcome.IsSuccess()) {
        if (outcome.GetError().GetErrorType() !=
        Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error deleting server certificate " << certificateName <<
                ": " << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Certificate '" << certificateName
                << "' not found." << std::endl;
        }
    }
    else {
        std::cout << "Successfully deleted server certificate " << certificateName
            << std::endl;
    }

    return result;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteServerCertificate](#)를 참조하세요.

## DeleteUser

다음 코드 예시는 DeleteUser의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::IAM::IAMClient iam(clientConfig);

Aws::IAM::Model::DeleteUserRequest request;
request.SetUserName(userName);
auto outcome = iam.DeleteUser(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Error deleting IAM user " << userName << ": " <<
        outcome.GetError().GetMessage() << std::endl;;
}
else {
    std::cout << "Successfully deleted IAM user " << userName << std::endl;
}

return outcome.IsSuccess();
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteUser](#)를 참조하세요.

## DetachRolePolicy

다음 코드 예시는 DetachRolePolicy의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

Aws::IAM::IAMClient iam(clientConfig);

Aws::IAM::Model::DetachRolePolicyRequest detachRequest;
detachRequest.SetRoleName(roleName);
detachRequest.SetPolicyArn(policyArn);

auto detachOutcome = iam.DetachRolePolicy(detachRequest);
if (!detachOutcome.IsSuccess()) {
    std::cerr << "Failed to detach policy " << policyArn << " from role "
               << roleName << ": " << detachOutcome.GetError().GetMessage() <<
               std::endl;
}
else {
    std::cout << "Successfully detached policy " << policyArn << " from role "
              << roleName << std::endl;
}

return detachOutcome.IsSuccess();

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DetachRolePolicy](#)를 참조하세요.

## GetAccessKeyLastUsed

다음 코드 예시는 GetAccessKeyLastUsed의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

bool AwsDoc::IAM::accessKeyLastUsed(const Aws::String &secretKeyID,
                                     const Aws::Client::ClientConfiguration
                                     &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetAccessKeyLastUsedRequest request;

    request.SetAccessKeyId(secretKeyID);

```

```

    Aws::IAM::Model::GetAccessKeyLastUsedOutcome outcome = iam.GetAccessKeyLastUsed(
        request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error querying last used time for access key " <<
            secretKeyID << ":" << outcome.GetError().GetMessage() <<
std::endl;
    }
    else {
        Aws::String lastUsedTimeString =
            outcome.GetResult()
                .GetAccessKeyLastUsed()
                .GetLastUsedDate()
                .ToGmtString(Aws::Utils::DateFormat::ISO_8601);
        std::cout << "Access key " << secretKeyID << " last used at time " <<
            lastUsedTimeString << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetAccessKeyLastUsed](#)를 참조하세요.

## GetPolicy

다음 코드 예시는 GetPolicy의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

bool AwsDoc::IAM::getPolicy(const Aws::String &policyArn,
                           const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetPolicyRequest request;
    request.SetPolicyArn(policyArn);
}

```

```

auto outcome = iam.GetPolicy(request);
if (!outcome.IsSuccess()) {
    std::cerr << "Error getting policy " << policyArn << ": " <<
        outcome.GetError().GetMessage() << std::endl;
}
else {
    const auto &policy = outcome.GetResult().GetPolicy();
    std::cout << "Name: " << policy.GetPolicyName() << std::endl <<
        "ID: " << policy.GetPolicyId() << std::endl << "Arn: " <<
        policy.GetArn() << std::endl << "Description: " <<
        policy.GetDescription() << std::endl << "CreateDate: " <<
        policy.GetCreateDate().ToGmtString(Aws::Utils::DateFormat::ISO_8601)
            << std::endl;
}

return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetPolicy](#)를 참조하세요.

## GetServerCertificate

다음 코드 예시는 GetServerCertificate의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

bool AwsDoc::IAM::getServerCertificate(const Aws::String &certificateName,
                                       const Aws::Client::ClientConfiguration
                                       &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetServerCertificateRequest request;
    request.SetServerCertificateName(certificateName);
}

```

```

auto outcome = iam.GetServerCertificate(request);
bool result = true;
if (!outcome.IsSuccess()) {
    if (outcome.GetError().GetErrorType() !=
        Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
        std::cerr << "Error getting server certificate " << certificateName <<
            ": " << outcome.GetError().GetMessage() << std::endl;
        result = false;
    }
    else {
        std::cout << "Certificate '" << certificateName
            << "' not found." << std::endl;
    }
}
else {
    const auto &certificate = outcome.GetResult().GetServerCertificate();
    std::cout << "Name: " <<

certificate.GetServerCertificateMetadata().GetServerCertificateName()
    << std::endl << "Body: " << certificate.GetCertificateBody() <<
    std::endl << "Chain: " << certificate.GetCertificateChain() <<
    std::endl;
}

return result;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetServerCertificate](#)를 참조하세요.

## ListAccessKeys

다음 코드 예시는 ListAccessKeys의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

bool AwsDoc::IAM::listAccessKeys(const Aws::String &userName,
                                const Aws::Client::ClientConfiguration
                                &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListAccessKeysRequest request;
    request.SetUserName(userName);

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListAccessKeys(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list access keys for user " << userName
                      << ": " << outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(32) << "UserName" <<
                      std::setw(30) << "KeyID" << std::setw(20) << "Status" <<
                      std::setw(20) << "CreateDate" << std::endl;
            header = true;
        }

        const auto &keys = outcome.GetResult().GetAccessKeyMetadata();
        const Aws::String DATE_FORMAT = "%Y-%m-%d";

        for (const auto &key: keys) {
            Aws::String statusString =
                Aws::IAM::Model::StatusTypeMapper::GetNameForStatusType(
                    key.GetStatus());
            std::cout << std::left << std::setw(32) << key.GetUserName() <<
                      std::setw(30) << key.GetAccessKeyId() << std::setw(20) <<
                      statusString << std::setw(20) <<
                      key.GetCreateDate().ToGmtString(DATE_FORMAT.c_str()) <<
std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }
}

```

```

    }
}

return true;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ListAccessKeys](#)를 참조하세요.

## ListAccountAliases

다음 코드 예시는 ListAccountAliases의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

bool
AwsDoc::IAM::listAccountAliases(const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListAccountAliasesRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListAccountAliases(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list account aliases: " <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        const auto &aliases = outcome.GetResult().GetAccountAliases();
        if (!header) {
            if (aliases.size() == 0) {
                std::cout << "Account has no aliases" << std::endl;
            }
        }
    }
}

```

```

        break;
    }
    std::cout << std::left << std::setw(32) << "Alias" << std::endl;
    header = true;
}

for (const auto &alias: aliases) {
    std::cout << std::left << std::setw(32) << alias << std::endl;
}

if (outcome.GetResult().GetIsTruncated()) {
    request.SetMarker(outcome.GetResult().GetMarker());
}
else {
    done = true;
}
}

return true;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ListAccountAliases](#)를 참조하세요.

## ListPolicies

다음 코드 예시는 ListPolicies의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

bool AwsDoc::IAM::listPolicies(const Aws::Client::ClientConfiguration &clientConfig)
{
    const Aws::String DATE_FORMAT("%Y-%m-%d");
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListPoliciesRequest request;

```

```

bool done = false;
bool header = false;
while (!done) {
    auto outcome = iam.ListPolicies(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to list iam policies: " <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    if (!header) {
        std::cout << std::left << std::setw(55) << "Name" <<
            std::setw(30) << "ID" << std::setw(80) << "Arn" <<
            std::setw(64) << "Description" << std::setw(12) <<
            "CreateDate" << std::endl;
        header = true;
    }

    const auto &policies = outcome.GetResult().GetPolicies();
    for (const auto &policy: policies) {
        std::cout << std::left << std::setw(55) <<
            policy.GetPolicyName() << std::setw(30) <<
            policy.GetPolicyId() << std::setw(80) << policy.GetArn() <<
            std::setw(64) << policy.GetDescription() << std::setw(12) <<
            policy.GetCreateDate().ToGmtString(DATE_FORMAT.c_str()) <<
            std::endl;
    }

    if (outcome.GetResult().GetIsTruncated()) {
        request.SetMarker(outcome.GetResult().GetMarker());
    }
    else {
        done = true;
    }
}

return true;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ListPolicies](#)를 참조하세요.

## ListServerCertificates

다음 코드 예시는 ListServerCertificates의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::IAM::listServerCertificates(
    const Aws::Client::ClientConfiguration &clientConfig) {
    const Aws::String DATE_FORMAT = "%Y-%m-%d";

    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListServerCertificatesRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListServerCertificates(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list server certificates: " <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(55) << "Name" <<
                std::setw(30) << "ID" << std::setw(80) << "Arn" <<
                std::setw(14) << "UploadDate" << std::setw(14) <<
                "ExpirationDate" << std::endl;
            header = true;
        }

        const auto &certificates =
            outcome.GetResult().GetServerCertificateMetadataList();

        for (const auto &certificate: certificates) {
            std::cout << std::left << std::setw(55) <<
                certificate.GetServerCertificateName() << std::setw(30) <<
```

```

        certificate.GetServerCertificateId() << std::setw(80) <<
        certificate.GetArn() << std::setw(14) <<
        certificate.GetUploadDate().ToGmtString(DATE_FORMAT.c_str())
    <<

        std::setw(14) <<
        certificate.GetExpiration().ToGmtString(DATE_FORMAT.c_str())
    <<

        std::endl;
    }

    if (outcome.GetResult().GetIsTruncated()) {
        request.SetMarker(outcome.GetResult().GetMarker());
    }
    else {
        done = true;
    }
}

return true;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ListServerCertificates](#)를 참조하세요.

## ListUsers

다음 코드 예시는 ListUsers의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

bool AwsDoc::IAM::listUsers(const Aws::Client::ClientConfiguration &clientConfig) {
    const Aws::String DATE_FORMAT = "%Y-%m-%d";
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListUsersRequest request;

```

```

bool done = false;
bool header = false;
while (!done) {
    auto outcome = iam.ListUsers(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to list iam users:" <<
            outcome.GetError().GetMessage() << std::endl;
        return false;
    }

    if (!header) {
        std::cout << std::left << std::setw(32) << "Name" <<
            std::setw(30) << "ID" << std::setw(64) << "Arn" <<
            std::setw(20) << "CreateDate" << std::endl;
        header = true;
    }

    const auto &users = outcome.GetResult().GetUsers();
    for (const auto &user: users) {
        std::cout << std::left << std::setw(32) << user.GetUserName() <<
            std::setw(30) << user.GetUserId() << std::setw(64) <<
            user.GetArn() << std::setw(20) <<
            user.GetCreateDate().ToGmtString(DATE_FORMAT.c_str())
            << std::endl;
    }

    if (outcome.GetResult().GetIsTruncated()) {
        request.SetMarker(outcome.GetResult().GetMarker());
    }
    else {
        done = true;
    }
}

return true;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ListUsers](#)를 참조하세요.

## PutRolePolicy

다음 코드 예시는 PutRolePolicy의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::IAM::putRolePolicy(
    const Aws::String &roleName,
    const Aws::String &policyName,
    const Aws::String &policyDocument,
    const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient iamClient(clientConfig);
    Aws::IAM::Model::PutRolePolicyRequest request;

    request.SetRoleName(roleName);
    request.SetPolicyName(policyName);
    request.SetPolicyDocument(policyDocument);

    Aws::IAM::Model::PutRolePolicyOutcome outcome =
    iamClient.PutRolePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error putting policy on role. " <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully put the role policy." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [PutRolePolicy](#)를 참조하세요.

**UpdateAccessKey**

다음 코드 예시는 UpdateAccessKey의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::IAM::updateAccessKey(const Aws::String &userName,
                                   const Aws::String &accessKeyID,
                                   Aws::IAM::Model::StatusType status,
                                   const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::UpdateAccessKeyRequest request;
    request.SetUserName(userName);
    request.SetAccessKeyId(accessKeyID);
    request.SetStatus(status);

    auto outcome = iam.UpdateAccessKey(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated status of access key "
                    << accessKeyID << " for user " << userName << std::endl;
    }
    else {
        std::cerr << "Error updated status of access key " << accessKeyID <<
                    " for user " << userName << ": " <<
                    outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [UpdateAccessKey](#)를 참조하세요.

## UpdateServerCertificate

다음 코드 예시는 UpdateServerCertificate의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::IAM::updateServerCertificate(const Aws::String &currentCertificateName,
                                          const Aws::String &newCertificateName,
                                          const Aws::Client::ClientConfiguration
&clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::UpdateServerCertificateRequest request;
    request.SetServerCertificateName(currentCertificateName);
    request.SetNewServerCertificateName(newCertificateName);

    auto outcome = iam.UpdateServerCertificate(request);
    bool result = true;
    if (outcome.IsSuccess()) {
        std::cout << "Server certificate " << currentCertificateName
                  << " successfully renamed as " << newCertificateName
                  << std::endl;
    }
    else {
        if (outcome.GetError().GetErrorType() !=
            Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error changing name of server certificate " <<
                      currentCertificateName << " to " << newCertificateName << ":"
<<
                      outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Certificate '" << currentCertificateName
                  << "' not found." << std::endl;
        }
    }

    return result;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [UpdateServerCertificate](#)를 참조하세요.

## UpdateUser

다음 코드 예시는 UpdateUser의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::IAM::updateUser(const Aws::String &currentUserName,
                             const Aws::String &newUserName,
                             const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::UpdateUserRequest request;
    request.SetUserName(currentUserName);
    request.SetNewUserName(newUserName);

    auto outcome = iam.UpdateUser(request);
    if (outcome.IsSuccess()) {
        std::cout << "IAM user " << currentUserName <<
            " successfully updated with new user name " << newUserName <<
            std::endl;
    }
    else {
        std::cerr << "Error updating user name for IAM user " << currentUserName <<
            ":" << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [UpdateUser](#)를 참조하세요.

# AWS IoT SDK for C++를 사용한 예제

다음 코드 예제에서는 AWS IoT SDK for C++를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS IoT.

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

안녕하세요 AWS IoT

다음 코드 예제에서는 AWS IoT를 사용하여 시작하는 방법을 보여 줍니다.

SDK for C++

CMakeLists.txt CMake 파일의 코드입니다.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS iot)

# Set this project's name.
project("hello_iot")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
```

```

set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
"${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory for
    running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you may
    need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
"${CMAKE_CURRENT_BINARY_DIR}/${BIN_SUB_DIR}")
endif ()

add_executable(${PROJECT_NAME}
    hello_iot.cpp)

target_link_libraries(${PROJECT_NAME}
    ${AWSSDK_LINK_LIBRARIES})

```

hello\_iot.cpp 소스 파일의 코드입니다.

```

#include <aws/core/Aws.h>
#include <aws/iot/IoTClient.h>
#include <aws/iot/model/ListThingsRequest.h>
#include <iostream>

/*
 * A "Hello IoT" starter application which initializes an AWS IoT client and

```

```

* lists the AWS IoT topics in the current account.
*
* main function
*
* Usage: 'hello_iot'
*
*/

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optional: change the log level for debugging.
    // options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::IoT::IoTClient iotClient(clientConfig);
        // List the things in the current account.
        Aws::IoT::Model::ListThingsRequest listThingsRequest;

        Aws::String nextToken; // Used for pagination.
        Aws::Vector<Aws::IoT::Model::ThingAttribute> allThings;

        do {
            if (!nextToken.empty()) {
                listThingsRequest.SetNextToken(nextToken);
            }

            Aws::IoT::Model::ListThingsOutcome listThingsOutcome =
            iotClient.ListThings(
                listThingsRequest);
            if (listThingsOutcome.IsSuccess()) {
                const Aws::Vector<Aws::IoT::Model::ThingAttribute> &things =
            listThingsOutcome.GetResult().GetThings();
                allThings.insert(allThings.end(), things.begin(), things.end());
                nextToken = listThingsOutcome.GetResult().GetNextToken();
            }
            else {
                std::cerr << "List things failed"
                    << listThingsOutcome.GetError().GetMessage() << std::endl;
                break;
            }
        }
    }
}

```

```

    } while (!nextToken.empty());

    std::cout << allThings.size() << " thing(s) found." << std::endl;
    for (auto const &thing: allThings) {
        std::cout << thing.GetThingName() << std::endl;
    }
}

Aws::ShutdownAPI(options); // Should only be called once.
return 0;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [listThings](#)를 참조하세요.

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

## 기본 사항

### 기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- AWS IoT 사물을 생성합니다.
- 디바이스 인증서를 생성합니다.
- 속성을 사용하여 AWS IoT 사물을 업데이트합니다.
- 고유한 엔드포인트를 반환합니다.
- AWS IoT 인증서를 나열합니다.
- 새도우를 업데이트 AWS IoT 합니다.
- 상태 정보를 씁니다.
- 규칙을 생성합니다.
- 규칙을 나열합니다.
- 사물 이름을 사용하여 사물을 검색합니다.
- AWS IoT 사물을 삭제합니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

AWS IoT 사물을 생성합니다.

```
Aws::String thingName = askQuestion("Enter a thing name: ");

if (!createThing(thingName, clientConfiguration)) {
    std::cerr << "Exiting because createThing failed." << std::endl;
    cleanup("", "", "", "", "", false, clientConfiguration);
    return false;
}
```

```
///  
/*!  
 \param thingName: The name for the thing.  
 \param clientConfiguration: AWS client configuration.  
 \return bool: Function succeeded.  
 */  
bool AwsDoc::IoT::createThing(const Aws::String &thingName,  
                               const Aws::Client::ClientConfiguration  
                               &clientConfiguration) {  
    Aws::IoT::IoTClient iotClient(clientConfiguration);  
    Aws::IoT::Model::CreateThingRequest createThingRequest;  
    createThingRequest.SetThingName(thingName);  
  
    Aws::IoT::Model::CreateThingOutcome outcome = iotClient.CreateThing(  
        createThingRequest);  
    if (outcome.IsSuccess()) {  
        std::cout << "Successfully created thing " << thingName << std::endl;  
    }  
    else {  
        std::cerr << "Failed to create thing " << thingName << ": " <<  
            outcome.GetError().GetMessage() << std::endl;  
    }  
}
```

```
    return outcome.IsSuccess();
}
```

디바이스 인증서를 생성합니다.

```
Aws::String certificateARN;
Aws::String certificateID;
if (askYesNoQuestion("Would you like to create a certificate for your thing? (y/
n) ")) {
    Aws::String outputFolder;
    if (askYesNoQuestion(
        "Would you like to save the certificate and keys to file? (y/n) "))
    {
        outputFolder = std::filesystem::current_path();
        outputFolder += "/device_keys_and_certificates";

        std::filesystem::create_directories(outputFolder);

        std::cout << "The certificate and keys will be saved to the folder: "
                    << outputFolder << std::endl;
    }

    if (!createKeysAndCertificate(outputFolder, certificateARN, certificateID,
                                clientConfiguration)) {
        std::cerr << "Exiting because createKeysAndCertificate failed."
                    << std::endl;
        cleanup(thingName, "", "", "", "", false, clientConfiguration);
        return false;
    }

    std::cout << "\nNext, the certificate will be attached to the thing.\n"
                << std::endl;
    if (!attachThingPrincipal(certificateARN, thingName, clientConfiguration)) {
        std::cerr << "Exiting because attachThingPrincipal failed." <<
std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "",
                false,
                clientConfiguration);
        return false;
    }
}
```

```

//! Create keys and certificate for an Aws IoT device.
//! This routine will save certificates and keys to an output folder, if provided.
/*!
    \param outputFolder: Location for storing output in files, ignored when string is
    empty.
    \param certificateARNResult: A string to receive the ARN of the created
    certificate.
    \param certificateID: A string to receive the ID of the created certificate.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::IoT::createKeysAndCertificate(const Aws::String &outputFolder,
                                           Aws::String &certificateARNResult,
                                           Aws::String &certificateID,
                                           const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::CreateKeysAndCertificateRequest
createKeysAndCertificateRequest;

    Aws::IoT::Model::CreateKeysAndCertificateOutcome outcome =
        client.CreateKeysAndCertificate(createKeysAndCertificateRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created a certificate and keys" << std::endl;
        certificateARNResult = outcome.GetResult().GetCertificateArn();
        certificateID = outcome.GetResult().GetCertificateId();
        std::cout << "Certificate ARN: " << certificateARNResult << ", certificate
ID: "
                << certificateID << std::endl;

        if (!outputFolder.empty()) {
            std::cout << "Writing certificate and keys to the folder '" <<
outputFolder
                << "'." << std::endl;
            std::cout << "Be sure these files are stored securely." << std::endl;

            Aws::String certificateFilePath = outputFolder + "/certificate.pem.crt";
            std::ofstream certificateFile(certificateFilePath);
            if (!certificateFile.is_open()) {
                std::cerr << "Error opening certificate file, '" <<
certificateFilePath
                    << "'."
                    << std::endl;
            }
        }
    }
}

```

```

        return false;
    }
    certificateFile << outcome.GetResult().GetCertificatePem();
    certificateFile.close();

    const Aws::IoT::Model::KeyPair &keyPair =
outcome.GetResult().GetKeyPair();

    Aws::String privateKeyFilePath = outputFolder + "/private.pem.key";
    std::ofstream privateKeyFile(privateKeyFilePath);
    if (!privateKeyFile.is_open()) {
        std::cerr << "Error opening private key file, '" <<
privateKeyFilePath
                << "'."
                << std::endl;
        return false;
    }
    privateKeyFile << keyPair.GetPrivateKey();
    privateKeyFile.close();

    Aws::String publicKeyFilePath = outputFolder + "/public.pem.key";
    std::ofstream publicKeyFile(publicKeyFilePath);
    if (!publicKeyFile.is_open()) {
        std::cerr << "Error opening public key file, '" << publicKeyFilePath
                << "'."
                << std::endl;
        return false;
    }
    publicKeyFile << keyPair.GetPublicKey();
}
}
else {
    std::cerr << "Error creating keys and certificate: "
                << outcome.GetError().GetMessage() << std::endl;
}

return outcome.IsSuccess();
}

//! Attach a principal to an AWS IoT thing.
/*!
    \param principal: A principal to attach.
    \param thingName: The name for the thing.
    \param clientConfiguration: AWS client configuration.

```

```

    \return bool: Function succeeded.
    */
bool AwsDoc::IoT::attachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::AttachThingPrincipalRequest request;
    request.SetPrincipal(principal);
    request.SetThingName(thingName);
    Aws::IoT::Model::AttachThingPrincipalOutcome outcome =
client.AttachThingPrincipal(
    request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully attached principal to thing." << std::endl;
    }
    else {
        std::cerr << "Failed to attach principal to thing." <<
outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

AWS IoT 사물에 대해 다양한 작업을 수행합니다.

```

    if (!updateThing(thingName, { {"location", "Office"}, {"firmwareVersion",
"v2.0"} }, clientConfiguration)) {
        std::cerr << "Exiting because updateThing failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
clientConfiguration);
        return false;
    }

    printAsterisksLine();

    std::cout << "Now an endpoint will be retrieved for your account.\n" <<
std::endl;
    std::cout << "An IoT Endpoint refers to a specific URL or Uniform Resource
Locator that serves as the entry point\n"
    << "for communication between IoT devices and the AWS IoT service." <<
std::endl;

```

```

askQuestion("Press Enter to continue:", alwaysTrueTest);

Aws::String endpoint;
if (!describeEndpoint(endpoint, clientConfiguration)) {
    std::cerr << "Exiting because getEndpoint failed." << std::endl;
    cleanup(thingName, certificateARN, certificateID, "", "", false,
            clientConfiguration);
    return false;
}
std::cout << "Your endpoint is " << endpoint << "." << std::endl;
printAsterisksLine();

std::cout << "Now the certificates in your account will be listed." <<
std::endl;
askQuestion("Press Enter to continue:", alwaysTrueTest);

if (!listCertificates(clientConfiguration)) {
    std::cerr << "Exiting because listCertificates failed." << std::endl;
    cleanup(thingName, certificateARN, certificateID, "", "", false,
            clientConfiguration);
    return false;
}

printAsterisksLine();

std::cout << "Now the shadow for the thing will be updated.\n" << std::endl;
std::cout << "A thing shadow refers to a feature that enables you to create a
virtual representation, or \"shadow,\"\"n"
    << "of a physical device or thing. The thing shadow allows you to synchronize
and control the state of a device between\n"
    << "the cloud and the device itself. and the AWS IoT service. For example, you
can write and retrieve JSON data from a thing shadow." << std::endl;
askQuestion("Press Enter to continue:", alwaysTrueTest);

if (!updateThingShadow(thingName, R"({"state":{"reported":
{"temperature":25,"humidity":50}}})", clientConfiguration)) {
    std::cerr << "Exiting because updateThingShadow failed." << std::endl;
    cleanup(thingName, certificateARN, certificateID, "", "", false,
            clientConfiguration);
    return false;
}

printAsterisksLine();

```

```

    std::cout << "Now, the state information for the shadow will be retrieved.\n" <<
std::endl;
    askQuestion("Press Enter to continue:", alwaysTrueTest);

    Aws::String shadowState;
    if (!getThingShadow(thingName, shadowState, clientConfiguration)) {
        std::cerr << "Exiting because getThingShadow failed." << std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
            clientConfiguration);
        return false;
    }
    std::cout << "The retrieved shadow state is: " << shadowState << std::endl;

    printAsterisksLine();

    std::cout << "A rule with now be added to to the thing.\n" << std::endl;
    std::cout << "Any user who has permission to create rules will be able to access
data processed by the rule." << std::endl;
    std::cout << "In this case, the rule will use an Simple Notification Service
(SNS) topic and an IAM rule." << std::endl;
    std::cout << "These resources will be created using a CloudFormation template."
<< std::endl;
    std::cout << "Stack creation may take a few minutes." << std::endl;

    askQuestion("Press Enter to continue: ", alwaysTrueTest);
    Aws::Map<Aws::String, Aws::String> outputs
=createCloudFormationStack(STACK_NAME,clientConfiguration);
    if (outputs.empty()) {
        std::cerr << "Exiting because createCloudFormationStack failed." <<
std::endl;
        cleanup(thingName, certificateARN, certificateID, "", "", false,
            clientConfiguration);
        return false;
    }

    // Retrieve the topic ARN and role ARN from the CloudFormation stack outputs.
    auto topicArnIter = outputs.find(SNS_TOPIC_ARN_OUTPUT);
    auto roleArnIter = outputs.find(ROLE_ARN_OUTPUT);
    if ((topicArnIter == outputs.end()) || (roleArnIter == outputs.end())) {
        std::cerr << "Exiting because output '" << SNS_TOPIC_ARN_OUTPUT <<
        "' or '" << ROLE_ARN_OUTPUT << "'not found in the CloudFormation stack." <<
std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME, "",

```

```

        false,
        clientConfiguration);
    return false;
}

Aws::String topicArn = topicArnIter->second;
Aws::String roleArn = roleArnIter->second;
Aws::String sqlStatement = "SELECT * FROM '";
sqlStatement += MQTT_MESSAGE_TOPIC_FILTER;
sqlStatement += "'";

printAsterisksLine();

std::cout << "Now a rule will be created.\n" << std::endl;
std::cout << "Rules are an administrator-level action. Any user who has
permission\n"
           << "to create rules will be able to access data processed by the
rule." << std::endl;
std::cout << "In this case, the rule will use an SNS topic" << std::endl;
std::cout << "and the following SQL statement '" << sqlStatement << "'." <<
std::endl;
std::cout << "For more information on IoT SQL, see https://docs.aws.amazon.com/iot/latest/developerguide/iot-sql-reference.html" << std::endl;
Aws::String ruleName = askQuestion("Enter a rule name: ");
if (!createTopicRule(ruleName, topicArn, sqlStatement, roleArn,
clientConfiguration)) {
    std::cerr << "Exiting because createRule failed." << std::endl;
    cleanup(thingName, certificateARN, certificateID, STACK_NAME, "",
        false,
        clientConfiguration);
    return false;
}

printAsterisksLine();

std::cout << "Now your rules will be listed.\n" << std::endl;
askQuestion("Press Enter to continue: ", alwaysTrueTest);
if (!listTopicRules(clientConfiguration)) {
    std::cerr << "Exiting because listRules failed." << std::endl;
    cleanup(thingName, certificateARN, certificateID, STACK_NAME, ruleName,
        false,
        clientConfiguration);
    return false;
}

```

```

printAsterisksLine();
Aws::String queryString = "thingName:" + thingName;
std::cout << "Now the AWS IoT fleet index will be queried with the query\n'"
<< queryString << "'.\n" << std::endl;
std::cout << "For query information, see https://docs.aws.amazon.com/iot/latest/
developerguide/query-syntax.html" << std::endl;

std::cout << "For this query to work, thing indexing must be enabled in your
account.\n"
<< "This can be done with the awscli command line by calling 'aws iot update-
indexing-configuration'\n"
<< "or it can be done programmatically." << std::endl;
std::cout << "For more information, see https://docs.aws.amazon.com/iot/latest/
developerguide/managing-index.html" << std::endl;
if (askYesNoQuestion("Do you want to enable thing indexing in your account? (y/
n) "))
{
    Aws::IoT::Model::ThingIndexingConfiguration thingIndexingConfiguration;

thingIndexingConfiguration.SetThingIndexingMode(Aws::IoT::Model::ThingIndexingMode::REGISTR

thingIndexingConfiguration.SetThingConnectivityIndexingMode(Aws::IoT::Model::ThingConnectiv
// The ThingGroupIndexingConfiguration object is ignored if not set.
    Aws::IoT::Model::ThingGroupIndexingConfiguration
thingGroupIndexingConfiguration;
    if (!updateIndexingConfiguration(thingIndexingConfiguration,
thingGroupIndexingConfiguration, clientConfiguration)) {
        std::cerr << "Exiting because updateIndexingConfiguration failed." <<
std::endl;
        cleanup(thingName, certificateARN, certificateID, STACK_NAME,
            ruleName, false,
            clientConfiguration);
        return false;
    }
}

if (!searchIndex(queryString, clientConfiguration)) {

    std::cerr << "Exiting because searchIndex failed." << std::endl;
    cleanup(thingName, certificateARN, certificateID, STACK_NAME, ruleName,
        false,
        clientConfiguration);
    return false;
}

```

```
}
```

```

//! Update an AWS IoT thing with attributes.
/*!
\param thingName: The name for the thing.
\param attributeMap: A map of key/value attributes/
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::IoT::updateThing(const Aws::String &thingName,
                              const std::map<Aws::String, Aws::String>
                              &attributeMap,
                              const Aws::Client::ClientConfiguration
                              &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::UpdateThingRequest request;
    request.SetThingName(thingName);
    Aws::IoT::Model::AttributePayload attributePayload;
    for (const auto &attribute: attributeMap) {
        attributePayload.AddAttributes(attribute.first, attribute.second);
    }
    request.SetAttributePayload(attributePayload);

    Aws::IoT::Model::UpdateThingOutcome outcome = iotClient.UpdateThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to update thing " << thingName << ":" <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Describe the endpoint specific to the AWS account making the call.
/*!
\param endpointResult: String to receive the endpoint result.
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::IoT::describeEndpoint(Aws::String &endpointResult,

```

```

        const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::String endpoint;
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DescribeEndpointRequest describeEndpointRequest;
    describeEndpointRequest.SetEndpointType(
        "iot:Data-ATS"); // Recommended endpoint type.

    Aws::IoT::Model::DescribeEndpointOutcome outcome = iotClient.DescribeEndpoint(
        describeEndpointRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully described endpoint." << std::endl;
        endpointResult = outcome.GetResult().GetEndpointAddress();
    }
    else {
        std::cerr << "Error describing endpoint" << outcome.GetError().GetMessage()
            << std::endl;
    }

    return outcome.IsSuccess();
}

//! List certificates registered in the AWS account making the call.
/*!
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */
bool AwsDoc::IoT::listCertificates(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListCertificatesRequest request;

    Aws::Vector<Aws::IoT::Model::Certificate> allCertificates;
    Aws::String marker; // Used to paginate results.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::IoT::Model::ListCertificatesOutcome outcome =
        iotClient.ListCertificates(
            request);
    } while (!outcome.IsSuccess());
}

```

```

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListCertificatesResult &result =
outcome.GetResult();
            marker = result.GetNextMarker();
            allCertificates.insert(allCertificates.end(),
                                result.GetCertificates().begin(),
                                result.GetCertificates().end());
        }
        else {
            std::cerr << "Error: " << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << allCertificates.size() << " certificate(s) found." << std::endl;

    for (auto &certificate: allCertificates) {
        std::cout << "Certificate ID: " << certificate.GetCertificateId() <<
std::endl;
        std::cout << "Certificate ARN: " << certificate.GetCertificateArn()
            << std::endl;
        std::cout << std::endl;
    }

    return true;
}

//! Update the shadow of an AWS IoT thing.
/*!
    \param thingName: The name for the thing.
    \param document: The state information, in JSON format.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::IoT::updateThingShadow(const Aws::String &thingName,
                                    const Aws::String &document,
                                    const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient iotDataPlaneClient(clientConfiguration);
    Aws::IoTDataPlane::Model::UpdateThingShadowRequest updateThingShadowRequest;
    updateThingShadowRequest.SetThingName(thingName);
    std::shared_ptr<std::stringstream> streamBuf =
std::make_shared<std::stringstream>(
    document);

```

```

    updateThingShadowRequest.SetBody(streamBuf);
    Aws::IoTDataPlane::Model::UpdateThingShadowOutcome outcome =
    iotDataPlaneClient.UpdateThingShadow(
        updateThingShadowRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing shadow." << std::endl;
    }
    else {
        std::cerr << "Error while updating thing shadow."
            << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Get the shadow of an AWS IoT thing.
/*!
    \param thingName: The name for the thing.
    \param documentResult: String to receive the state information, in JSON format.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::IoT::getThingShadow(const Aws::String &thingName,
    Aws::String &documentResult,
    const Aws::Client::ClientConfiguration
    &clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient iotClient(clientConfiguration);
    Aws::IoTDataPlane::Model::GetThingShadowRequest request;
    request.SetThingName(thingName);
    auto outcome = iotClient.GetThingShadow(request);
    if (outcome.IsSuccess()) {
        std::stringstream ss;
        ss << outcome.GetResult().GetPayload().rdbuf();
        documentResult = ss.str();
    }
    else {
        std::cerr << "Error getting thing shadow: " <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Create an AWS IoT rule with an SNS topic as the target.

```

```

/ * !
\ param ruleName: The name for the rule.
\ param snsTopic: The SNS topic ARN for the action.
\ param sql: The SQL statement used to query the topic.
\ param roleARN: The IAM role ARN for the action.
\ param clientConfiguration: AWS client configuration.
\ return bool: Function succeeded.
*/
bool
AwsDoc::IoT::createTopicRule(const Aws::String &ruleName,
                             const Aws::String &snsTopicARN, const Aws::String &sql,
                             const Aws::String &roleARN,
                             const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::CreateTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::SnsAction snsAction;
    snsAction.SetTargetArn(snsTopicARN);
    snsAction.SetRoleArn(roleARN);

    Aws::IoT::Model::Action action;
    action.SetSns(snsAction);

    Aws::IoT::Model::TopicRulePayload topicRulePayload;
    topicRulePayload.SetSql(sql);
    topicRulePayload.SetActions({action});

    request.SetTopicRulePayload(topicRulePayload);
    auto outcome = iotClient.CreateTopicRule(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created topic rule " << ruleName << "." <<
std::endl;
    }
    else {
        std::cerr << "Error creating topic rule " << ruleName << ": " <<
outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}

// ! Lists the AWS IoT topic rules.

```

```

/ * !
\ param clientConfiguration: AWS client configuration.
\ return bool: Function succeeded.
*/
bool AwsDoc::IoT::listTopicRules(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListTopicRulesRequest request;

    Aws::Vector<Aws::IoT::Model::TopicRuleListItem> allRules;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::ListTopicRulesOutcome outcome = iotClient.ListTopicRules(
            request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListTopicRulesResult &result =
outcome.GetResult();
            allRules.insert(allRules.end(),
                result.GetRules().cbegin(),
                result.GetRules().cend());

            nextToken = result.GetNextToken();
        }
        else {
            std::cerr << "ListTopicRules error: " <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    } while (!nextToken.empty());

    std::cout << "ListTopicRules: " << allRules.size() << " rule(s) found."
        << std::endl;
    for (auto &rule: allRules) {
        std::cout << " Rule name: " << rule.GetRuleName() << ", rule ARN: "
            << rule.GetRuleArn() << "." << std::endl;
    }

    return true;
}

```

```

}

//! Query the AWS IoT fleet index.
//! For query information, see https://docs.aws.amazon.com/iot/latest/developerguide/query-syntax.html
//!
\param query: The query string.
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::IoT::searchIndex(const Aws::String &query,
                              const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::SearchIndexRequest request;
    request.SetQueryString(query);

    Aws::Vector<Aws::IoT::Model::ThingDocument> allThingDocuments;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::SearchIndexOutcome outcome =
iotClient.SearchIndex(request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::SearchIndexResult &result = outcome.GetResult();
            allThingDocuments.insert(allThingDocuments.end(),
                                    result.GetThings().cbegin(),
                                    result.GetThings().cend());
            nextToken = result.GetNextToken();
        }
        else {
            std::cerr << "Error in SearchIndex: " << outcome.GetError().GetMessage()
                << std::endl;
            return false;
        }
    } while (!nextToken.empty());
}

```

```

    std::cout << allThingDocuments.size() << " thing document(s) found." <<
std::endl;
    for (const auto thingDocument: allThingDocuments) {
        std::cout << " Thing name: " << thingDocument.GetThingName() << "."
            << std::endl;
    }
    return true;
}

```

리소스를 정리합니다.

```

bool
AwsDoc::IoT::cleanup(const Aws::String &thingName, const Aws::String
    &certificateARN,
                    const Aws::String &certificateID, const Aws::String &stackName,
                    const Aws::String &ruleName, bool askForConfirmation,
                    const Aws::Client::ClientConfiguration &clientConfiguration) {
    bool result = true;

    if (!ruleName.empty() && (!askForConfirmation ||
        askYesNoQuestion("Delete the rule '" + ruleName +
            "'? (y/n) "))) {
        result &= deleteTopicRule(ruleName, clientConfiguration);
    }

    Aws::CloudFormation::CloudFormationClient
    cloudFormationClient(clientConfiguration);

    if (!stackName.empty() && (!askForConfirmation ||
        askYesNoQuestion(
            "Delete the CloudFormation stack '" +
stackName +
            "'? (y/n) "))) {
        result &= deleteStack(stackName, clientConfiguration);
    }

    if (!certificateARN.empty() && (!askForConfirmation ||
        askYesNoQuestion("Delete the certificate '" +
            certificateARN + "'? (y/n) ")))
    {
        result &= detachThingPrincipal(certificateARN, thingName,
            clientConfiguration);
    }
}

```

```

        result &= deleteCertificate(certificateID, clientConfiguration);
    }

    if (!thingName.empty() && (!askForConfirmation ||
                               askYesNoQuestion("Delete the thing '" + thingName +
                                                  "'? (y/n) "))) {
        result &= deleteThing(thingName, clientConfiguration);
    }

    return result;
}

```

```

//! Detach a principal from an AWS IoT thing.
/*!
 \param principal: A principal to detach.
 \param thingName: The name for the thing.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::IoT::detachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DetachThingPrincipalRequest detachThingPrincipalRequest;
    detachThingPrincipalRequest.SetThingName(thingName);
    detachThingPrincipalRequest.SetPrincipal(principal);

    Aws::IoT::Model::DetachThingPrincipalOutcome outcome =
    iotClient.DetachThingPrincipal(
        detachThingPrincipalRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully detached principal " << principal << " from thing "
        << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to detach principal " << principal << " from thing "
        << thingName << ": "
        << outcome.GetError().GetMessage() << std::endl;
    }
}

```

```

    }

    return outcome.IsSuccess();
}

//! Delete a certificate.
/*!
    \param certificateID: The ID of a certificate.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteCertificate(const Aws::String &certificateID,
                                    const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DeleteCertificateRequest request;
    request.SetCertificateId(certificateID);

    Aws::IoT::Model::DeleteCertificateOutcome outcome = iotClient.DeleteCertificate(
        request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted certificate " << certificateID <<
std::endl;
    }
    else {
        std::cerr << "Error deleting certificate " << certificateID << ": " <<
outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Delete an AWS IoT rule.
/*!
    \param ruleName: The name for the rule.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteTopicRule(const Aws::String &ruleName,
                                   const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

```

```

    Aws::IoT::Model::DeleteTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::DeleteTopicRuleOutcome outcome = iotClient.DeleteTopicRule(
        request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted rule " << ruleName << std::endl;
    }
    else {
        std::cerr << "Failed to delete rule " << ruleName <<
            ": " << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Delete an AWS IoT thing.
/*!
    \param thingName: The name for the thing.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
                              &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteThingRequest request;
    request.SetThingName(thingName);
    const auto outcome = iotClient.DeleteThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Error deleting thing " << thingName << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

## 작업

### AttachThingPrincipal

다음 코드 예시는 AttachThingPrincipal의 사용 방법을 보여줍니다.

SDK for C++

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Attach a principal to an AWS IoT thing.
/!*
 \param principal: A principal to attach.
 \param thingName: The name for the thing.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::IoT::attachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::AttachThingPrincipalRequest request;
    request.SetPrincipal(principal);
    request.SetThingName(thingName);
    Aws::IoT::Model::AttachThingPrincipalOutcome outcome =
client.AttachThingPrincipal(
    request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully attached principal to thing." << std::endl;
    }
    else {
        std::cerr << "Failed to attach principal to thing." <<
outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [AttachThingPrincipal](#)을 참조하세요.

## CreateKeysAndCertificate

다음 코드 예시는 CreateKeysAndCertificate의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Create keys and certificate for an Aws IoT device.
//! This routine will save certificates and keys to an output folder, if provided.
/*!
    \param outputFolder: Location for storing output in files, ignored when string is
    empty.
    \param certificateARNResult: A string to receive the ARN of the created
    certificate.
    \param certificateID: A string to receive the ID of the created certificate.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::IoT::createKeysAndCertificate(const Aws::String &outputFolder,
                                          Aws::String &certificateARNResult,
                                          Aws::String &certificateID,
                                          const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient client(clientConfiguration);
    Aws::IoT::Model::CreateKeysAndCertificateRequest
createKeysAndCertificateRequest;

    Aws::IoT::Model::CreateKeysAndCertificateOutcome outcome =
        client.CreateKeysAndCertificate(createKeysAndCertificateRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created a certificate and keys" << std::endl;
        certificateARNResult = outcome.GetResult().GetCertificateArn();
        certificateID = outcome.GetResult().GetCertificateId();
    }
}

```

```

std::cout << "Certificate ARN: " << certificateARNResult << ", certificate
ID: "
    << certificateID << std::endl;

if (!outputFolder.empty()) {
    std::cout << "Writing certificate and keys to the folder '" <<
outputFolder
    << "'." << std::endl;
    std::cout << "Be sure these files are stored securely." << std::endl;

    Aws::String certificateFilePath = outputFolder + "/certificate.pem.crt";
    std::ofstream certificateFile(certificateFilePath);
    if (!certificateFile.is_open()) {
        std::cerr << "Error opening certificate file, '" <<
certificateFilePath
        << "'."
        << std::endl;
        return false;
    }
    certificateFile << outcome.GetResult().GetCertificatePem();
    certificateFile.close();

    const Aws::IoT::Model::KeyPair &keyPair =
outcome.GetResult().GetKeyPair();

    Aws::String privateKeyFilePath = outputFolder + "/private.pem.key";
    std::ofstream privateKeyFile(privateKeyFilePath);
    if (!privateKeyFile.is_open()) {
        std::cerr << "Error opening private key file, '" <<
privateKeyFilePath
        << "'."
        << std::endl;
        return false;
    }
    privateKeyFile << keyPair.GetPrivateKey();
    privateKeyFile.close();

    Aws::String publicKeyFilePath = outputFolder + "/public.pem.key";
    std::ofstream publicKeyFile(publicKeyFilePath);
    if (!publicKeyFile.is_open()) {
        std::cerr << "Error opening public key file, '" << publicKeyFilePath
        << "'."
        << std::endl;
        return false;
    }

```

```

        }
        publicKeyFile << keyPair.GetPublicKey();
    }
}
else {
    std::cerr << "Error creating keys and certificate: "
               << outcome.GetError().GetMessage() << std::endl;
}

return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조에서 [CreateKeysAndCertificate](#)를 참조하세요.

## CreateThing

다음 코드 예시는 CreateThing의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Create an AWS IoT thing.
/*!
    \param thingName: The name for the thing.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::IoT::createThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
                              &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::CreateThingRequest createThingRequest;
    createThingRequest.SetThingName(thingName);

    Aws::IoT::Model::CreateThingOutcome outcome = iotClient.CreateThing(
        createThingRequest);
}

```

```

    if (outcome.IsSuccess()) {
        std::cout << "Successfully created thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to create thing " << thingName << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CreateThing](#)을 참조하세요.

## CreateTopicRule

다음 코드 예시는 CreateTopicRule의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

/*! Create an AWS IoT rule with an SNS topic as the target.
 *!
 * \param ruleName: The name for the rule.
 * \param snsTopic: The SNS topic ARN for the action.
 * \param sql: The SQL statement used to query the topic.
 * \param roleARN: The IAM role ARN for the action.
 * \param clientConfiguration: AWS client configuration.
 * \return bool: Function succeeded.
 */
bool
AwsDoc::IoT::createTopicRule(const Aws::String &ruleName,
                             const Aws::String &snsTopicARN, const Aws::String &sql,
                             const Aws::String &roleARN,
                             const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

```

```

    Aws::IoT::Model::CreateTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::SnsAction snsAction;
    snsAction.SetTargetArn(snsTopicARN);
    snsAction.SetRoleArn(roleARN);

    Aws::IoT::Model::Action action;
    action.SetSns(snsAction);

    Aws::IoT::Model::TopicRulePayload topicRulePayload;
    topicRulePayload.SetSql(sql);
    topicRulePayload.SetActions({action});

    request.SetTopicRulePayload(topicRulePayload);
    auto outcome = iotClient.CreateTopicRule(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully created topic rule " << ruleName << "." <<
std::endl;
    }
    else {
        std::cerr << "Error creating topic rule " << ruleName << ": " <<
        outcome.GetError().GetMessage() << std::endl;
    }
    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CreateTopicRule](#)을 참조하세요.

## DeleteCertificate

다음 코드 예시는 DeleteCertificate의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Delete a certificate.
/*!
    \param certificateID: The ID of a certificate.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::IoT::deleteCertificate(const Aws::String &certificateID,
                                   const Aws::Client::ClientConfiguration
                                   &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DeleteCertificateRequest request;
    request.SetCertificateId(certificateID);

    Aws::IoT::Model::DeleteCertificateOutcome outcome = iotClient.DeleteCertificate(
        request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted certificate " << certificateID <<
std::endl;
    }
    else {
        std::cerr << "Error deleting certificate " << certificateID << ": " <<
outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteCertificate](#)를 참조하세요.

## DeleteThing

다음 코드 예시는 DeleteThing의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Delete an AWS IoT thing.
/*!
    \param thingName: The name for the thing.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::IoT::deleteThing(const Aws::String &thingName,
                              const Aws::Client::ClientConfiguration
                              &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteThingRequest request;
    request.SetThingName(thingName);
    const auto outcome = iotClient.DeleteThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Error deleting thing " << thingName << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteThing](#)을 참조하세요.

**DeleteTopicRule**

다음 코드 예시는 DeleteTopicRule의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Delete an AWS IoT rule.
/*!
 \param ruleName: The name for the rule.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
 */
bool AwsDoc::IoT::deleteTopicRule(const Aws::String &ruleName,
                                   const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DeleteTopicRuleRequest request;
    request.SetRuleName(ruleName);

    Aws::IoT::Model::DeleteTopicRuleOutcome outcome = iotClient.DeleteTopicRule(
        request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted rule " << ruleName << std::endl;
    }
    else {
        std::cerr << "Failed to delete rule " << ruleName <<
            ": " << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteTopicRule](#)를 참조하세요.

## DescribeEndpoint

다음 코드 예시는 DescribeEndpoint의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Describe the endpoint specific to the AWS account making the call.
/*!
    \param endpointResult: String to receive the endpoint result.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::IoT::describeEndpoint(Aws::String &endpointResult,
                                   const Aws::Client::ClientConfiguration
                                   &clientConfiguration) {
    Aws::String endpoint;
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::DescribeEndpointRequest describeEndpointRequest;
    describeEndpointRequest.SetEndpointType(
        "iot:Data-ATS"); // Recommended endpoint type.

    Aws::IoT::Model::DescribeEndpointOutcome outcome = iotClient.DescribeEndpoint(
        describeEndpointRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully described endpoint." << std::endl;
        endpointResult = outcome.GetResult().GetEndpointAddress();
    }
    else {
        std::cerr << "Error describing endpoint" << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DescribeEndpoint](#)를 참조하세요.

## DescribeThing

다음 코드 예시는 DescribeThing의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Describe an AWS IoT thing.
/*!
    \param thingName: The name for the thing.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
 */
bool AwsDoc::IoT::describeThing(const Aws::String &thingName,
                                const Aws::Client::ClientConfiguration
                                &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DescribeThingRequest request;
    request.SetThingName(thingName);

    Aws::IoT::Model::DescribeThingOutcome outcome =
    iotClient.DescribeThing(request);

    if (outcome.IsSuccess()) {
        const Aws::IoT::Model::DescribeThingResult &result = outcome.GetResult();
        std::cout << "Retrieved thing '" << result.GetThingName() << "' <<
std::endl;
        std::cout << "thingArn: " << result.GetThingArn() << std::endl;
        std::cout << result.GetAttributes().size() << " attribute(s) retrieved"
        << std::endl;
        for (const auto &attribute: result.GetAttributes()) {
            std::cout << " attribute: " << attribute.first << "=" <<
attribute.second
            << std::endl;
        }
    }
    else {

```

```

        std::cerr << "Error describing thing " << thingName << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DescribeThing](#)을 참조하세요.

## DetachThingPrincipal

다음 코드 예시는 DetachThingPrincipal의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Detach a principal from an AWS IoT thing.
/*!
    \param principal: A principal to detach.
    \param thingName: The name for the thing.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::IoT::detachThingPrincipal(const Aws::String &principal,
                                       const Aws::String &thingName,
                                       const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::DetachThingPrincipalRequest detachThingPrincipalRequest;
    detachThingPrincipalRequest.SetThingName(thingName);
    detachThingPrincipalRequest.SetPrincipal(principal);

    Aws::IoT::Model::DetachThingPrincipalOutcome outcome =
    iotClient.DetachThingPrincipal(
        detachThingPrincipalRequest);

```

```

    if (outcome.IsSuccess()) {
        std::cout << "Successfully detached principal " << principal << " from thing "
        << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to detach principal " << principal << " from thing "
        << thingName << ": "
        << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DetachThingPrincipal](#)을 참조하세요.

## ListCertificates

다음 코드 예시는 ListCertificates의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! List certificates registered in the AWS account making the call.
/*!
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::IoT::listCertificates(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::ListCertificatesRequest request;

    Aws::Vector<Aws::IoT::Model::Certificate> allCertificates;

```

```

    Aws::String marker; // Used to paginate results.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::IoT::Model::ListCertificatesOutcome outcome =
        iotClient.ListCertificates(
            request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::ListCertificatesResult &result =
            outcome.GetResult();
            marker = result.GetNextMarker();
            allCertificates.insert(allCertificates.end(),
                                  result.GetCertificates().begin(),
                                  result.GetCertificates().end());
        }
        else {
            std::cerr << "Error: " << outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    } while (!marker.empty());

    std::cout << allCertificates.size() << " certificate(s) found." << std::endl;

    for (auto &certificate: allCertificates) {
        std::cout << "Certificate ID: " << certificate.GetCertificateId() <<
        std::endl;
        std::cout << "Certificate ARN: " << certificate.GetCertificateArn()
        << std::endl;
        std::cout << std::endl;
    }

    return true;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ListCertificates](#)를 참조하세요.

## SearchIndex

다음 코드 예시는 SearchIndex의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Query the AWS IoT fleet index.
//! For query information, see https://docs.aws.amazon.com/iot/latest/
//! developerguide/query-syntax.html
//!
//! \param query: The query string.
//! \param clientConfiguration: AWS client configuration.
//! \return bool: Function succeeded.
//!
bool AwsDoc::IoT::searchIndex(const Aws::String &query,
                              const Aws::Client::ClientConfiguration
                              &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::SearchIndexRequest request;
    request.SetQueryString(query);

    Aws::Vector<Aws::IoT::Model::ThingDocument> allThingDocuments;
    Aws::String nextToken; // Used for pagination.
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        Aws::IoT::Model::SearchIndexOutcome outcome =
        iotClient.SearchIndex(request);

        if (outcome.IsSuccess()) {
            const Aws::IoT::Model::SearchIndexResult &result = outcome.GetResult();
            allThingDocuments.insert(allThingDocuments.end(),
                                    result.GetThings().cbegin(),
                                    result.GetThings().cend());
            nextToken = result.GetNextToken();
        }
    }
}

```

```

        else {
            std::cerr << "Error in SearchIndex: " << outcome.GetError().GetMessage()
                << std::endl;
            return false;
        }
    } while (!nextToken.empty());

    std::cout << allThingDocuments.size() << " thing document(s) found." <<
std::endl;
    for (const auto thingDocument: allThingDocuments) {
        std::cout << "  Thing name: " << thingDocument.GetThingName() << "."
            << std::endl;
    }
    return true;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [SearchIndex](#)를 참조하세요.

## UpdateIndexingConfiguration

다음 코드 예시는 UpdateIndexingConfiguration의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Update the indexing configuration.
/*!
    \param thingIndexingConfiguration: A ThingIndexingConfiguration object which is
    ignored if not set.
    \param thingGroupIndexingConfiguration: A ThingGroupIndexingConfiguration object
    which is ignored if not set.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::IoT::updateIndexingConfiguration(

```

```

        const Aws::IoT::Model::ThingIndexingConfiguration
&thingIndexingConfiguration,
        const Aws::IoT::Model::ThingGroupIndexingConfiguration
&thingGroupIndexingConfiguration,
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);

    Aws::IoT::Model::UpdateIndexingConfigurationRequest request;

    if (thingIndexingConfiguration.ThingIndexingModeHasBeenSet()) {
        request.SetThingIndexingConfiguration(thingIndexingConfiguration);
    }

    if (thingGroupIndexingConfiguration.ThingGroupIndexingModeHasBeenSet()) {
        request.SetThingGroupIndexingConfiguration(thingGroupIndexingConfiguration);
    }

    Aws::IoT::Model::UpdateIndexingConfigurationOutcome outcome =
    iotClient.UpdateIndexingConfiguration(
        request);

    if (outcome.IsSuccess()) {
        std::cout << "UpdateIndexingConfiguration succeeded." << std::endl;
    }
    else {
        std::cerr << "UpdateIndexingConfiguration failed."
            << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [UpdateIndexingConfiguration](#)을 참조하세요.

## UpdateThing

다음 코드 예시는 UpdateThing의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Update an AWS IoT thing with attributes.
/*!
    \param thingName: The name for the thing.
    \param attributeMap: A map of key/value attributes/
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::IoT::updateThing(const Aws::String &thingName,
                              const std::map<Aws::String, Aws::String>
                              &attributeMap,
                              const Aws::Client::ClientConfiguration
                              &clientConfiguration) {
    Aws::IoT::IoTClient iotClient(clientConfiguration);
    Aws::IoT::Model::UpdateThingRequest request;
    request.SetThingName(thingName);
    Aws::IoT::Model::AttributePayload attributePayload;
    for (const auto &attribute: attributeMap) {
        attributePayload.AddAttributes(attribute.first, attribute.second);
    }
    request.SetAttributePayload(attributePayload);

    Aws::IoT::Model::UpdateThingOutcome outcome = iotClient.UpdateThing(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing " << thingName << std::endl;
    }
    else {
        std::cerr << "Failed to update thing " << thingName << ":" <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [UpdateThing](#)을 참조하세요.

## AWS IoT data SDK for C++를 사용한 예제

다음 코드 예제에서는와 AWS SDK for C++ 함께를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 AWS IoT data.

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

#### GetThingShadow

다음 코드 예시는 GetThingShadow의 사용 방법을 보여줍니다.

SDK for C++

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
//! Get the shadow of an AWS IoT thing.
/*!
    \param thingName: The name for the thing.
    \param documentResult: String to receive the state information, in JSON format.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
```

```
bool AwsDoc::IoT::getThingShadow(const Aws::String &thingName,
                                Aws::String &documentResult,
                                const Aws::Client::ClientConfiguration
                                &clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient iotClient(clientConfiguration);
    Aws::IoTDataPlane::Model::GetThingShadowRequest request;
    request.SetThingName(thingName);
    auto outcome = iotClient.GetThingShadow(request);
    if (outcome.IsSuccess()) {
        std::stringstream ss;
        ss << outcome.GetResult().GetPayload().rdbuf();
        documentResult = ss.str();
    }
    else {
        std::cerr << "Error getting thing shadow: " <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetThingShadow](#)를 참조하세요.

## UpdateThingShadow

다음 코드 예시는 UpdateThingShadow의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
///  
/*!  
    \param thingName: The name for the thing.  
    \param document: The state information, in JSON format.  
    \param clientConfiguration: AWS client configuration.
```

```

    \return bool: Function succeeded.
    */
bool AwsDoc::IoT::updateThingShadow(const Aws::String &thingName,
                                     const Aws::String &document,
                                     const Aws::Client::ClientConfiguration
                                     &clientConfiguration) {
    Aws::IoTDataPlane::IoTDataPlaneClient iotDataPlaneClient(clientConfiguration);
    Aws::IoTDataPlane::Model::UpdateThingShadowRequest updateThingShadowRequest;
    updateThingShadowRequest.SetThingName(thingName);
    std::shared_ptr<std::stringstream> streamBuf =
std::make_shared<std::stringstream>(
    document);
    updateThingShadowRequest.SetBody(streamBuf);
    Aws::IoTDataPlane::Model::UpdateThingShadowOutcome outcome =
iotDataPlaneClient.UpdateThingShadow(
    updateThingShadowRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated thing shadow." << std::endl;
    }
    else {
        std::cerr << "Error while updating thing shadow."
            << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [UpdateThingShadow](#)를 참조하세요.

## SDK for C++를 사용한 Lambda 예제

다음 코드 예제에서는 Lambda와 AWS SDK for C++ 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](#)
- [기본 사항](#)
- [작업](#)
- [시나리오](#)

## 시작하기

### Hello Lambda

다음 코드 예제에서는 Lambda 사용을 시작하는 방법을 보여줍니다.

### SDK for C++

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

CMakeLists.txt CMake 파일의 코드입니다.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS lambda)

# Set this project's name.
project("hello_lambda")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
```

```

set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
"${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory for
running and debugging.

    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you may
need to uncomment this

                                # and set the proper subdirectory to the
executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
"${CMAKE_CURRENT_BINARY_DIR}/${BIN_SUB_DIR}")
endif ()

add_executable(${PROJECT_NAME}
    hello_lambda.cpp)

target_link_libraries(${PROJECT_NAME}
    ${AWSSDK_LINK_LIBRARIES})

```

hello\_lambda.cpp 소스 파일의 코드입니다.

```

#include <aws/core/Aws.h>
#include <aws/lambda/LambdaClient.h>
#include <aws/lambda/model/ListFunctionsRequest.h>
#include <iostream>

/*
 * A "Hello Lambda" starter application which initializes an AWS Lambda (Lambda)
client and lists the Lambda functions.
 */

```

```

*   main function
*
*   Usage: 'hello_lambda'
*
*/

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::Lambda::LambdaClient lambdaClient(clientConfig);
        std::vector<Aws::String> functions;
        Aws::String marker; // Used for pagination.

        do {
            Aws::Lambda::Model::ListFunctionsRequest request;
            if (!marker.empty()) {
                request.SetMarker(marker);
            }

            Aws::Lambda::Model::ListFunctionsOutcome outcome =
lambdaClient.ListFunctions(
                request);

            if (outcome.IsSuccess()) {
                const Aws::Lambda::Model::ListFunctionsResult &listFunctionsResult =
outcome.GetResult();
                std::cout << listFunctionsResult.GetFunctions().size()
                    << " lambda functions were retrieved." << std::endl;

                for (const Aws::Lambda::Model::FunctionConfiguration
&functionConfiguration: listFunctionsResult.GetFunctions()) {
                    functions.push_back(functionConfiguration.GetFunctionName());
                    std::cout << functions.size() << " "
                        << functionConfiguration.GetDescription() <<
std::endl;
                    std::cout << " "

```

```

        <<
        Aws::Lambda::Model::RuntimeMapper::GetNameForRuntime(
            functionConfiguration.GetRuntime()) << ": "
        << functionConfiguration.GetHandler()
        << std::endl;
    }
    marker = listFunctionsResult.GetNextMarker();
} else {
    std::cerr << "Error with Lambda::ListFunctions. "
    << outcome.GetError().GetMessage()
    << std::endl;
    result = 1;
    break;
}
} while (!marker.empty());
}

Aws::ShutdownAPI(options); // Should only be called once.
return result;
}

```

- API 세부 정보는 [AWS SDK for C++ API 참조](#)의 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 생성](#)을 참조하세요.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예제 리포지토리](#)에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Get started with functions scenario.
/*!
\param clientConfig: AWS client configuration.
\return bool: Successful completion.
*/
bool AwsDoc::Lambda::getStartedWithFunctionsScenario(
    const Aws::Client::ClientConfiguration &clientConfig) {

    Aws::Lambda::LambdaClient client(clientConfig);

    // 1. Create an AWS Identity and Access Management (IAM) role for Lambda
    function.
    Aws::String roleArn;
    if (!getIamRoleArn(roleArn, clientConfig)) {
        return false;
    }

    // 2. Create a Lambda function.
    int seconds = 0;
    do {
        Aws::Lambda::Model::CreateFunctionRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        request.SetDescription(LAMBDA_DESCRIPTION); // Optional.
#ifdef USE_CPP_LAMBDA_FUNCTION
        request.SetRuntime(Aws::Lambda::Model::Runtime::provided_al2);
        request.SetTimeout(15);
        request.SetMemorySize(128);

        // Assume the AWS Lambda function was built in Docker with same architecture
        // as this code.
#ifdef __x86_64__
        request.SetArchitectures({Aws::Lambda::Model::Architecture::x86_64});
#elif defined(__aarch64__)
        request.SetArchitectures({Aws::Lambda::Model::Architecture::arm64});

```

```

#else
#error "Unimplemented architecture"
#endif // defined(architecture)
#else
    request.SetRuntime(Aws::Lambda::Model::Runtime::python3_9);
#endif

    request.SetRole(roleArn);
    request.SetHandler(LAMBDA_HANDLER_NAME);
    request.SetPublish(true);
    Aws::Lambda::Model::FunctionCode code;
    std::ifstream ifstream(INCREMENT_LAMBDA_CODE.c_str(),
                           std::ios_base::in | std::ios_base::binary);
    if (!ifstream.is_open()) {
        std::cerr << "Error opening file " << INCREMENT_LAMBDA_CODE << "." <<
std::endl;

#if USE_CPP_LAMBDA_FUNCTION
        std::cerr
            << "The cpp Lambda function must be built following the
instructions in the cpp_lambda/README.md file. "
            << std::endl;
#endif

        deleteIamRole(clientConfig);
        return false;
    }

    Aws::StringStream buffer;
    buffer << ifstream.rdbuf();

    code.SetZipFile(Aws::Utils::ByteBuffer((unsigned char *)
buffer.str().c_str(),
                                           buffer.str().length()));
    request.SetCode(code);

    Aws::Lambda::Model::CreateFunctionOutcome outcome = client.CreateFunction(
        request);

    if (outcome.IsSuccess()) {
        std::cout << "The lambda function was successfully created. " << seconds
            << " seconds elapsed." << std::endl;
        break;
    }
    else if (outcome.GetError().GetErrorType() ==
        Aws::Lambda::LambdaErrors::INVALID_PARAMETER_VALUE &&

```

```

        outcome.GetError().GetMessage().find("role") >= 0) {
    if ((seconds % 5) == 0) { // Log status every 10 seconds.
        std::cout
            << "Waiting for the IAM role to become available as a
CreateFunction parameter. "
            << seconds
            << " seconds elapsed." << std::endl;

        std::cout << outcome.GetError().GetMessage() << std::endl;
    }
}
else {
    std::cerr << "Error with CreateFunction. "
        << outcome.GetError().GetMessage()
        << std::endl;
    deleteIamRole(clientConfig);
    return false;
}
++seconds;
std::this_thread::sleep_for(std::chrono::seconds(1));
} while (60 > seconds);

std::cout << "The current Lambda function increments 1 by an input." <<
std::endl;

// 3. Invoke the Lambda function.
{
    int increment = askQuestionForInt("Enter an increment integer: ");

    Aws::Lambda::Model::InvokeResult invokeResult;
    Aws::Utils::Json::JsonValue jsonPayload;
    jsonPayload.WithString("action", "increment");
    jsonPayload.WithInteger("number", increment);
    if (invokeLambdaFunction(jsonPayload, Aws::Lambda::Model::LogType::Tail,
        invokeResult, client)) {
        Aws::Utils::Json::JsonValue jsonValue(invokeResult.GetPayload());
        Aws::Map<Aws::String, Aws::Utils::Json::JsonValue> values =
            jsonValue.View().GetAllObjects();
        auto iter = values.find("result");
        if (iter != values.end() && iter->second.IsIntegerType()) {
            {
                std::cout << INCREMENT_RESULT_PREFIX
                    << iter->second.AsInteger() << std::endl;
            }
        }
    }
}

```

```

    }
    else {
        std::cout << "There was an error in execution. Here is the log."
            << std::endl;
        Aws::Utils::ByteBuffer buffer =
        Aws::Utils::HashingUtils::Base64Decode(
            invokeResult.GetLogResult());
        std::cout << "With log " << buffer.GetUnderlyingData() << std::endl;
    }
}

std::cout
    << "The Lambda function will now be updated with new code. Press return
to continue, ";
    Aws::String answer;
    std::getline(std::cin, answer);

// 4. Update the Lambda function code.
{
    Aws::Lambda::Model::UpdateFunctionCodeRequest request;
    request.SetFunctionName(LAMBDA_NAME);
    std::ifstream ifstream(CALCULATOR_LAMBDA_CODE.c_str(),
        std::ios_base::in | std::ios_base::binary);
    if (!ifstream.is_open()) {
        std::cerr << "Error opening file " << INCREMENT_LAMBDA_CODE << "." <<
std::endl;

#ifdef USE_CPP_LAMBDA_FUNCTION
        std::cerr
            << "The cpp Lambda function must be built following the
instructions in the cpp_lambda/README.md file. "
            << std::endl;
#endif

        deleteLambdaFunction(client);
        deleteIamRole(clientConfig);
        return false;
    }

    Aws::StringStream buffer;
    buffer << ifstream.rdbuf();
    request.SetZipFile(
        Aws::Utils::ByteBuffer((unsigned char *) buffer.str().c_str(),
            buffer.str().length()));

```

```

        request.SetPublish(true);

        Aws::Lambda::Model::UpdateFunctionCodeOutcome outcome =
client.UpdateFunctionCode(
            request);

        if (outcome.IsSuccess()) {
            std::cout << "The lambda code was successfully updated." << std::endl;
        }
        else {
            std::cerr << "Error with Lambda::UpdateFunctionCode. "
                << outcome.GetError().GetMessage()
                << std::endl;
        }
    }

    std::cout
        << "This function uses an environment variable to control the logging
level."
        << std::endl;
    std::cout
        << "UpdateFunctionConfiguration will be used to set the LOG_LEVEL to
DEBUG."
        << std::endl;
    seconds = 0;

    // 5. Update the Lambda function configuration.
    do {
        ++seconds;
        std::this_thread::sleep_for(std::chrono::seconds(1));
        Aws::Lambda::Model::UpdateFunctionConfigurationRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        Aws::Lambda::Model::Environment environment;
        environment.AddVariables("LOG_LEVEL", "DEBUG");
        request.SetEnvironment(environment);

        Aws::Lambda::Model::UpdateFunctionConfigurationOutcome outcome =
client.UpdateFunctionConfiguration(
            request);

        if (outcome.IsSuccess()) {
            std::cout << "The lambda configuration was successfully updated."
                << std::endl;
            break;
        }
    }

```

```

    }

    // RESOURCE_IN_USE: function code update not completed.
    else if (outcome.GetError().GetErrorType() !=
             Aws::Lambda::LambdaErrors::RESOURCE_IN_USE) {
        if ((seconds % 10) == 0) { // Log status every 10 seconds.
            std::cout << "Lambda function update in progress . After " <<
seconds
                        << " seconds elapsed." << std::endl;
        }
    }
    else {
        std::cerr << "Error with Lambda::UpdateFunctionConfiguration. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

} while (0 < seconds);

if (0 > seconds) {
    std::cerr << "Function failed to become active." << std::endl;
}
else {
    std::cout << "Updated function active after " << seconds << " seconds."
              << std::endl;
}

std::cout
    << "\nThe new code applies an arithmetic operator to two variables, x an
y."
    << std::endl;
std::vector<Aws::String> operators = {"plus", "minus", "times", "divided-by"};
for (size_t i = 0; i < operators.size(); ++i) {
    std::cout << "    " << i + 1 << " " << operators[i] << std::endl;
}

// 6. Invoke the updated Lambda function.
do {
    int operatorIndex = askQuestionForIntRange("Select an operator index 1 - 4
", 1,
                                              4);

    int x = askQuestionForInt("Enter an integer for the x value ");
    int y = askQuestionForInt("Enter an integer for the y value ");

```

```

    Aws::Utils::Json::JsonValue calculateJsonPayload;
    calculateJsonPayload.WithString("action", operators[operatorIndex - 1]);
    calculateJsonPayload.WithInteger("x", x);
    calculateJsonPayload.WithInteger("y", y);
    Aws::Lambda::Model::InvokeResult calculatedResult;
    if (invokeLambdaFunction(calculateJsonPayload,
                            Aws::Lambda::Model::LogType::Tail,
                            calculatedResult, client)) {
        Aws::Utils::Json::JsonValue jsonValue(calculatedResult.GetPayload());
        Aws::Map<Aws::String, Aws::Utils::Json::JsonView> values =
            jsonValue.View().GetAllObjects();
        auto iter = values.find("result");
        if (iter != values.end() && iter->second.IsIntegerType()) {
            std::cout << ARITHMETIC_RESULT_PREFIX << x << " "
                      << operators[operatorIndex - 1] << " "
                      << y << " is " << iter->second.AsInteger() << std::endl;
        }
        else if (iter != values.end() && iter->second.IsFloatingPointType()) {
            std::cout << ARITHMETIC_RESULT_PREFIX << x << " "
                      << operators[operatorIndex - 1] << " "
                      << y << " is " << iter->second.AsDouble() << std::endl;
        }
        else {
            std::cout << "There was an error in execution. Here is the log."
                      << std::endl;
            Aws::Utils::ByteBuffer buffer =
                Aws::Utils::HashingUtils::Base64Decode(
                    calculatedResult.GetLogResult());
            std::cout << "With log " << buffer.GetUnderlyingData() << std::endl;
        }
    }

    answer = askQuestion("Would you like to try another operation? (y/n) ");
} while (answer == "y");

std::cout
    << "A list of the lambda functions will be retrieved. Press return to
continue, ";
std::getline(std::cin, answer);

// 7. List the Lambda functions.

std::vector<Aws::String> functions;
Aws::String marker;

```

```

do {
    Aws::Lambda::Model::ListFunctionsRequest request;
    if (!marker.empty()) {
        request.SetMarker(marker);
    }

    Aws::Lambda::Model::ListFunctionsOutcome outcome = client.ListFunctions(
        request);

    if (outcome.IsSuccess()) {
        const Aws::Lambda::Model::ListFunctionsResult &result =
outcome.GetResult();
        std::cout << result.GetFunctions().size()
            << " lambda functions were retrieved." << std::endl;

        for (const Aws::Lambda::Model::FunctionConfiguration
&functionConfiguration: result.GetFunctions()) {
            functions.push_back(functionConfiguration.GetFunctionName());
            std::cout << functions.size() << " "
                << functionConfiguration.GetDescription() << std::endl;
            std::cout << " "
                << Aws::Lambda::Model::RuntimeMapper::GetNameForRuntime(
                    functionConfiguration.GetRuntime()) << ": "
                << functionConfiguration.GetHandler()
                << std::endl;
        }
        marker = result.GetNextMarker();
    }
    else {
        std::cerr << "Error with Lambda::ListFunctions. "
            << outcome.GetError().GetMessage()
            << std::endl;
    }
} while (!marker.empty());

// 8. Get a Lambda function.
if (!functions.empty()) {
    std::stringstream question;
    question << "Choose a function to retrieve between 1 and " <<
functions.size()
        << " ";
    int functionIndex = askQuestionForIntRange(question.str(), 1,

```

```

static_cast<int>(functions.size()));

    Aws::String functionName = functions[functionIndex - 1];

    Aws::Lambda::Model::GetFunctionRequest request;
    request.SetFunctionName(functionName);

    Aws::Lambda::Model::GetFunctionOutcome outcome =
    client.GetFunction(request);

    if (outcome.IsSuccess()) {
        std::cout << "Function retrieve.\n" <<

outcome.GetResult().GetConfiguration().Jsonize().View().WriteReadable()
        << std::endl;
    }
    else {
        std::cerr << "Error with Lambda::GetFunction. "
        << outcome.GetError().GetMessage()
        << std::endl;
    }
}

std::cout << "The resources will be deleted. Press return to continue, ";
std::getline(std::cin, answer);

// 9. Delete the Lambda function.
bool result = deleteLambdaFunction(client);

// 10. Delete the IAM role.
return result && deleteIamRole(clientConfig);
}

//! Routine which invokes a Lambda function and returns the result.
/*!
 \param jsonPayload: Payload for invoke function.
 \param logType: Log type setting for invoke function.
 \param invokeResult: InvokeResult object to receive the result.
 \param client: Lambda client.
 \return bool: Successful completion.
 */
bool
AwsDoc::Lambda::invokeLambdaFunction(const Aws::Utils::Json::JsonValue &jsonPayload,

```

```

        Aws::Lambda::Model::LogType logType,
        Aws::Lambda::Model::InvokeResult &invokeResult,
        const Aws::Lambda::LambdaClient &client) {

    int seconds = 0;
    bool result = false;
    /*
     * In this example, the Invoke function can be called before recently created
resources are
     * available. The Invoke function is called repeatedly until the resources are
     * available.
     */
    do {
        Aws::Lambda::Model::InvokeRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        request.SetLogType(logType);
        std::shared_ptr<Aws::IOStream> payload = Aws::MakeShared<Aws::StringStream>(
            "FunctionTest");
        *payload << jsonPayload.View().WriteReadable();
        request.SetBody(payload);
        request.SetContentType("application/json");
        Aws::Lambda::Model::InvokeOutcome outcome = client.Invoke(request);

        if (outcome.IsSuccess()) {
            invokeResult = std::move(outcome.GetResult());
            result = true;
            break;
        }

        // ACCESS_DENIED: because the role is not available yet.
        // RESOURCE_CONFLICT: because the Lambda function is being created or
updated.
        else if ((outcome.GetError().GetErrorType() ==
            Aws::Lambda::LambdaErrors::ACCESS_DENIED) ||
            (outcome.GetError().GetErrorType() ==
            Aws::Lambda::LambdaErrors::RESOURCE_CONFLICT)) {
            if ((seconds % 5) == 0) { // Log status every 10 seconds.
                std::cout << "Waiting for the invoke api to be available, status "
<<
                    ((outcome.GetError().GetErrorType() ==
                        Aws::Lambda::LambdaErrors::ACCESS_DENIED ?
                        "ACCESS_DENIED" : "RESOURCE_CONFLICT")) << ". " <<
seconds
                    << " seconds elapsed." << std::endl;
            }
        }
    }
}

```

```
    }  
    else {  
        std::cerr << "Error with Lambda::InvokeRequest. "  
                   << outcome.GetError().GetMessage()  
                   << std::endl;  
        break;  
    }  
    ++seconds;  
    std::this_thread::sleep_for(std::chrono::seconds(1));  
} while (seconds < 60);  
  
return result;  
}
```

• API 세부 정보는 AWS SDK for C++ API 참조의 다음 항목을 참조하세요.

- [CreateFunction](#)
- [DeleteFunction](#)
- [GetFunction](#)
- [간접 호출](#)
- [ListFunctions](#)
- [UpdateFunctionCode](#)
- [UpdateFunctionConfiguration](#)

## 작업

### CreateFunction

다음 코드 예시는 CreateFunction의 사용 방법을 보여줍니다.

SDK for C++

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region in which the bucket was created
    (overrides config file).
    // clientConfig.region = "us-east-1";

    Aws::Lambda::LambdaClient client(clientConfig);

    Aws::Lambda::Model::CreateFunctionRequest request;
    request.SetFunctionName(LAMBDA_NAME);
    request.SetDescription(LAMBDA_DESCRIPTION); // Optional.
#ifdef USE_CPP_LAMBDA_FUNCTION
    request.SetRuntime(Aws::Lambda::Model::Runtime::provided_al2);
    request.SetTimeout(15);
    request.SetMemorySize(128);

    // Assume the AWS Lambda function was built in Docker with same architecture
    // as this code.
#ifdef __x86_64__
    request.SetArchitectures({Aws::Lambda::Model::Architecture::x86_64});
#elif defined(__aarch64__)
    request.SetArchitectures({Aws::Lambda::Model::Architecture::arm64});
#else
#error "Unimplemented architecture"
#endif // defined(architecture)
#else
    request.SetRuntime(Aws::Lambda::Model::Runtime::python3_9);
#endif

    request.SetRole(roleArn);
    request.SetHandler(LAMBDA_HANDLER_NAME);
    request.SetPublish(true);
    Aws::Lambda::Model::FunctionCode code;
    std::ifstream ifstream(INCREMENT_LAMBDA_CODE.c_str(),
                           std::ios_base::in | std::ios_base::binary);
    if (!ifstream.is_open()) {
        std::cerr << "Error opening file " << INCREMENT_LAMBDA_CODE << "." <<
        std::endl;
    }

#ifdef USE_CPP_LAMBDA_FUNCTION
    std::cerr
        << "The cpp Lambda function must be built following the
instructions in the cpp_lambda/README.md file. "
        << std::endl;
#endif
#endif

```

```

        deleteIamRole(clientConfig);
        return false;
    }

    Aws::StringStream buffer;
    buffer << ifstream.rdbuf();

    code.SetZipFile(Aws::Utils::ByteBuffer((unsigned char *)
buffer.str().c_str(),
                                           buffer.str().length()));
    request.SetCode(code);

    Aws::Lambda::Model::CreateFunctionOutcome outcome = client.CreateFunction(
        request);

    if (outcome.IsSuccess()) {
        std::cout << "The lambda function was successfully created. " << seconds
            << " seconds elapsed." << std::endl;
        break;
    }

    else {
        std::cerr << "Error with CreateFunction. "
            << outcome.GetError().GetMessage()
            << std::endl;
        deleteIamRole(clientConfig);
        return false;
    }
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CreateFunction](#)을 참조하세요.

## DeleteFunction

다음 코드 예시는 DeleteFunction의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region in which the bucket was created
    (overrides config file).
    // clientConfig.region = "us-east-1";

    Aws::Lambda::LambdaClient client(clientConfig);

    Aws::Lambda::Model::DeleteFunctionRequest request;
    request.SetFunctionName(LAMBDA_NAME);

    Aws::Lambda::Model::DeleteFunctionOutcome outcome = client.DeleteFunction(
        request);

    if (outcome.IsSuccess()) {
        std::cout << "The lambda function was successfully deleted." << std::endl;
    }
    else {
        std::cerr << "Error with Lambda::DeleteFunction. "
            << outcome.GetError().GetMessage()
            << std::endl;
    }
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteFunction](#)을 참조하세요.

## GetFunction

다음 코드 예시는 GetFunction의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region in which the bucket was created
    (overrides config file).
    // clientConfig.region = "us-east-1";

```

```

Aws::Lambda::LambdaClient client(clientConfig);

Aws::Lambda::Model::GetFunctionRequest request;
request.SetFunctionName(functionName);

Aws::Lambda::Model::GetFunctionOutcome outcome =
client.GetFunction(request);

if (outcome.IsSuccess()) {
    std::cout << "Function retrieve.\n" <<
outcome.GetResult().GetConfiguration().Jsonize().View().WriteReadable()
    << std::endl;
}
else {
    std::cerr << "Error with Lambda::GetFunction. "
    << outcome.GetError().GetMessage()
    << std::endl;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetFunction](#)을 참조하세요.

## Invoke

다음 코드 예시는 Invoke의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region in which the bucket was created
(overrides config file).
// clientConfig.region = "us-east-1";

Aws::Lambda::LambdaClient client(clientConfig);

```

```

Aws::Lambda::Model::InvokeRequest request;
request.SetFunctionName(LAMBDA_NAME);
request.SetLogType(logType);
std::shared_ptr<Aws::IOStream> payload = Aws::MakeShared<Aws::StringStream>(
    "FunctionTest");
*payload << jsonPayload.View().WriteReadable();
request.SetBody(payload);
request.SetContentType("application/json");
Aws::Lambda::Model::InvokeOutcome outcome = client.Invoke(request);

if (outcome.IsSuccess()) {
    invokeResult = std::move(outcome.GetResult());
    result = true;
    break;
}

else {
    std::cerr << "Error with Lambda::InvokeRequest. "
               << outcome.GetError().GetMessage()
               << std::endl;
    break;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [간접 호출](#)을 참조하세요.

## ListFunctions

다음 코드 예시는 ListFunctions의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region in which the bucket was created
(overrides config file).

```

```

// clientConfig.region = "us-east-1";

Aws::Lambda::LambdaClient client(clientConfig);

std::vector<Aws::String> functions;
Aws::String marker;

do {
    Aws::Lambda::Model::ListFunctionsRequest request;
    if (!marker.empty()) {
        request.SetMarker(marker);
    }

    Aws::Lambda::Model::ListFunctionsOutcome outcome = client.ListFunctions(
        request);

    if (outcome.IsSuccess()) {
        const Aws::Lambda::Model::ListFunctionsResult &result =
outcome.GetResult();
        std::cout << result.GetFunctions().size()
            << " lambda functions were retrieved." << std::endl;

        for (const Aws::Lambda::Model::FunctionConfiguration
&functionConfiguration: result.GetFunctions()) {
            functions.push_back(functionConfiguration.GetFunctionName());
            std::cout << functions.size() << " "
                << functionConfiguration.GetDescription() << std::endl;
            std::cout << " "
                << Aws::Lambda::Model::RuntimeMapper::GetNameForRuntime(
                    functionConfiguration.GetRuntime()) << ": "
                << functionConfiguration.GetHandler()
                << std::endl;
        }
        marker = result.GetNextMarker();
    }
    else {
        std::cerr << "Error with Lambda::ListFunctions. "
            << outcome.GetError().GetMessage()
            << std::endl;
    }
} while (!marker.empty());

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ListFunctions](#)를 참조하세요.

## UpdateFunctionCode

다음 코드 예시는 UpdateFunctionCode의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region in which the bucket was created
    (overrides config file).
    // clientConfig.region = "us-east-1";

    Aws::Lambda::LambdaClient client(clientConfig);

    Aws::Lambda::Model::UpdateFunctionCodeRequest request;
    request.SetFunctionName(LAMBDA_NAME);
    std::ifstream ifstream(CALCULATOR_LAMBDA_CODE.c_str(),
                           std::ios_base::in | std::ios_base::binary);
    if (!ifstream.is_open()) {
        std::cerr << "Error opening file " << INCREMENT_LAMBDA_CODE << "." <<
std::endl;

#ifdef USE_CPP_LAMBDA_FUNCTION
        std::cerr
            << "The cpp Lambda function must be built following the
instructions in the cpp_lambda/README.md file. "
            << std::endl;
#endif

        deleteLambdaFunction(client);
        deleteIamRole(clientConfig);
        return false;
    }

    Aws::StringStream buffer;
    buffer << ifstream.rdbuf();
    request.SetZipFile(
        Aws::Utils::ByteBuffer((unsigned char *) buffer.str().c_str(),
                                buffer.str().length()));

```

```

    request.SetPublish(true);

    Aws::Lambda::Model::UpdateFunctionCodeOutcome outcome =
    client.UpdateFunctionCode(
        request);

    if (outcome.IsSuccess()) {
        std::cout << "The lambda code was successfully updated." << std::endl;
    }
    else {
        std::cerr << "Error with Lambda::UpdateFunctionCode. "
            << outcome.GetError().GetMessage()
            << std::endl;
    }
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [UpdateFunctionCode](#)를 참조하세요.

## UpdateFunctionConfiguration

다음 코드 예시는 UpdateFunctionConfiguration의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region in which the bucket was created
    (overrides config file).
    // clientConfig.region = "us-east-1";

    Aws::Lambda::LambdaClient client(clientConfig);

    Aws::Lambda::Model::UpdateFunctionConfigurationRequest request;
    request.SetFunctionName(LAMBDA_NAME);
    Aws::Lambda::Model::Environment environment;
    environment.AddVariables("LOG_LEVEL", "DEBUG");
    request.SetEnvironment(environment);

```

```
Aws::Lambda::Model::UpdateFunctionConfigurationOutcome outcome =
client.UpdateFunctionConfiguration(
    request);

if (outcome.IsSuccess()) {
    std::cout << "The lambda configuration was successfully updated."
    << std::endl;
    break;
}

else {
    std::cerr << "Error with Lambda::UpdateFunctionConfiguration. "
    << outcome.GetError().GetMessage()
    << std::endl;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [UpdateFunctionConfiguration](#)을 참조하세요.

## 시나리오

### 사진을 관리하기 위한 서버리스 애플리케이션 만들기

다음 코드 예시에서는 사용자가 레이블을 사용하여 사진을 관리할 수 있는 서버리스 애플리케이션을 생성하는 방법을 보여줍니다.

#### SDK for C++

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 [AWS 커뮤니티](#)의 게시물을 참조하세요.

이 예제에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition

- Amazon S3
- Amazon SNS

## SDK for C++를 사용한 MediaConvert 예제

다음 코드 예제에서는 MediaConvert와 AWS SDK for C++ 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

#### CreateJob

다음 코드 예시는 CreateJob의 사용 방법을 보여줍니다.

SDK for C++

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
#!/ Create an AWS Elemental MediaConvert job.
/*!
    \param mediaConvertRole: An Amazon Resource Name (ARN) for the AWS Identity and
                             Access Management (IAM) role for the job.
    \param fileInput: A URI to an input file that is stored in Amazon Simple Storage
    Service
                       (Amazon S3) or on an HTTP(S) server.
```

```

\param fileOutput: A URI for an Amazon S3 output location and the output file name
base.
\param jobSettingsFile: An optional JSON settings file.
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/

```

```

bool AwsDoc::MediaConvert::createJob(const Aws::String &mediaConvertRole,
                                     const Aws::String &fileInput,
                                     const Aws::String &fileOutput,
                                     const Aws::String &jobSettingsFile,
                                     const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::MediaConvert::Model::CreateJobRequest createJobRequest;

    createJobRequest.SetRole(mediaConvertRole);
    Aws::Http::HeaderValueCollection hvc;
    hvc.emplace("Customer", "Amazon");
    createJobRequest.SetUserMetadata(hvc);

    if (!jobSettingsFile.empty()) // Use a JSON file for the job settings.
    {
        std::ifstream jobSettingsStream(jobSettingsFile, std::ios::ate);
        if (!jobSettingsStream) {
            std::cerr << "Unable to open the job template file." << std::endl;
            return false;
        }
        std::vector<char> buffer(jobSettingsStream.tellg());
        jobSettingsStream.seekg(0);
        jobSettingsStream.read(buffer.data(), buffer.size());
        std::string jobSettingsJSON(buffer.data(), buffer.size());
        size_t pos = jobSettingsJSON.find(INPUT_FILE_PLACEHOLDER);
        if (pos != std::string::npos) {
            jobSettingsJSON.replace(pos, strlen(INPUT_FILE_PLACEHOLDER), fileInput);
        }

        pos = jobSettingsJSON.find(OUTPUT_FILE_PLACEHOLDER);
        if (pos != std::string::npos) {
            jobSettingsJSON.replace(pos, strlen(OUTPUT_FILE_PLACEHOLDER),
fileOutput);
        }
        Aws::Utils::Json::JsonValue jsonValue(jobSettingsJSON);
        Aws::MediaConvert::Model::JobSettings jobSettings(jsonValue);
    }
}

```

```

        createJobRequest.SetSettings(jobSettings);
    }
    else { // Configure the job settings programmatically.
        Aws::MediaConvert::Model::JobSettings jobSettings;
        jobSettings.SetAdAvailOffset(0);
        Aws::MediaConvert::Model::TimecodeConfig timecodeConfig;

timecodeConfig.SetSource(Aws::MediaConvert::Model::TimecodeSource::EMBEDDED);
        jobSettings.SetTimecodeConfig(timecodeConfig);

        // Configure the output group.
        Aws::MediaConvert::Model::OutputGroup outputGroup;
        outputGroup.SetName("File Group");
        Aws::MediaConvert::Model::OutputGroupSettings outputGroupSettings;
        outputGroupSettings.SetType(
            Aws::MediaConvert::Model::OutputGroupType::FILE_GROUP_SETTINGS);
        Aws::MediaConvert::Model::FileGroupSettings fileGroupSettings;
        fileGroupSettings.SetDestination(fileOutput);
        outputGroupSettings.SetFileGroupSettings(fileGroupSettings);
        outputGroup.SetOutputGroupSettings(outputGroupSettings);

        Aws::MediaConvert::Model::Output output;
        output.SetNameModifier("_1");

        Aws::MediaConvert::Model::VideoDescription videoDescription;
        videoDescription.SetScalingBehavior(
            Aws::MediaConvert::Model::ScalingBehavior::DEFAULT);
        videoDescription.SetTimecodeInsertion(
            Aws::MediaConvert::Model::VideoTimecodeInsertion::DISABLED);
        videoDescription.SetAntiAlias(Aws::MediaConvert::Model::AntiAlias::ENABLED);
        videoDescription.SetSharpness(50);

        videoDescription.SetAfdSignaling(Aws::MediaConvert::Model::AfdSignaling::NONE);
        videoDescription.SetDropFrameTimecode(
            Aws::MediaConvert::Model::DropFrameTimecode::ENABLED);

        videoDescription.SetRespondToAfd(Aws::MediaConvert::Model::RespondToAfd::NONE);
        videoDescription.SetColorMetadata(
            Aws::MediaConvert::Model::ColorMetadata::INSERT);

        Aws::MediaConvert::Model::VideoCodecSettings videoCodecSettings;
        videoCodecSettings.SetCodec(Aws::MediaConvert::Model::VideoCodec::H_264);
        Aws::MediaConvert::Model::H264Settings h264Settings;
        h264Settings.SetNumberReferenceFrames(3);
    }
}

```

```

h264Settings.SetSyntax(Aws::MediaConvert::Model::H264Syntax::DEFAULT);
h264Settings.SetSoftness(0);
h264Settings.SetGopClosedCadence(1);
h264Settings.SetGopSize(90);
h264Settings.SetSlices(1);
h264Settings.SetGopBReference(
    Aws::MediaConvert::Model::H264GopBReference::DISABLED);
h264Settings.SetSlowPal(Aws::MediaConvert::Model::H264SlowPal::DISABLED);
h264Settings.SetSpatialAdaptiveQuantization(
    Aws::MediaConvert::Model::H264SpatialAdaptiveQuantization::ENABLED);
h264Settings.SetTemporalAdaptiveQuantization(
    Aws::MediaConvert::Model::H264TemporalAdaptiveQuantization::ENABLED);
h264Settings.SetFlickerAdaptiveQuantization(
    Aws::MediaConvert::Model::H264FlickerAdaptiveQuantization::DISABLED);
h264Settings.SetEntropyEncoding(
    Aws::MediaConvert::Model::H264EntropyEncoding::CABAC);
h264Settings.SetBitrate(5000000);
h264Settings.SetFramerateControl(
    Aws::MediaConvert::Model::H264FramerateControl::SPECIFIED);
h264Settings.SetRateControlMode(
    Aws::MediaConvert::Model::H264RateControlMode::CBR);

h264Settings.SetCodecProfile(Aws::MediaConvert::Model::H264CodecProfile::MAIN);
h264Settings.SetTelecine(Aws::MediaConvert::Model::H264Telecine::NONE);
h264Settings.SetMinIInterval(0);
h264Settings.SetAdaptiveQuantization(
    Aws::MediaConvert::Model::H264AdaptiveQuantization::HIGH);
h264Settings.SetCodecLevel(Aws::MediaConvert::Model::H264CodecLevel::AUTO);
h264Settings.SetFieldEncoding(
    Aws::MediaConvert::Model::H264FieldEncoding::PAFF);
h264Settings.SetSceneChangeDetect(
    Aws::MediaConvert::Model::H264SceneChangeDetect::ENABLED);
h264Settings.SetQualityTuningLevel(
    Aws::MediaConvert::Model::H264QualityTuningLevel::SINGLE_PASS);
h264Settings.SetFramerateConversionAlgorithm(
    Aws::MediaConvert::Model::H264FramerateConversionAlgorithm::DUPLICATE_DROP);
h264Settings.SetUnregisteredSeiTimecode(
    Aws::MediaConvert::Model::H264UnregisteredSeiTimecode::DISABLED);
h264Settings.SetGopSizeUnits(
    Aws::MediaConvert::Model::H264GopSizeUnits::FRAMES);

```

```

h264Settings.SetParControl(Aws::MediaConvert::Model::H264ParControl::SPECIFIED);
h264Settings.SetNumberBFramesBetweenReferenceFrames(2);

h264Settings.SetRepeatPps(Aws::MediaConvert::Model::H264RepeatPps::DISABLED);
h264Settings.SetFramerateNumerator(30);
h264Settings.SetFramerateDenominator(1);
h264Settings.SetParNumerator(1);
h264Settings.SetParDenominator(1);
videoCodecSettings.SetH264Settings(h264Settings);
videoDescription.SetCodecSettings(videoCodecSettings);
output.SetVideoDescription(videoDescription);

Aws::MediaConvert::Model::AudioDescription audioDescription;
audioDescription.SetLanguageCodeControl(
    Aws::MediaConvert::Model::AudioLanguageCodeControl::FOLLOW_INPUT);
audioDescription.SetAudioSourceName(AUDIO_SOURCE_NAME);
Aws::MediaConvert::Model::AudioCodecSettings audioCodecSettings;
audioCodecSettings.SetCodec(Aws::MediaConvert::Model::AudioCodec::AAC);
Aws::MediaConvert::Model::AacSettings aacSettings;
aacSettings.SetAudioDescriptionBroadcasterMix(

Aws::MediaConvert::Model::AacAudioDescriptionBroadcasterMix::NORMAL);
aacSettings.SetRateControlMode(
    Aws::MediaConvert::Model::AacRateControlMode::CBR);
aacSettings.SetCodecProfile(Aws::MediaConvert::Model::AacCodecProfile::LC);
aacSettings.SetCodingMode(
    Aws::MediaConvert::Model::AacCodingMode::CODING_MODE_2_0);
aacSettings.SetRawFormat(Aws::MediaConvert::Model::AacRawFormat::NONE);
aacSettings.SetSampleRate(48000);

aacSettings.SetSpecification(Aws::MediaConvert::Model::AacSpecification::MPEG4);
aacSettings.SetBitrate(64000);
audioCodecSettings.SetAacSettings(aacSettings);
audioDescription.SetCodecSettings(audioCodecSettings);
Aws::Vector<Aws::MediaConvert::Model::AudioDescription> audioDescriptions;
audioDescriptions.emplace_back(audioDescription);
output.SetAudioDescriptions(audioDescriptions);

Aws::MediaConvert::Model::ContainerSettings mp4container;
mp4container.SetContainer(Aws::MediaConvert::Model::ContainerType::MP4);
Aws::MediaConvert::Model::Mp4Settings mp4Settings;
mp4Settings.SetCslgAtom(Aws::MediaConvert::Model::Mp4CslgAtom::INCLUDE);

```

```

mp4Settings.SetFreeSpaceBox(Aws::MediaConvert::Model::Mp4FreeSpaceBox::EXCLUDE);
    mp4Settings.SetMoovPlacement(
        Aws::MediaConvert::Model::Mp4MoovPlacement::PROGRESSIVE_DOWNLOAD);
mp4container.SetMp4Settings(mp4Settings);
output.SetContainerSettings(mp4container);

outputGroup.AddOutputs(output);
jobSettings.AddOutputGroups(outputGroup);

// Configure inputs.
Aws::MediaConvert::Model::Input input;
input.SetFilterEnable(Aws::MediaConvert::Model::InputFilterEnable::AUTO);
input.SetPsiControl(Aws::MediaConvert::Model::InputPsiControl::USE_PSI);
input.SetFilterStrength(0);

input.SetDeblockFilter(Aws::MediaConvert::Model::InputDeblockFilter::DISABLED);

input.SetDenoiseFilter(Aws::MediaConvert::Model::InputDenoiseFilter::DISABLED);
    input.SetTimecodeSource(
        Aws::MediaConvert::Model::InputTimecodeSource::EMBEDDED);
input.SetFileInput(fileInput);

Aws::MediaConvert::Model::AudioSelector audioSelector;
audioSelector.SetOffset(0);
audioSelector.SetDefaultSelection(
    Aws::MediaConvert::Model::AudioDefaultSelection::NOT_DEFAULT);
audioSelector.SetProgramSelection(1);
audioSelector.SetSelectorType(
    Aws::MediaConvert::Model::AudioSelectorType::TRACK);
audioSelector.AddTracks(1);
input.AddAudioSelectors(AUDIO_SOURCE_NAME, audioSelector);

Aws::MediaConvert::Model::VideoSelector videoSelector;
videoSelector.SetColorSpace(Aws::MediaConvert::Model::ColorSpace::FOLLOW);
input.SetVideoSelector(videoSelector);

jobSettings.AddInputs(input);

createJobRequest.SetSettings(jobSettings);
}

Aws::MediaConvert::MediaConvertClient client(clientConfiguration);
Aws::MediaConvert::Model::CreateJobOutcome outcome = client.CreateJob(

```

```

        createJobRequest);
    if (outcome.IsSuccess()) {
        std::cout << "Job successfully created with ID - "
                  << outcome.GetResult().GetJob().GetId() << std::endl;
    }
    else {
        std::cerr << "Error CreateJob - " << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CreateJob](#)을 참조하세요.

## GetJob

다음 코드 예시는 GetJob의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Retrieve the information for a specific completed transcoding job.
/*!
    \param jobID: A job ID.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::MediaConvert::getJob(const Aws::String &jobID,
                                   const Aws::Client::ClientConfiguration
                                   &clientConfiguration) {
    Aws::MediaConvert::MediaConvertClient client(clientConfiguration);

    Aws::MediaConvert::Model::GetJobRequest request;
    request.SetId(jobID);
}

```

```

const Aws::MediaConvert::Model::GetJobOutcome outcome = client.GetJob(
    request);
if (outcome.IsSuccess()) {
    std::cout << outcome.GetResult().GetJob().Jsonize().View().WriteReadable()
        << std::endl;
}
else {
    std::cerr << "DescribeEndpoints error - " << outcome.GetError().GetMessage()
        << std::endl;
}

return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetJob](#)을 참조하세요.

## ListJobs

다음 코드 예시는 ListJobs의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Retrieve a list of created jobs.
/*!
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::MediaConvert::listJobs(
    const Aws::Client::ClientConfiguration &clientConfiguration) {

    Aws::MediaConvert::MediaConvertClient client(clientConfiguration);

    bool result = true;

```

```

    Aws::String nextToken; // Used to handle paginated results.
    do {
        Aws::MediaConvert::Model::ListJobsRequest request;
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }
        const Aws::MediaConvert::Model::ListJobsOutcome outcome = client.ListJobs(
            request);
        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::MediaConvert::Model::Job> &jobs =
                outcome.GetResult().GetJobs();
            std::cout << jobs.size() << " jobs retrieved." << std::endl;
            for (const Aws::MediaConvert::Model::Job &job: jobs) {
                std::cout << " " << job.Jsonize().View().WriteReadable() <<
std::endl;
            }

            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cerr << "DescribeEndpoints error - " <<
outcome.GetError().GetMessage()
                << std::endl;
            result = false;
            break;
        }
    } while (!nextToken.empty());

    return result;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ListJobs](#)를 참조하세요.

## SDK for C++를 사용한 Amazon RDS 예제

다음 코드 예제에서는 AWS SDK for C++ Amazon RDS에서를 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](#)
- [기본 사항](#)
- [작업](#)
- [시나리오](#)

## 시작하기

Hello Amazon RDS

다음 코드 예제에서는 Amazon RDS 사용을 시작하는 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

CMakeLists.txt CMake 파일의 코드입니다.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS rds)

# Set this project's name.
project("hello_rds")
```

```

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
"${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory for
    running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
    may need to uncomment this

                                # and set the proper subdirectory to the
    executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
"${CMAKE_CURRENT_BINARY_DIR}/${BIN_SUB_DIR}")
endif ()

add_executable(${PROJECT_NAME}
    hello_rds.cpp)

target_link_libraries(${PROJECT_NAME}
    ${AWSSDK_LINK_LIBRARIES})

```

hello\_rds.cpp 소스 파일의 코드입니다.

```

#include <aws/core/Aws.h>
#include <aws/rds/RDSCClient.h>
#include <aws/rds/model/DescribeDBInstancesRequest.h>

```

```
#include <iostream>

/*
 * A "Hello Rds" starter application which initializes an Amazon Relational
 * Database Service (Amazon RDS) client and
 * describes the Amazon RDS instances.
 *
 * main function
 *
 * Usage: 'hello_rds'
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::RDS::RDSClient rdsClient(clientConfig);
        Aws::String marker;
        std::vector<Aws::String> instanceDBIDs;

        do {
            Aws::RDS::Model::DescribeDBInstancesRequest request;

            if (!marker.empty()) {
                request.SetMarker(marker);
            }

            Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
                rdsClient.DescribeDBInstances(request);

            if (outcome.IsSuccess()) {
                for (auto &instance: outcome.GetResult().GetDBInstances()) {
                    instanceDBIDs.push_back(instance.GetDBInstanceIdentifier());
                }
                marker = outcome.GetResult().GetMarker();
            } else {
```

```

        result = 1;
        std::cerr << "Error with RDS::DescribeDBInstances. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        break;
    }
} while (!marker.empty());

std::cout << instanceDBIDs.size() << " RDS instances found." << std::endl;
for (auto &instanceDBID: instanceDBIDs) {
    std::cout << "    Instance: " << instanceDBID << std::endl;
}
}

Aws::ShutdownAPI(options); // Should only be called once.
return result;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DescribeDBInstances](#)를 참조하세요.

## 기본 사항

### 기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 사용자 지정 DB 파라미터 그룹을 생성하고 파라미터 값을 설정합니다.
- 파라미터 그룹을 사용하도록 구성된 DB 인스턴스를 생성합니다. DB 인스턴스에는 데이터베이스도 포함되어 있습니다.
- 인스턴스의 스냅샷을 만듭니다.
- 인스턴스 및 파라미터 그룹을 삭제합니다.

### SDK for C++

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예제 리포지토리](#)에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    //! Routine which creates an Amazon RDS instance and demonstrates several operations
    //! on that instance.
    /*!
    \sa gettingStartedWithDBInstances()
    \param clientConfiguration: AWS client configuration.
    \return bool: Successful completion.
    */
    bool AwsDoc::RDS::gettingStartedWithDBInstances(
        const Aws::Client::ClientConfiguration &clientConfig) {
        Aws::RDS::RDSClient client(clientConfig);

        printAsterisksLine();
        std::cout << "Welcome to the Amazon Relational Database Service (Amazon RDS)"
            << std::endl;
        std::cout << "get started with DB instances demo." << std::endl;
        printAsterisksLine();

        std::cout << "Checking for an existing DB parameter group named '" <<
            PARAMETER_GROUP_NAME << "'." << std::endl;
        Aws::String dbParameterGroupFamily("Undefined");
        bool parameterGroupFound = true;
        {
            // 1. Check if the DB parameter group already exists.
            Aws::RDS::Model::DescribeDBParameterGroupsRequest request;
            request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);

            Aws::RDS::Model::DescribeDBParameterGroupsOutcome outcome =
                client.DescribeDBParameterGroups(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB parameter group named '" <<
                    PARAMETER_GROUP_NAME << "' already exists." << std::endl;
                dbParameterGroupFamily = outcome.GetResult().GetDBParameterGroups()
[0].GetDBParameterGroupFamily();
            }
            else if (outcome.GetError().GetErrorType() ==
                Aws::RDS::RDSErrors::D_B_PARAMETER_GROUP_NOT_FOUND_FAULT) {
                std::cout << "DB parameter group named '" <<
                    PARAMETER_GROUP_NAME << "' does not exist." << std::endl;
            }
        }
    }

```

```

        parameterGroupFound = false;
    }
    else {
        std::cerr << "Error with RDS::DescribeDBParameterGroups. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        return false;
    }
}

if (!parameterGroupFound) {
    Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

    // 2. Get available engine versions for the specified engine.
    if (!getDBEngineVersions(DB_ENGINE, NO_PARAMETER_GROUP_FAMILY,
                            engineVersions, client)) {
        return false;
    }

    std::cout << "Getting available database engine versions for " << DB_ENGINE
              << "."
              << std::endl;
    std::vector<Aws::String> families;
    for (const Aws::RDS::Model::DBEngineVersion &version: engineVersions) {
        Aws::String family = version.GetDBParameterGroupFamily();
        if (std::find(families.begin(), families.end(), family) ==
            families.end()) {
            families.push_back(family);
            std::cout << " " << families.size() << ": " << family << std::endl;
        }
    }

    int choice = askQuestionForIntRange("Which family do you want to use? ", 1,
                                       static_cast<int>(families.size()));
    dbParameterGroupFamily = families[choice - 1];
}

if (!parameterGroupFound) {
    // 3. Create a DB parameter group.
    Aws::RDS::Model::CreateDBParameterGroupRequest request;
    request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
    request.SetDBParameterGroupFamily(dbParameterGroupFamily);
    request.SetDescription("Example parameter group.");

    Aws::RDS::Model::CreateDBParameterGroupOutcome outcome =

```

```

        client.CreateDBParameterGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB parameter group was successfully created."
                  << std::endl;
    }
    else {
        std::cerr << "Error with RDS::CreateDBParameterGroup. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        return false;
    }
}

printAsterisksLine();
std::cout << "Let's set some parameter values in your parameter group."
          << std::endl;

Aws::String marker;
Aws::Vector<Aws::RDS::Model::Parameter> autoIncrementParameters;
// 4. Get the parameters in the DB parameter group.
if (!getDBParameters(PARAMETER_GROUP_NAME, AUTO_INCREMENT_PREFIX, NO_SOURCE,
                    autoIncrementParameters,
                    client)) {
    cleanUpResources(PARAMETER_GROUP_NAME, "", client);
    return false;
}

Aws::Vector<Aws::RDS::Model::Parameter> updateParameters;

for (Aws::RDS::Model::Parameter &autoIncParameter: autoIncrementParameters) {
    if (autoIncParameter.GetIsModifiable() &&
        (autoIncParameter.GetDataType() == "integer")) {
        std::cout << "The " << autoIncParameter.GetParameterName()
                  << " is described as: " <<
                  autoIncParameter.GetDescription() << "." << std::endl;
        if (autoIncParameter.ParameterValueHasBeenSet()) {
            std::cout << "The current value is "
                      << autoIncParameter.GetParameterValue()
                      << "." << std::endl;
        }
        std::vector<int> splitValues = splitToInts(
            autoIncParameter.GetAllowedValues(), '-');
        if (splitValues.size() == 2) {

```

```

        int newValue = askQuestionForIntRange(
            Aws::String("Enter a new value in the range ") +
            autoIncParameter.GetAllowedValues() + ": ",
            splitValues[0], splitValues[1]);
        autoIncParameter.SetParameterValue(std::to_string(newValue));
        updateParameters.push_back(autoIncParameter);
    }
    else {
        std::cerr << "Error parsing " << autoIncParameter.GetAllowedValues()
            << std::endl;
    }
}

{
    // 5. Modify the auto increment parameters in the group.
    Aws::RDS::Model::ModifyDBParameterGroupRequest request;
    request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
    request.SetParameters(updateParameters);

    Aws::RDS::Model::ModifyDBParameterGroupOutcome outcome =
        client.ModifyDBParameterGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB parameter group was successfully modified."
            << std::endl;
    }
    else {
        std::cerr << "Error with RDS::ModifyDBParameterGroup. "
            << outcome.GetError().GetMessage()
            << std::endl;
    }
}

std::cout
    << "You can get a list of parameters you've set by specifying a source
of 'user'."
    << std::endl;

    Aws::Vector<Aws::RDS::Model::Parameter> userParameters;
    // 6. Display the modified parameters in the group.
    if (!getDBParameters(PARAMETER_GROUP_NAME, NO_NAME_PREFIX, "user",
        userParameters,
```

```

        client)) {
    cleanUpResources(PARAMETER_GROUP_NAME, "", client);
    return false;
}

for (const auto &userParameter: userParameters) {
    std::cout << "  " << userParameter.GetParameterName() << ", " <<
        userParameter.GetDescription() << ", parameter value - "
        << userParameter.GetParameterValue() << std::endl;
}

printAsterisksLine();
std::cout << "Checking for an existing DB instance." << std::endl;

Aws::RDS::Model::DBInstance dbInstance;
// 7. Check if the DB instance already exists.
if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
    cleanUpResources(PARAMETER_GROUP_NAME, "", client);
    return false;
}

if (dbInstance.DbInstancePortHasBeenSet()) {
    std::cout << "The DB instance already exists." << std::endl;
}
else {
    std::cout << "Let's create a DB instance." << std::endl;
    const Aws::String administratorName = askQuestion(
        "Enter an administrator username for the database: ");
    const Aws::String administratorPassword = askQuestion(
        "Enter a password for the administrator (at least 8 characters): ");
    Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

    // 8. Get a list of available engine versions.
    if (!getDBEngineVersions(DB_ENGINE, dbParameterGroupFamily, engineVersions,
        client)) {
        cleanUpResources(PARAMETER_GROUP_NAME, "", client);
        return false;
    }

    std::cout << "The available engines for your parameter group are:" <<
std::endl;

    int index = 1;

```

```

    for (const Aws::RDS::Model::DBEngineVersion &engineVersion: engineVersions)
    {
        std::cout << " " << index << ": " << engineVersion.GetEngineVersion()
            << std::endl;
        ++index;
    }
    int choice = askQuestionForIntRange("Which engine do you want to use? ", 1,
static_cast<int>(engineVersions.size()));
    const Aws::RDS::Model::DBEngineVersion engineVersion = engineVersions[choice
-
                                                                    1];

    Aws::String dbInstanceClass;
    // 9. Get a list of micro instance classes.
    if (!chooseMicroDBInstanceClass(engineVersion.GetEngine(),
        engineVersion.GetEngineVersion(),
        dbInstanceClass,
        client)) {
        cleanUpResources(PARAMETER_GROUP_NAME, "", client);
        return false;
    }

    std::cout << "Creating a DB instance named '" << DB_INSTANCE_IDENTIFIER
        << "' and database '" << DB_NAME << "'.\n"
        << "The DB instance is configured to use your custom parameter
group '"
        << PARAMETER_GROUP_NAME << "',\n"
        << "selected engine version " << engineVersion.GetEngineVersion()
        << ",\n"
        << "selected DB instance class '" << dbInstanceClass << "',"
        << " and " << DB_ALLOCATED_STORAGE << " GiB of " <<
DB_STORAGE_TYPE
        << " storage.\nThis typically takes several minutes." <<
std::endl;

    Aws::RDS::Model::CreateDBInstanceRequest request;
    request.SetDBName(DB_NAME);
    request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
    request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
    request.SetEngine(engineVersion.GetEngine());
    request.SetEngineVersion(engineVersion.GetEngineVersion());
    request.SetDBInstanceClass(dbInstanceClass);
    request.SetStorageType(DB_STORAGE_TYPE);

```

```

request.SetAllocatedStorage(DB_ALLOCATED_STORAGE);
request.SetMasterUsername(administratorName);
request.SetMasterUserPassword(administratorPassword);

Aws::RDS::Model::CreateDBInstanceOutcome outcome =
    client.CreateDBInstance(request);

if (outcome.IsSuccess()) {
    std::cout << "The DB instance creation has started."
                << std::endl;
}
else {
    std::cerr << "Error with RDS::CreateDBInstance. "
                << outcome.GetError().GetMessage()
                << std::endl;
    cleanUpResources(PARAMETER_GROUP_NAME, "", client);
    return false;
}
}

std::cout << "Waiting for the DB instance to become available." << std::endl;

int counter = 0;
// 11. Wait for the DB instance to become available.
do {
    std::this_thread::sleep_for(std::chrono::seconds(1));
    ++counter;
    if (counter > 900) {
        std::cerr << "Wait for instance to become available timed out after "
                    << counter
                    << " seconds." << std::endl;
        cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER, client);
        return false;
    }

    dbInstance = Aws::RDS::Model::DBInstance();
    if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
        cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER, client);
        return false;
    }

    if ((counter % 20) == 0) {
        std::cout << "Current DB instance status is '"
                    << dbInstance.GetDBInstanceStatus()

```

```

        << " ' after " << counter << " seconds." << std::endl;
    }
} while (dbInstance.GetDBInstanceStatus() != "available");

if (dbInstance.GetDBInstanceStatus() == "available") {
    std::cout << "The DB instance has been created." << std::endl;
}

printAsterisksLine();

// 12. Display the connection string that can be used to connect a 'mysql' shell
to the database.
displayConnection(dbInstance);

printAsterisksLine();

if (askYesNoQuestion(
    "Do you want to create a snapshot of your DB instance (y/n)? ") ) {
    Aws::String snapshotID(DB_INSTANCE_IDENTIFIER + "-" +
        Aws::String(Aws::Utils::UUID::RandomUUID()));
    {
        std::cout << "Creating a snapshot named " << snapshotID << "." <<
std::endl;
        std::cout << "This typically takes a few minutes." << std::endl;

        // 13. Create a snapshot of the DB instance.
        Aws::RDS::Model::CreateDBSnapshotRequest request;
        request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
        request.SetDBSnapshotIdentifier(snapshotID);

        Aws::RDS::Model::CreateDBSnapshotOutcome outcome =
            client.CreateDBSnapshot(request);

        if (outcome.IsSuccess()) {
            std::cout << "Snapshot creation has started."
                << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBSnapshot. "
                << outcome.GetError().GetMessage()
                << std::endl;
            cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
            return false;
        }
    }
}

```

```

    }
}

std::cout << "Waiting for snapshot to become available." << std::endl;

Aws::RDS::Model::DBSnapshot snapshot;
counter = 0;
do {
    std::this_thread::sleep_for(std::chrono::seconds(1));
    ++counter;
    if (counter > 600) {
        std::cerr << "Wait for snapshot to be available timed out after "
                    << counter
                    << " seconds." << std::endl;
        cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
        return false;
    }

    // 14. Wait for the snapshot to become available.
    Aws::RDS::Model::DescribeDBSnapshotsRequest request;
    request.SetDBSnapshotIdentifier(snapshotID);

    Aws::RDS::Model::DescribeDBSnapshotsOutcome outcome =
        client.DescribeDBSnapshots(request);

    if (outcome.IsSuccess()) {
        snapshot = outcome.GetResult().GetDBSnapshots()[0];
    }
    else {
        std::cerr << "Error with RDS::DescribeDBSnapshots. "
                    << outcome.GetError().GetMessage()
                    << std::endl;
        cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
        return false;
    }

    if ((counter % 20) == 0) {
        std::cout << "Current snapshot status is '"
                    << snapshot.GetStatus()
                    << "' after " << counter << " seconds." << std::endl;
    }
} while (snapshot.GetStatus() != "available");

```

```

        if (snapshot.GetStatus() != "available") {
            std::cout << "A snapshot has been created." << std::endl;
        }
    }

    printAsterisksLine();

    bool result = true;
    if (askYesNoQuestion(
        "Do you want to delete the DB instance and parameter group (y/n)? ") {
        result = cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
    }

    return result;
}

//! Routine which gets DB parameters using the 'DescribeDBParameters' api.
/*!
\sa getDBParameters()
\param parameterGroupName: The name of the parameter group.
\param namePrefix: Prefix string to filter results by parameter name.
\param source: A source such as 'user', ignored if empty.
\param parametersResult: Vector of 'Parameter' objects returned by the routine.
\param client: 'RDSClient' instance.
\return bool: Successful completion.
*/
bool AwsDoc::RDS::getDBParameters(const Aws::String &parameterGroupName,
                                   const Aws::String &namePrefix,
                                   const Aws::String &source,
                                   Aws::Vector<Aws::RDS::Model::Parameter>
&parametersResult,
                                   const Aws::RDS::RDSClient &client) {
    Aws::String marker;
    do {
        Aws::RDS::Model::DescribeDBParametersRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);

```

```

    }

    Aws::RDS::Model::DescribeDBParametersOutcome outcome =
        client.DescribeDBParameters(request);

    if (outcome.IsSuccess()) {
        const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
            outcome.GetResult().GetParameters();
        for (const Aws::RDS::Model::Parameter &parameter: parameters) {
            if (!namePrefix.empty()) {
                if (parameter.GetParameterName().find(namePrefix) == 0) {
                    parametersResult.push_back(parameter);
                }
            }
            else {
                parametersResult.push_back(parameter);
            }
        }

        marker = outcome.GetResult().GetMarker();
    }
    else {
        std::cerr << "Error with RDS::DescribeDBParameters. "
            << outcome.GetError().GetMessage()
            << std::endl;
        return false;
    }
} while (!marker.empty());

return true;
}

//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
    \sa getDBEngineVersions()
    \param engineName: A DB engine name.
    \param parameterGroupFamily: A parameter group family name, ignored if empty.
    \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
    routine.
    \param client: 'RDSClient' instance.
    \return bool: Successful completion.
*/

```

```

bool AwsDoc::RDS::getDBEngineVersions(const Aws::String &engineName,
                                       const Aws::String &parameterGroupFamily,
                                       Aws::Vector<Aws::RDS::Model::DBEngineVersion>
&engineVersionsResult,
                                       const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // Used for pagination.

    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
            client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            auto &engineVersions = outcome.GetResult().GetDBEngineVersions();
            engineVersionsResult.insert(engineVersionsResult.end(),
engineVersions.begin(),
                                       engineVersions.end());
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeDBEngineVersionsRequest. "
                        << outcome.GetError().GetMessage()
                        << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}

```

```

//! Routine which gets a DB instance description.
/*!
\sa describeDBInstance()
\param dbInstanceIdIdentifier: A DB instance identifier.
\param instanceResult: The 'DBInstance' object containing the description.
\param client: 'RDSClient' instance.
\return bool: Successful completion.
*/
bool AwsDoc::RDS::describeDBInstance(const Aws::String &dbInstanceIdIdentifier,
                                     Aws::RDS::Model::DBInstance &instanceResult,
                                     const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBInstancesRequest request;
    request.SetDBInstanceIdIdentifier(dbInstanceIdIdentifier);

    Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
        client.DescribeDBInstances(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        instanceResult = outcome.GetResult().GetDBInstances()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
             Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with RDS::DescribeDBInstances. "
                   << outcome.GetError().GetMessage()
                   << std::endl;
    }

    // This example does not log an error if the DB instance does not exist.
    // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}

//! Routine which gets available 'micro' DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
\sa chooseMicroDBInstanceClass()
\param engineName: The DB engine name.
\param engineVersion: The DB engine version.

```

```

\param dbInstanceClass: String for DB instance class chosen by the user.
\param client: 'RDSClient' instance.
\return bool: Successful completion.
*/
bool AwsDoc::RDS::chooseMicroDBInstanceClass(const Aws::String &engine,
                                              const Aws::String &engineVersion,
                                              Aws::String &dbInstanceClass,
                                              const Aws::RDS::RDSClient &client) {
    std::vector<Aws::String> instanceClasses;
    Aws::String marker;
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
            client.DescribeOrderableDBInstanceOptions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption> &options =
                outcome.GetResult().GetOrderableDBInstanceOptions();
            for (const Aws::RDS::Model::OrderableDBInstanceOption &option: options)
            {
                const Aws::String &instanceClass = option.GetDBInstanceClass();
                if (instanceClass.find("micro") != std::string::npos) {
                    if (std::find(instanceClasses.begin(), instanceClasses.end(),
                                instanceClass) ==
                        instanceClasses.end()) {
                        instanceClasses.push_back(instanceClass);
                    }
                }
            }
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeOrderableDBInstanceOptions. "
                << outcome.GetError().GetMessage()
                << std::endl;
            return false;
        }
    } while (!marker.empty());
}

```

```

    std::cout << "The available micro DB instance classes for your database engine
are:"
        << std::endl;
    for (int i = 0; i < instanceClasses.size(); ++i) {
        std::cout << "    " << i + 1 << ": " << instanceClasses[i] << std::endl;
    }

    int choice = askQuestionForIntRange(
        "Which micro DB instance class do you want to use? ",
        1, static_cast<int>(instanceClasses.size()));
    dbInstanceClass = instanceClasses[choice - 1];
    return true;
}

//! Routine which deletes resources created by the scenario.
/*!
\sa cleanUpResources()
\param parameterGroupName: A parameter group name, this may be empty.
\param dbInstanceIdentifier: A DB instance identifier, this may be empty.
\param client: 'RDSClient' instance.
\return bool: Successful completion.
*/
bool AwsDoc::RDS::cleanUpResources(const Aws::String &parameterGroupName,
                                   const Aws::String &dbInstanceIdentifier,
                                   const Aws::RDS::RDSClient &client) {
    bool result = true;
    if (!dbInstanceIdentifier.empty()) {
        {
            // 15. Delete the DB instance.
            Aws::RDS::Model::DeleteDBInstanceRequest request;
            request.SetDBInstanceIdentifier(dbInstanceIdentifier);
            request.SetSkipFinalSnapshot(true);
            request.SetDeleteAutomatedBackups(true);

            Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
                client.DeleteDBInstance(request);

            if (outcome.IsSuccess()) {
                std::cout << "DB instance deletion has started."
                    << std::endl;
            }
            else {
                std::cerr << "Error with RDS::DeleteDBInstance. "

```

```

        << outcome.GetError().GetMessage()
        << std::endl;
        result = false;
    }
}

std::cout
    << "Waiting for DB instance to delete before deleting the parameter
group."
    << std::endl;
std::cout << "This may take a while." << std::endl;

int counter = 0;
Aws::RDS::Model::DBInstance dbInstance;
do {
    std::this_thread::sleep_for(std::chrono::seconds(1));
    ++counter;
    if (counter > 800) {
        std::cerr << "Wait for instance to delete timed out after " <<
counter
            << " seconds." << std::endl;
        return false;
    }

    dbInstance = Aws::RDS::Model::DBInstance();
    // 16. Wait for the DB instance to be deleted.
    if (!describeDBInstance(dbInstanceIdentifier, dbInstance, client)) {
        return false;
    }

    if (dbInstance.DBInstanceIdentifierHasBeenSet() && (counter % 20) == 0)
{
        std::cout << "Current DB instance status is '"
            << dbInstance.GetDBInstanceStatus()
            << "' after " << counter << " seconds." << std::endl;
    }
} while (dbInstance.DBInstanceIdentifierHasBeenSet());
}

if (!parameterGroupName.empty()) {
    // 17. Delete the parameter group.
    Aws::RDS::Model::DeleteDBParameterGroupRequest request;
    request.SetDBParameterGroupName(parameterGroupName);

```

```
Aws::RDS::Model::DeleteDBParameterGroupOutcome outcome =
    client.DeleteDBParameterGroup(request);

if (outcome.IsSuccess()) {
    std::cout << "The DB parameter group was successfully deleted."
               << std::endl;
}
else {
    std::cerr << "Error with RDS::DeleteDBParameterGroup. "
               << outcome.GetError().GetMessage()
               << std::endl;
    result = false;
}
}

return result;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 다음 주제를 참조하세요.

- [CreateDBInstance](#)
- [CreateDBParameterGroup](#)
- [CreateDBSnapshot](#)
- [DeleteDBInstance](#)
- [DeleteDBParameterGroup](#)
- [DescribeDBEngineVersions](#)
- [DescribeDBInstances](#)
- [DescribeDBParameterGroups](#)
- [DescribeDBParameters](#)
- [DescribeDBSnapshots](#)
- [DescribeOrderableDBInstanceOptions](#)
- [ModifyDBParameterGroup](#)

## 작업

### CreateDBInstance

다음 코드 예시는 CreateDBInstance의 사용 방법을 보여줍니다.

SDK for C++

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

Aws::RDS::Model::CreateDBInstanceRequest request;
request.SetDBName(DB_NAME);
request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
request.SetEngine(engineVersion.GetEngine());
request.SetEngineVersion(engineVersion.GetEngineVersion());
request.SetDBInstanceClass(dbInstanceClass);
request.SetStorageType(DB_STORAGE_TYPE);
request.SetAllocatedStorage(DB_ALLOCATED_STORAGE);
request.SetMasterUsername(administratorName);
request.SetMasterUserPassword(administratorPassword);

Aws::RDS::Model::CreateDBInstanceOutcome outcome =
    client.CreateDBInstance(request);

if (outcome.IsSuccess()) {
    std::cout << "The DB instance creation has started."
              << std::endl;
}
else {
    std::cerr << "Error with RDS::CreateDBInstance. "
              << outcome.GetError().GetMessage()
```

```

        << std::endl;
        cleanUpResources(PARAMETER_GROUP_NAME, "", client);
        return false;
    }

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CreateDBInstance](#)를 참조하세요.

## CreateDBParameterGroup

다음 코드 예시는 CreateDBParameterGroup의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::CreateDBParameterGroupRequest request;
    request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
    request.SetDBParameterGroupFamily(dbParameterGroupFamily);
    request.SetDescription("Example parameter group.");

    Aws::RDS::Model::CreateDBParameterGroupOutcome outcome =
        client.CreateDBParameterGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB parameter group was successfully created."
        << std::endl;
    }
    else {
        std::cerr << "Error with RDS::CreateDBParameterGroup. "
        << outcome.GetError().GetMessage()
        << std::endl;
    }

```

```
        return false;
    }
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CreateDBParameterGroup](#)을 참조하세요.

## CreateDBSnapshot

다음 코드 예시는 CreateDBSnapshot의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::CreateDBSnapshotRequest request;
    request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
    request.SetDBSnapshotIdentifier(snapshotID);

    Aws::RDS::Model::CreateDBSnapshotOutcome outcome =
        client.CreateDBSnapshot(request);

    if (outcome.IsSuccess()) {
        std::cout << "Snapshot creation has started."
                  << std::endl;
    }
    else {
        std::cerr << "Error with RDS::CreateDBSnapshot. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
        return false;
    }
```

```
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CreateDBSnapshot](#)을 참조하세요.

## DeleteDBInstance

다음 코드 예시는 DeleteDBInstance의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::DeleteDBInstanceRequest request;
    request.SetDBInstanceIdentifier(dbInstanceIdentifier);
    request.SetSkipFinalSnapshot(true);
    request.SetDeleteAutomatedBackups(true);

    Aws::RDS::Model::DeleteDBInstanceOutcome outcome =
        client.DeleteDBInstance(request);

    if (outcome.IsSuccess()) {
        std::cout << "DB instance deletion has started."
                  << std::endl;
    }
    else {
        std::cerr << "Error with RDS::DeleteDBInstance. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        result = false;
    }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteDBInstance](#)를 참조하세요.

## DeleteDBParameterGroup

다음 코드 예시는 DeleteDBParameterGroup의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

Aws::RDS::Model::DeleteDBParameterGroupRequest request;
request.SetDBParameterGroupName(parameterGroupName);

Aws::RDS::Model::DeleteDBParameterGroupOutcome outcome =
    client.DeleteDBParameterGroup(request);

if (outcome.IsSuccess()) {
    std::cout << "The DB parameter group was successfully deleted."
                << std::endl;
}
else {
    std::cerr << "Error with RDS::DeleteDBParameterGroup. "
                << outcome.GetError().GetMessage()
                << std::endl;
    result = false;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteDBParameterGroup](#)을 참조하세요.

## DescribeDBEngineVersions

다음 코드 예시는 DescribeDBEngineVersions의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::getDBEngineVersions(const Aws::String &engineName,
                                       const Aws::String &parameterGroupFamily,
                                       Aws::Vector<Aws::RDS::Model::DBEngineVersion>
&engineVersionsResult,
                                       const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // Used for pagination.
```

```
do {
    if (!marker.empty()) {
        request.SetMarker(marker);
    }

    Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
        client.DescribeDBEngineVersions(request);

    if (outcome.IsSuccess()) {
        auto &engineVersions = outcome.GetResult().GetDBEngineVersions();
        engineVersionsResult.insert(engineVersionsResult.end(),
engineVersions.begin(),
                                engineVersions.end());
        marker = outcome.GetResult().GetMarker();
    }
    else {
        std::cerr << "Error with RDS::DescribeDBEngineVersionsRequest. "
                    << outcome.GetError().GetMessage()
                    << std::endl;
        return false;
    }

} while (!marker.empty());

return true;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DescribeDBEngineVersions](#)을 참조하세요.

## DescribeDBInstances

다음 코드 예시는 DescribeDBInstances의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

    //! Routine which gets a DB instance description.
    /*!
    \sa describeDBInstance()
    \param dbInstanceIdIdentifier: A DB instance identifier.
    \param instanceResult: The 'DBInstance' object containing the description.
    \param client: 'RDSClient' instance.
    \return bool: Successful completion.
    */
    bool AwsDoc::RDS::describeDBInstance(const Aws::String &dbInstanceIdIdentifier,
                                         Aws::RDS::Model::DBInstance &instanceResult,
                                         const Aws::RDS::RDSClient &client) {
        Aws::RDS::Model::DescribeDBInstancesRequest request;
        request.SetDBInstanceIdIdentifier(dbInstanceIdIdentifier);

        Aws::RDS::Model::DescribeDBInstancesOutcome outcome =
            client.DescribeDBInstances(request);

        bool result = true;
        if (outcome.IsSuccess()) {
            instanceResult = outcome.GetResult().GetDBInstances()[0];
        }
        else if (outcome.GetError().GetErrorType() !=
                 Aws::RDS::RDSErrors::D_B_INSTANCE_NOT_FOUND_FAULT) {
            result = false;
            std::cerr << "Error with RDS::DescribeDBInstances. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }

```

```

    }
    // This example does not log an error if the DB instance does not exist.
    // Instead, instanceResult is set to empty.
    else {
        instanceResult = Aws::RDS::Model::DBInstance();
    }

    return result;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DescribeDBInstances](#)를 참조하세요.

## DescribeDBParameterGroups

다음 코드 예시는 DescribeDBParameterGroups의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

Aws::RDS::Model::DescribeDBParameterGroupsRequest request;
request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);

Aws::RDS::Model::DescribeDBParameterGroupsOutcome outcome =
    client.DescribeDBParameterGroups(request);

if (outcome.IsSuccess()) {
    std::cout << "DB parameter group named '" <<
        PARAMETER_GROUP_NAME << "' already exists." << std::endl;
    dbParameterGroupFamily = outcome.GetResult().GetDBParameterGroups()
[0].GetDBParameterGroupFamily();
}

```

```

    }

    else {
        std::cerr << "Error with RDS::DescribeDBParameterGroups. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        return false;
    }
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DescribeDBParameterGroups](#)를 참조하세요.

## DescribeDBParameters

다음 코드 예시는 DescribeDBParameters의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    Aws::RDS::RDSClient client(clientConfig);

    /*! Routine which gets DB parameters using the 'DescribeDBParameters' api.
    */
    \sa getDBParameters()
    \param parameterGroupName: The name of the parameter group.
    \param namePrefix: Prefix string to filter results by parameter name.
    \param source: A source such as 'user', ignored if empty.
    \param parametersResult: Vector of 'Parameter' objects returned by the routine.
    \param client: 'RDSClient' instance.
    \return bool: Successful completion.
    */
    bool AwsDoc::RDS::getDBParameters(const Aws::String &parameterGroupName,

```

```

const Aws::String &namePrefix,
const Aws::String &source,
Aws::Vector<Aws::RDS::Model::Parameter>
&parametersResult,

const Aws::RDS::RDSClient &client) {
    Aws::String marker;
    do {
        Aws::RDS::Model::DescribeDBParametersRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBParametersOutcome outcome =
            client.DescribeDBParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
                else {
                    parametersResult.push_back(parameter);
                }
            }

            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with RDS::DescribeDBParameters. "
                << outcome.GetError().GetMessage()
                << std::endl;
            return false;
        }
    } while (!marker.empty());

    return true;
}

```

```
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DescribeDBParameters](#) 참조하세요.

## DescribeDBSnapshots

다음 코드 예시는 DescribeDBSnapshots의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

    Aws::RDS::Model::DescribeDBSnapshotsRequest request;
    request.SetDBSnapshotIdentifier(snapshotID);

    Aws::RDS::Model::DescribeDBSnapshotsOutcome outcome =
        client.DescribeDBSnapshots(request);

    if (outcome.IsSuccess()) {
        snapshot = outcome.GetResult().GetDBSnapshots()[0];
    }
    else {
        std::cerr << "Error with RDS::DescribeDBSnapshots. "
                    << outcome.GetError().GetMessage()
                    << std::endl;
        cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
        return false;
    }
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DescribeDBSnapshots](#)를 참조하세요.

## DescribeOrderableDBInstanceOptions

다음 코드 예시는 DescribeOrderableDBInstanceOptions의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

//! Routine which gets available 'micro' DB instance classes, displays the list
//! to the user, and returns the user selection.
/*!
 \sa chooseMicroDBInstanceClass()
 \param engineName: The DB engine name.
 \param engineVersion: The DB engine version.
 \param dbInstanceClass: String for DB instance class chosen by the user.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::chooseMicroDBInstanceClass(const Aws::String &engine,
                                              const Aws::String &engineVersion,
                                              Aws::String &dbInstanceClass,
                                              const Aws::RDS::RDSClient &client) {

    std::vector<Aws::String> instanceClasses;
    Aws::String marker;
    do {
        Aws::RDS::Model::DescribeOrderableDBInstanceOptionsRequest request;
        request.SetEngine(engine);
        request.SetEngineVersion(engineVersion);
        if (!marker.empty()) {
```

```

        request.SetMarker(marker);
    }

    Aws::RDS::Model::DescribeOrderableDBInstanceOptionsOutcome outcome =
        client.DescribeOrderableDBInstanceOptions(request);

    if (outcome.IsSuccess()) {
        const Aws::Vector<Aws::RDS::Model::OrderableDBInstanceOption> &options =
            outcome.GetResult().GetOrderableDBInstanceOptions();
        for (const Aws::RDS::Model::OrderableDBInstanceOption &option: options)
        {
            const Aws::String &instanceClass = option.GetDBInstanceClass();
            if (instanceClass.find("micro") != std::string::npos) {
                if (std::find(instanceClasses.begin(), instanceClasses.end(),
                    instanceClass) ==
                    instanceClasses.end()) {
                    instanceClasses.push_back(instanceClass);
                }
            }
        }
        marker = outcome.GetResult().GetMarker();
    }
    else {
        std::cerr << "Error with RDS::DescribeOrderableDBInstanceOptions. "
            << outcome.GetError().GetMessage()
            << std::endl;
        return false;
    }
} while (!marker.empty());

std::cout << "The available micro DB instance classes for your database engine
are:"
    << std::endl;
for (int i = 0; i < instanceClasses.size(); ++i) {
    std::cout << "    " << i + 1 << ": " << instanceClasses[i] << std::endl;
}

int choice = askQuestionForIntRange(
    "Which micro DB instance class do you want to use? ",
    1, static_cast<int>(instanceClasses.size()));
dbInstanceClass = instanceClasses[choice - 1];
return true;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DescribeOrderableDBInstanceOptions](#)를 참조하세요.

## ModifyDBParameterGroup

다음 코드 예시는 ModifyDBParameterGroup의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::RDS::RDSClient client(clientConfig);

Aws::RDS::Model::ModifyDBParameterGroupRequest request;
request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
request.SetParameters(updateParameters);

Aws::RDS::Model::ModifyDBParameterGroupOutcome outcome =
    client.ModifyDBParameterGroup(request);

if (outcome.IsSuccess()) {
    std::cout << "The DB parameter group was successfully modified."
              << std::endl;
}
else {
    std::cerr << "Error with RDS::ModifyDBParameterGroup. "
              << outcome.GetError().GetMessage()
              << std::endl;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ModifyDBParameterGroup](#)을 참조하세요.

## 시나리오

### Aurora 서버리스 작업 항목 트래커 만들기

다음 코드 예제에서는 Amazon Aurora Serverless 데이터베이스에서 작업 항목을 추적하고 Amazon Simple Email Service(Amazon SES)를 사용하여 보고서를 보내는 웹 애플리케이션을 만드는 방법을 보여줍니다.

#### SDK for C++

Amazon Aurora Serverless 데이터베이스에 저장된 작업 항목을 추적하고 보고하는 웹 애플리케이션을 만드는 방법을 보여줍니다.

Amazon Aurora Serverless 데이터를 쿼리하고 React 애플리케이션에서 사용하도록 C++ REST API를 설정하는 방법에 대한 지침과 전체 소스 코드는 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Aurora
- Amazon RDS
- Amazon RDS 데이터 서비스
- Amazon SES

## SDK for C++를 사용한 Amazon RDS Data Service 예제

다음 코드 예제에서는 Amazon RDS Data Service와 AWS SDK for C++ 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

### Aurora 서버리스 작업 항목 트래커 만들기

다음 코드 예제에서는 Amazon Aurora Serverless 데이터베이스에서 작업 항목을 추적하고 Amazon Simple Email Service(Amazon SES)를 사용하여 보고서를 보내는 웹 애플리케이션을 만드는 방법을 보여줍니다.

#### SDK for C++

Amazon Aurora Serverless 데이터베이스에 저장된 작업 항목을 추적하고 보고하는 웹 애플리케이션을 만드는 방법을 보여줍니다.

Amazon Aurora Serverless 데이터를 쿼리하고 React 애플리케이션에서 사용하도록 C++ REST API를 설정하는 방법에 대한 지침과 전체 소스 코드는 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Aurora
- Amazon RDS
- Amazon RDS 데이터 서비스
- Amazon SES

## SDK for C++를 사용한 Amazon Rekognition 예제

다음 코드 예제에서는 AWS SDK for C++ Amazon Rekognition에서 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

- [시작하기](#)

- [작업](#)
- [시나리오](#)

## 시작하기

### Amazon Rekognition 시작

다음 코드 예제에서는 Amazon Rekognition 사용을 시작하는 방법을 보여줍니다.

SDK for C++

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

CMakeLists.txt CMake 파일의 코드입니다.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS rekognition)

# Set this project's name.
project("hello_rekognition")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
"${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()
```

```
# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory for
    # running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
    # may need to uncomment this
                                # and set the proper subdirectory to the
    executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
        ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
    hello_rekognition.cpp)

target_link_libraries(${PROJECT_NAME}
    ${AWSSDK_LINK_LIBRARIES})
```

hello\_rekognition.cpp 소스 파일의 코드입니다.

```
#include <aws/core/Aws.h>
#include <aws/rekognition/RekognitionClient.h>
#include <aws/rekognition/model/ListCollectionsRequest.h>
#include <iostream>

/*
 * A "Hello Rekognition" starter application which initializes an Amazon
 * Rekognition client and
 * lists the Amazon Rekognition collections in the current account and region.
 *
 * main function
 *
 * Usage: 'hello_rekognition'
 *
 */

int main(int argc, char **argv) {
```

```

    Aws::SDKOptions options;
    // Optional: change the log level for debugging.
    // options.loggingOptions.logLevel = Aws::Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::Rekognition::RekognitionClient rekognitionClient(clientConfig);
        Aws::Rekognition::Model::ListCollectionsRequest request;
        Aws::Rekognition::Model::ListCollectionsOutcome outcome =
            rekognitionClient.ListCollections(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::String>& collectionsIds =
outcome.GetResult().GetCollectionIds();
            if (!collectionsIds.empty()) {
                std::cout << "collectionsIds: " << std::endl;
                for (auto &collectionId : collectionsIds) {
                    std::cout << "- " << collectionId << std::endl;
                }
            } else {
                std::cout << "No collections found" << std::endl;
            }
        } else {
            std::cerr << "Error with ListCollections: " << outcome.GetError()
                << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}

```

- API에 대한 세부 정보는 AWS SDK for C++ API 참조의 [ListCollections](#)를 참조하세요.

## 작업

### DetectLabels

다음 코드 예시는 DetectLabels의 사용 방법을 보여줍니다.

자세한 내용은 [이미지에서 레이블 감지](#)를 참조하세요.

SDK for C++

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
#!/ Detect instances of real-world entities within an image by using Amazon
Rekognition
/*!
\param imageBucket: The Amazon Simple Storage Service (Amazon S3) bucket
containing an image.
\param imageKey: The Amazon S3 key of an image object.
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::Rekognition::detectLabels(const Aws::String &imageBucket,
                                       const Aws::String &imageKey,
                                       const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::Rekognition::RekognitionClient rekognitionClient(clientConfiguration);

    Aws::Rekognition::Model::DetectLabelsRequest request;
    Aws::Rekognition::Model::S3Object s3object;
    s3object.SetBucket(imageBucket);
    s3object.SetName(imageKey);

    Aws::Rekognition::Model::Image image;
    image.SetS3Object(s3object);

    request.SetImage(image);
```

```

const Aws::Rekognition::Model::DetectLabelsOutcome outcome =
rekognitionClient.DetectLabels(request);

if (outcome.IsSuccess()) {
    const Aws::Vector<Aws::Rekognition::Model::Label> &labels =
outcome.GetResult().GetLabels();
    if (labels.empty()) {
        std::cout << "No labels detected" << std::endl;
    } else {
        for (const Aws::Rekognition::Model::Label &label: labels) {
            std::cout << label.GetName() << ": " << label.GetConfidence() <<
std::endl;
        }
    }
} else {
    std::cerr << "Error while detecting labels: '"
        << outcome.GetError().GetMessage()
        << "'" << std::endl;
}

return outcome.IsSuccess();
}

```

- API에 대한 세부 정보는 AWS SDK for C++ API 참조의 [DetectLabels](#)를 참조하세요.

## 시나리오

### 사진을 관리하기 위한 서버리스 애플리케이션 만들기

다음 코드 예시에서는 사용자가 레이블을 사용하여 사진을 관리할 수 있는 서버리스 애플리케이션을 생성하는 방법을 보여줍니다.

#### SDK for C++

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 [AWS 커뮤니티](#)의 게시물을 참조하세요.

이 예제에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

## SDK for C++를 사용한 Amazon S3용 예제

다음 코드 예제에서는 AWS SDK for C++ Amazon S3에서 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](#)
- [기본 사항](#)
- [작업](#)
- [시나리오](#)

## 시작하기

Hello Amazon S3

다음 코드 예제에서는 Amazon S3 사용을 시작하는 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

CMakeLists.txt CMake 파일의 코드입니다.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS s3)

# Set this project's name.
project("hello_s3")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
"${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory for
    running and debugging.

    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you may
    need to uncomment this
    # and set the proper subdirectory to the executables' location.
```

```

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
    ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
    hello_s3.cpp)

target_link_libraries(${PROJECT_NAME}
    ${AWSSDK_LINK_LIBRARIES})

```

hello\_s3.cpp 소스 파일의 코드입니다.

```

#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <iostream>
#include <aws/core/auth/AWSCredentialsProviderChain.h>
using namespace Aws;
using namespace Aws::Auth;

/*
 * A "Hello S3" starter application which initializes an Amazon Simple Storage
 * Service (Amazon S3) client
 * and lists the Amazon S3 buckets in the selected region.
 *
 * main function
 *
 * Usage: 'hello_s3'
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
    }
}

```

```

    // You don't normally have to test that you are authenticated. But the S3
    // service permits anonymous requests, thus the s3Client will return "success" and 0
    // buckets even if you are unauthenticated, which can be confusing to a new user.
    auto provider = Aws::MakeShared<DefaultAWSCredentialsProviderChain>("alloc-
tag");
    auto creds = provider->GetAWSCredentials();
    if (creds.IsEmpty()) {
        std::cerr << "Failed authentication" << std::endl;
    }

    Aws::S3::S3Client s3Client(clientConfig);
    auto outcome = s3Client.ListBuckets();

    if (!outcome.IsSuccess()) {
        std::cerr << "Failed with error: " << outcome.GetError() << std::endl;
        result = 1;
    } else {
        std::cout << "Found " << outcome.GetResult().GetBuckets().size()
            << " buckets\n";
        for (auto &bucket: outcome.GetResult().GetBuckets()) {
            std::cout << bucket.GetName() << std::endl;
        }
    }
}

Aws::ShutdownAPI(options); // Should only be called once.
return result;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ListBuckets](#)를 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 C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예제 리포지토리](#)에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
#include <iostream>
#include <aws/core/Aws.h>
#include <aws/s3/S3Client.h>
#include <aws/s3/model/CopyObjectRequest.h>
#include <aws/s3/model/CreateBucketRequest.h>
#include <aws/s3/model/DeleteBucketRequest.h>
#include <aws/s3/model/DeleteObjectRequest.h>
#include <aws/s3/model/GetObjectRequest.h>
#include <aws/s3/model/ListObjectsV2Request.h>
#include <aws/s3/model/PutObjectRequest.h>
#include <aws/s3/model/BucketLocationConstraint.h>
#include <aws/s3/model/CreateBucketConfiguration.h>
#include <aws/core/Utils/UUID.h>
#include <aws/core/Utils/StringUtils.h>
#include <aws/core/Utils/memory/stl/AWSAllocator.h>
#include <fstream>
#include "s3_examples.h"

namespace AwsDoc {
    namespace S3 {

        //! Delete an S3 bucket.
        /*!
         * \param bucketName: The S3 bucket's name.
         * \param client: An S3 client.
         * \return bool: Function succeeded.
         */
        static bool
        deleteBucket(const Aws::String &bucketName, Aws::S3::S3Client &client);
    }
}
```

```

    //! Delete an object in an S3 bucket.
    /*!
    \param bucketName: The S3 bucket's name.
    \param key: The key for the object in the S3 bucket.
    \param client: An S3 client.
    \return bool: Function succeeded.
    */
    static bool
    deleteObjectFromBucket(const Aws::String &bucketName, const Aws::String
&key,
                           Aws::S3::S3Client &client);
}

//! Scenario to create, copy, and delete S3 buckets and objects.
/*!
\param bucketNamePrefix: A prefix for a bucket name.
\param uploadFilePath: Path to file to upload to an Amazon S3 bucket.
\param saveFilePath: Path for saving a downloaded S3 object.
\param clientConfig: Aws client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::S3::S3_GettingStartedScenario(const Aws::String &bucketNamePrefix,
      const Aws::String &uploadFilePath,
      const Aws::String &saveFilePath,
      const Aws::Client::ClientConfiguration
&clientConfig) {

    Aws::S3::S3Client client(clientConfig);

    // Create a unique bucket name which is only temporary and will be deleted.
    // Format: <bucketNamePrefix> + "-" + lowercase UUID.
    Aws::String uuid = Aws::Utils::UUID::RandomUUID();
    Aws::String bucketName = bucketNamePrefix +
        Aws::Utils::StringUtils::ToLower(uuid.c_str());

    // 1. Create a bucket.
    {
        Aws::S3::Model::CreateBucketRequest request;
        request.SetBucket(bucketName);

        if (clientConfig.region != Aws::Region::US_EAST_1) {
            Aws::S3::Model::CreateBucketConfiguration createBucketConfiguration;

```

```

        createBucketConfiguration.WithLocationConstraint(
            Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                clientConfig.region));
        request.WithCreateBucketConfiguration(createBucketConfiguration);
    }

    Aws::S3::Model::CreateBucketOutcome outcome = client.CreateBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: createBucket: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
        return false;
    } else {
        std::cout << "Created the bucket, '" << bucketName <<
            "', in the region, '" << clientConfig.region << "'." <<
std::endl;
    }
}

// 2. Upload a local file to the bucket.
Aws::String key = "key-for-test";
{
    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    std::shared_ptr<Aws::FStream> input_data =
        Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
            uploadFilePath,
            std::ios_base::in |
            std::ios_base::binary);

    if (!input_data->is_open()) {
        std::cerr << "Error: unable to open file, '" << uploadFilePath << "'."
            << std::endl;
        AwsDoc::S3::deleteBucket(bucketName, client);
        return false;
    }

    request.SetBody(input_data);
}

```

```

    Aws::S3::Model::PutObjectOutcome outcome =
        client.PutObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putObject: " <<
            outcome.GetError().GetMessage() << std::endl;
        AwsDoc::S3::deleteObjectFromBucket(bucketName, key, client);
        AwsDoc::S3::deleteBucket(bucketName, client);
        return false;
    } else {
        std::cout << "Added the object with the key, '" << key
            << "', to the bucket, '"
            << bucketName << "'." << std::endl;
    }
}

// 3. Download the object to a local file.
{
    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    Aws::S3::Model::GetObjectOutcome outcome =
        client.GetObject(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObject: " <<
            err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        std::cout << "Downloaded the object with the key, '" << key
            << "', in the bucket, '"
            << bucketName << "'." << std::endl;

        Aws::IOStream &ioStream = outcome.GetResultWithOwnership().
            GetBody();
        Aws::OFStream outStream(saveFilePath,
                                std::ios_base::out | std::ios_base::binary);
        if (!outStream.is_open()) {
            std::cout << "Error: unable to open file, '" << saveFilePath << "'."
                << std::endl;
        } else {
            outStream << ioStream.rdbuf();

```

```

        std::cout << "Wrote the downloaded object to the file '"
                    << saveFilePath << "'. " << std::endl;
    }
}

// 4. Copy the object to a different "folder" in the bucket.
Aws::String copiedToKey = "test-folder/" + key;
{
    Aws::S3::Model::CopyObjectRequest request;
    request.WithBucket(bucketName)
           .WithKey(copiedToKey)
           .WithCopySource(bucketName + "/" + key);

    Aws::S3::Model::CopyObjectOutcome outcome =
        client.CopyObject(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error: copyObject: " <<
                    outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Copied the object with the key, '" << key
                  << "', to the key, '" << copiedToKey
                  << "', in the bucket, '" << bucketName << "'. " << std::endl;
    }
}

// 5. List objects in the bucket.
{
    Aws::S3::Model::ListObjectsV2Request request;
    request.WithBucket(bucketName);

    Aws::String continuationToken;
    Aws::Vector<Aws::S3::Model::Object> allObjects;

    do {
        if (!continuationToken.empty()) {
            request.SetContinuationToken(continuationToken);
        }
        Aws::S3::Model::ListObjectsV2Outcome outcome = client.ListObjectsV2(
            request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error: ListObjects: " <<
                        outcome.GetError().GetMessage() << std::endl;

```

```

        break;
    } else {
        Aws::Vector<Aws::S3::Model::Object> objects =
            outcome.GetResult().GetContents();
        allObjects.insert(allObjects.end(), objects.begin(), objects.end());
        continuationToken = outcome.GetResult().GetContinuationToken();
    }
} while (!continuationToken.empty());

std::cout << allObjects.size() << " objects in the bucket, '" << bucketName
    << "':" << std::endl;

for (Aws::S3::Model::Object &object: allObjects) {
    std::cout << "    '" << object.GetKey() << "'" << std::endl;
}
}

// 6. Delete all objects in the bucket.
// All objects in the bucket must be deleted before deleting the bucket.
AwsDoc::S3::deleteObjectFromBucket(bucketName, copiedToKey, client);
AwsDoc::S3::deleteObjectFromBucket(bucketName, key, client);

// 7. Delete the bucket.
return AwsDoc::S3::deleteBucket(bucketName, client);
}

bool AwsDoc::S3::deleteObjectFromBucket(const Aws::String &bucketName,
                                       const Aws::String &key,
                                       Aws::S3::S3Client &client) {
    Aws::S3::Model::DeleteObjectRequest request;
    request.SetBucket(bucketName);
    request.SetKey(key);

    Aws::S3::Model::DeleteObjectOutcome outcome =
        client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: deleteObject: " <<
            outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Deleted the object with the key, '" << key
            << "', from the bucket, '"
            << bucketName << "':" << std::endl;
    }
}

```

```

    return outcome.IsSuccess();
}

bool
AwsDoc::S3::deleteBucket(const Aws::String &bucketName, Aws::S3::S3Client &client) {
    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
        client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucket: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
        std::cout << "Deleted the bucket, '" << bucketName << "'." << std::endl;
    }
    return outcome.IsSuccess();
}

```

• API 세부 정보는 AWS SDK for C++ API 참조의 다음 항목을 참조하세요.

- [CopyObject](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

## 작업

### AbortMultipartUpload

다음 코드 예시는 AbortMultipartUpload의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Abort a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/

bool AwsDoc::S3::abortMultipartUpload(const Aws::String &bucket,
                                       const Aws::String &key,
                                       const Aws::String &uploadID,
                                       const Aws::S3::S3Client &client) {
    Aws::S3::Model::AbortMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);

    Aws::S3::Model::AbortMultipartUploadOutcome outcome =
        client.AbortMultipartUpload(request);

    if (outcome.IsSuccess()) {
        std::cout << "Multipart upload aborted." << std::endl;
    } else {
        std::cerr << "Error aborting multipart upload: " <<
outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [AbortMultipartUpload](#)를 참조하세요.

## CompleteMultipartUpload

다음 코드 예시는 CompleteMultipartUpload의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
#!/ Complete a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param parts: A vector of CompleteParts.
    \param client: The S3 client instance used to perform the upload operation.
    \return CompleteMultipartUploadOutcome: The request outcome.
*/
Aws::S3::Model::CompleteMultipartUploadOutcome
AwsDoc::S3::completeMultipartUpload(const Aws::String &bucket,

const Aws::String &key,

const Aws::String &uploadID,

const Aws::Vector<Aws::S3::Model::CompletedPart> &parts,

const Aws::S3::S3Client &client) {
    Aws::S3::Model::CompletedMultipartUpload completedMultipartUpload;
    completedMultipartUpload.SetParts(parts);

    Aws::S3::Model::CompleteMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetMultipartUpload(completedMultipartUpload);

    Aws::S3::Model::CompleteMultipartUploadOutcome outcome =
        client.CompleteMultipartUpload(request);
```

```

    if (!outcome.IsSuccess()) {
        std::cerr << "Error completing multipart upload: " <<
outcome.GetError().GetMessage() << std::endl;
    }
    return outcome;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CompleteMultipartUpload](#)를 참조하세요.

## CopyObject

다음 코드 예시는 CopyObject의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

bool AwsDoc::S3::copyObject(const Aws::String &objectKey, const Aws::String
&fromBucket, const Aws::String &toBucket,
                        const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CopyObjectRequest request;

    request.WithCopySource(fromBucket + "/" + objectKey)
        .WithKey(objectKey)
        .WithBucket(toBucket);

    Aws::S3::Model::CopyObjectOutcome outcome = client.CopyObject(request);
    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: copyObject: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully copied " << objectKey << " from " << fromBucket
<<
            " to " << toBucket << "." << std::endl;
    }
}

```

```

    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CopyObject](#)를 참조하세요.

## CreateBucket

다음 코드 예시는 CreateBucket의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

bool AwsDoc::S3::createBucket(const Aws::String &bucketName,
                               const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::CreateBucketRequest request;
    request.SetBucket(bucketName);

    if (clientConfig.region != "us-east-1") {
        Aws::S3::Model::CreateBucketConfiguration createBucketConfig;
        createBucketConfig.SetLocationConstraint(
            Aws::S3::Model::BucketLocationConstraintMapper::GetBucketLocationConstraintForName(
                clientConfig.region));
        request.SetCreateBucketConfiguration(createBucketConfig);
    }

    Aws::S3::Model::CreateBucketOutcome outcome = client.CreateBucket(request);
    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: createBucket: " <<
                    err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
        std::cout << "Created bucket " << bucketName <<

```

```

        " in the specified AWS Region." << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CreateBucket](#)을 참조하세요.

## CreateMultipartUpload

다음 코드 예시는 CreateMultipartUpload의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Create a multipart upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param client: The S3 client instance used to perform the upload operation.
    \return Aws::String: Upload ID or empty string if failed.
*/
Aws::String
AwsDoc::S3::createMultipartUpload(const Aws::String &bucket, const Aws::String &key,
                                   Aws::S3::Model::ChecksumAlgorithm
                                   checksumAlgorithm,
                                   const Aws::S3::S3Client &client) {
    Aws::S3::Model::CreateMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }
}

```

```

    Aws::S3::Model::CreateMultipartUploadOutcome outcome =
        client.CreateMultipartUpload(request);

    Aws::String uploadID;
    if (outcome.IsSuccess()) {
        uploadID = outcome.GetResult().GetUploadId();
    } else {
        std::cerr << "Error creating multipart upload: " <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return uploadID;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CreateMultipartUpload](#)를 참조하세요.

## DeleteBucket

다음 코드 예시는 DeleteBucket의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

bool AwsDoc::S3::deleteBucket(const Aws::String &bucketName,
                               const Aws::S3::S3ClientConfiguration &clientConfig) {

    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::DeleteBucketRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketOutcome outcome =
        client.DeleteBucket(request);

    if (!outcome.IsSuccess()) {

```

```

        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucket: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
        std::cout << "The bucket was deleted" << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteBucket](#)을 참조하세요.

## DeleteBucketPolicy

다음 코드 예시는 DeleteBucketPolicy의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

bool AwsDoc::S3::deleteBucketPolicy(const Aws::String &bucketName,
                                     const Aws::S3::S3ClientConfiguration
&clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::DeleteBucketPolicyRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketPolicyOutcome outcome =
    client.DeleteBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: deleteBucketPolicy: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {

```

```

        std::cout << "Policy was deleted from the bucket." << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteBucketPolicy](#)를 참조하세요.

## DeleteBucketWebsite

다음 코드 예시는 DeleteBucketWebsite의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

bool AwsDoc::S3::deleteBucketWebsite(const Aws::String &bucketName,
                                       const Aws::S3::S3ClientConfiguration
&clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::DeleteBucketWebsiteOutcome outcome =
        client.DeleteBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: deleteBucketWebsite: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
        std::cout << "Website configuration was removed." << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteBucketWebsite](#)를 참조하세요.

## DeleteObject

다음 코드 예시는 DeleteObject의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::S3::deleteObject(const Aws::String &objectKey,
                               const Aws::String &fromBucket,
                               const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectRequest request;

    request.WithKey(objectKey)
           .WithBucket(fromBucket);

    Aws::S3::Model::DeleteObjectOutcome outcome =
        client.DeleteObject(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error: deleteObject: " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully deleted the object." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteObject](#)를 참조하세요.

## DeleteObjects

다음 코드 예시는 DeleteObjects의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::S3::deleteObjects(const std::vector<Aws::String> &objectKeys,
                               const Aws::String &fromBucket,
                               const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    Aws::S3::Model::DeleteObjectsRequest request;

    Aws::S3::Model::Delete deleteObject;
    for (const Aws::String &objectKey: objectKeys) {
        deleteObject.AddObjects(Aws::S3::Model::ObjectIdentifier().WithKey(objectKey));
    }

    request.SetDelete(deleteObject);
    request.SetBucket(fromBucket);

    Aws::S3::Model::DeleteObjectsOutcome outcome =
        client.DeleteObjects(request);

    if (!outcome.IsSuccess()) {
        auto err = outcome.GetError();
        std::cerr << "Error deleting objects. " <<
            err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully deleted the objects.";
        for (size_t i = 0; i < objectKeys.size(); ++i) {
            std::cout << objectKeys[i];
            if (i < objectKeys.size() - 1) {
                std::cout << ", ";
            }
        }
    }
}
```

```

        std::cout << " from bucket " << fromBucket << "." << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteObjects](#)를 참조하세요.

## GetBucketAcl

다음 코드 예시는 GetBucketAcl의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

bool AwsDoc::S3::getBucketAcl(const Aws::String &bucketName,
                              const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketAclRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketAclOutcome outcome =
        s3Client.GetBucketAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getBucketAcl: "
                  << err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        Aws::Vector<Aws::S3::Model::Grant> grants =
            outcome.GetResult().GetGrants();

        for (auto it = grants.begin(); it != grants.end(); it++) {
            Aws::S3::Model::Grant grant = *it;

```

```

        Aws::S3::Model::Grantee grantee = grant.GetGrantee();

        std::cout << "For bucket " << bucketName << ": "
                  << std::endl << std::endl;

        if (grantee.TypeHasBeenSet()) {
            std::cout << "Type:          "
                      << getGranteeTypeString(grantee.GetType()) << std::endl;
        }

        if (grantee.DisplayNameHasBeenSet()) {
            std::cout << "Display name:  "
                      << grantee.GetDisplayName() << std::endl;
        }

        if (grantee.EmailAddressHasBeenSet()) {
            std::cout << "Email address: "
                      << grantee.GetEmailAddress() << std::endl;
        }

        if (grantee.IDHasBeenSet()) {
            std::cout << "ID:           "
                      << grantee.GetID() << std::endl;
        }

        if (grantee.URIHasBeenSet()) {
            std::cout << "URI:          "
                      << grantee.GetURI() << std::endl;
        }

        std::cout << "Permission:    " <<
                  getPermissionString(grant.GetPermission()) <<
                  std::endl << std::endl;
    }
}

return outcome.IsSuccess();
}

//! Routine which converts a built-in type enumeration to a human-readable string.
/*!
 \param type: Type enumeration.
 \return String: Human-readable string.
 */

```

```

Aws::String getGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:
            return "Predefined Amazon S3 group";
        case Aws::S3::Model::Type::NOT_SET:
            return "Not set";
        default:
            return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable string.
/*!
    \param permission: Permission enumeration.
    \return String: Human-readable string.
*/

Aws::String getPermissionString(const Aws::S3::Model::Permission &permission) {
    switch (permission) {
        case Aws::S3::Model::Permission::FULL_CONTROL:
            return "Can list objects in this bucket, create/overwrite/delete "
                "objects in this bucket, and read/write this "
                "bucket's permissions";
        case Aws::S3::Model::Permission::NOT_SET:
            return "Permission not set";
        case Aws::S3::Model::Permission::READ:
            return "Can list objects in this bucket";
        case Aws::S3::Model::Permission::READ_ACP:
            return "Can read this bucket's permissions";
        case Aws::S3::Model::Permission::WRITE:
            return "Can create, overwrite, and delete objects in this bucket";
        case Aws::S3::Model::Permission::WRITE_ACP:
            return "Can write this bucket's permissions";
        default:
            return "Permission unknown";
    }

    return "Permission unknown";
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetBucketAcl](#)을 참조하세요.

## GetBucketPolicy

다음 코드 예시는 GetBucketPolicy의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::S3::getBucketPolicy(const Aws::String &bucketName,
                                const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketPolicyRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketPolicyOutcome outcome =
        s3Client.GetBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getBucketPolicy: "
                  << err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {
        Aws::StringStream policy_stream;
        Aws::String line;

        outcome.GetResult().GetPolicy() >> line;
        policy_stream << line;

        std::cout << "Retrieve the policy for bucket '" << bucketName << "':\n\n" <<
            policy_stream.str() << std::endl;
    }
}
```

```

    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetBucketPolicy](#)를 참조하세요.

## GetBucketWebsite

다음 코드 예시는 GetBucketWebsite의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

bool AwsDoc::S3::getWebsiteConfig(const Aws::String &bucketName,
                                   const Aws::S3::S3ClientConfiguration
                                   &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetBucketWebsiteRequest request;
    request.SetBucket(bucketName);

    Aws::S3::Model::GetBucketWebsiteOutcome outcome =
        s3Client.GetBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();

        std::cerr << "Error: GetBucketWebsite: "
                   << err.GetMessage() << std::endl;
    } else {
        Aws::S3::Model::GetBucketWebsiteResult websiteResult = outcome.GetResult();

        std::cout << "Success: GetBucketWebsite: "
                  << std::endl << std::endl
                  << "For bucket '" << bucketName << "':"

```

```

        << std::endl
        << "Index page : "
        << websiteResult.GetIndexDocument().GetSuffix()
        << std::endl
        << "Error page: "
        << websiteResult.GetErrorDocument().GetKey()
        << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetBucketWebsite](#)를 참조하세요.

## GetObject

다음 코드 예시는 GetObject의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

bool AwsDoc::S3::getObject(const Aws::String &objectKey,
                           const Aws::String &fromBucket,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::GetObjectRequest request;
    request.SetBucket(fromBucket);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectOutcome outcome =
        client.GetObject(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObject: " <<

```

```

        err.GetExceptionName() << ": " << err.GetMessage() << std::endl;
    } else {
        std::cout << "Successfully retrieved '" << objectKey << "' from '"
            << fromBucket << "'." << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetObject](#)를 참조하세요.

## GetObjectAcl

다음 코드 예시는 GetObjectAcl의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

bool AwsDoc::S3::getObjectAcl(const Aws::String &bucketName,
                              const Aws::String &objectKey,
                              const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::GetObjectAclRequest request;
    request.SetBucket(bucketName);
    request.SetKey(objectKey);

    Aws::S3::Model::GetObjectAclOutcome outcome =
        s3Client.GetObjectAcl(request);

    if (!outcome.IsSuccess()) {
        const Aws::S3::S3Error &err = outcome.GetError();
        std::cerr << "Error: getObjectAcl: "
            << err.GetExceptionName() << ": " << err.GetMessage() <<
std::endl;
    } else {

```

```

    Aws::Vector<Aws::S3::Model::Grant> grants =
        outcome.GetResult().GetGrants();

    for (auto it = grants.begin(); it != grants.end(); it++) {
        std::cout << "For object " << objectKey << ": "
            << std::endl << std::endl;

        Aws::S3::Model::Grant grant = *it;
        Aws::S3::Model::Grantee grantee = grant.GetGrantee();

        if (grantee.TypeHasBeenSet()) {
            std::cout << "Type:          "
                << getGranteeTypeString(grantee.GetType()) << std::endl;
        }

        if (grantee.DisplayNameHasBeenSet()) {
            std::cout << "Display name:  "
                << grantee.GetDisplayName() << std::endl;
        }

        if (grantee.EmailAddressHasBeenSet()) {
            std::cout << "Email address: "
                << grantee.GetEmailAddress() << std::endl;
        }

        if (grantee.IDHasBeenSet()) {
            std::cout << "ID:           "
                << grantee.GetID() << std::endl;
        }

        if (grantee.URIHasBeenSet()) {
            std::cout << "URI:          "
                << grantee.GetURI() << std::endl;
        }

        std::cout << "Permission:    " <<
            getPermissionString(grant.GetPermission()) <<
            std::endl << std::endl;
    }
}

return outcome.IsSuccess();
}

```

```

//! Routine which converts a built-in type enumeration to a human-readable string.
/*!
\param type: Type enumeration.
\return String: Human-readable string
*/
Aws::String getGranteeTypeString(const Aws::S3::Model::Type &type) {
    switch (type) {
        case Aws::S3::Model::Type::AmazonCustomerByEmail:
            return "Email address of an AWS account";
        case Aws::S3::Model::Type::CanonicalUser:
            return "Canonical user ID of an AWS account";
        case Aws::S3::Model::Type::Group:
            return "Predefined Amazon S3 group";
        case Aws::S3::Model::Type::NOT_SET:
            return "Not set";
        default:
            return "Type unknown";
    }
}

//! Routine which converts a built-in type enumeration to a human-readable string.
/*!
\param permission: Permission enumeration.
\return String: Human-readable string
*/
Aws::String getPermissionString(const Aws::S3::Model::Permission &permission) {
    switch (permission) {
        case Aws::S3::Model::Permission::FULL_CONTROL:
            return "Can read this object's data and its metadata, "
                "and read/write this object's permissions";
        case Aws::S3::Model::Permission::NOT_SET:
            return "Permission not set";
        case Aws::S3::Model::Permission::READ:
            return "Can read this object's data and its metadata";
        case Aws::S3::Model::Permission::READ_ACP:
            return "Can read this object's permissions";
        // case Aws::S3::Model::Permission::WRITE // Not applicable.
        case Aws::S3::Model::Permission::WRITE_ACP:
            return "Can write this object's permissions";
        default:
            return "Permission unknown";
    }
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetObjectAcl](#)을 참조하세요.

## GetObjectAttributes

다음 코드 예시는 GetObjectAttributes의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// ! Routine which retrieves the hash value of an object stored in an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object is stored.
    \param key: The unique identifier (key) of the object within the S3 bucket.
    \param hashMethod: The hashing algorithm used to calculate the hash value of the
    object.
    \param[out] hashData: The retrieved hash.
    \param[out] partHashes: The part hashes if available.
    \param client: The S3 client instance used to retrieve the object.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::retrieveObjectHash(const Aws::String &bucket, const Aws::String
    &key,
                                     AwsDoc::S3::HASH_METHOD hashMethod,
                                     Aws::String &hashData,
                                     std::vector<Aws::String> *partHashes,
                                     const Aws::S3::S3Client &client) {
    Aws::S3::Model::GetObjectAttributesRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (hashMethod == MD5) {
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::ETag);
        request.SetObjectAttributes(attributes);
    }
}
```

```

    Aws::S3::Model::GetObjectAttributesOutcome outcome =
client.GetObjectAttributes(
    request);
    if (outcome.IsSuccess()) {
        const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
        hashData = result.GetETag();
    } else {
        std::cerr << "Error retrieving object etag attributes." <<
outcome.GetError().GetMessage() << std::endl;
        return false;
    }
} else { // hashMethod != MD5
    Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
    attributes.push_back(Aws::S3::Model::ObjectAttributes::Checksum);
    request.SetObjectAttributes(attributes);

    Aws::S3::Model::GetObjectAttributesOutcome outcome =
client.GetObjectAttributes(
    request);
    if (outcome.IsSuccess()) {
        const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
        switch (hashMethod) {
            case AwsDoc::S3::DEFAULT: // NOLINT(*-branch-clone)
                break; // Default is not supported.
#pragma clang diagnostic push
#pragma ide diagnostic ignored "UnreachableCode"
            case AwsDoc::S3::MD5:
                break; // MD5 is not supported.
#pragma clang diagnostic pop
            case AwsDoc::S3::SHA1:
                hashData = result.GetChecksum().GetChecksumSHA1();
                break;
            case AwsDoc::S3::SHA256:
                hashData = result.GetChecksum().GetChecksumSHA256();
                break;
            case AwsDoc::S3::CRC32:
                hashData = result.GetChecksum().GetChecksumCRC32();
                break;
            case AwsDoc::S3::CRC32C:
                hashData = result.GetChecksum().GetChecksumCRC32C();
                break;
            default:

```

```

        std::cerr << "Unknown hash method." << std::endl;
        return false;
    }
} else {
    std::cerr << "Error retrieving object checksum attributes." <<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}

if (nullptr != partHashes) {
    attributes.clear();
    attributes.push_back(Aws::S3::Model::ObjectAttributes::ObjectParts);
    request.SetObjectAttributes(attributes);
    outcome = client.GetObjectAttributes(request);
    if (outcome.IsSuccess()) {
        const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
        const Aws::Vector<Aws::S3::Model::ObjectPart> parts =
result.GetObjectParts().GetParts();
        for (const Aws::S3::Model::ObjectPart &part: parts) {
            switch (hashMethod) {
                case AwsDoc::S3::DEFAULT: // Default is not supported.
NOLINT(*-branch-clone)
                    break;
                case AwsDoc::S3::MD5: // MD5 is not supported.
                    break;
                case AwsDoc::S3::SHA1:
                    partHashes->push_back(part.GetChecksumSHA1());
                    break;
                case AwsDoc::S3::SHA256:
                    partHashes->push_back(part.GetChecksumSHA256());
                    break;
                case AwsDoc::S3::CRC32:
                    partHashes->push_back(part.GetChecksumCRC32());
                    break;
                case AwsDoc::S3::CRC32C:
                    partHashes->push_back(part.GetChecksumCRC32C());
                    break;
                default:
                    std::cerr << "Unknown hash method." << std::endl;
                    return false;
            }
        }
    } else {

```

```

        std::cerr << "Error retrieving object attributes for object parts."
<<
        outcome.GetError().GetMessage() << std::endl;
        return false;
    }
}
}

return true;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetObjectAttributes](#)를 참조하세요.

## ListBuckets

다음 코드 예시는 ListBuckets의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

bool AwsDoc::S3::listBuckets(const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    auto outcome = client.ListBuckets();

    bool result = true;
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed with error: " << outcome.GetError() << std::endl;
        result = false;
    } else {
        std::cout << "Found " << outcome.GetResult().GetBuckets().size() << "
buckets\n";
        for (auto &&b: outcome.GetResult().GetBuckets()) {
            std::cout << b.GetName() << std::endl;
        }
    }
}

```

```
    return result;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ListBuckets](#)를 참조하세요.

## ListObjectsV2

다음 코드 예시는 ListObjectsV2의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::S3::listObjects(const Aws::String &bucketName,
                             Aws::Vector<Aws::String> &keysResult,
                             const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::ListObjectsV2Request request;
    request.WithBucket(bucketName);

    Aws::String continuationToken; // Used for pagination.
    Aws::Vector<Aws::S3::Model::Object> allObjects;

    do {
        if (!continuationToken.empty()) {
            request.SetContinuationToken(continuationToken);
        }

        auto outcome = s3Client.ListObjectsV2(request);

        if (!outcome.IsSuccess()) {
            std::cerr << "Error: listObjects: " <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        } else {
```

```

        Aws::Vector<Aws::S3::Model::Object> objects =
            outcome.GetResult().GetContents();

        allObjects.insert(allObjects.end(), objects.begin(), objects.end());
        continuationToken = outcome.GetResult().GetNextContinuationToken();
    }
} while (!continuationToken.empty());

std::cout << allObjects.size() << " object(s) found:" << std::endl;

for (const auto &object: allObjects) {
    std::cout << " " << object.GetKey() << std::endl;
    keysResult.push_back(object.GetKey());
}

return true;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ListObjectsV2](#)를 참조하세요.

## PutBucketAcl

다음 코드 예시는 PutBucketAcl의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

bool AwsDoc::S3::putBucketAcl(const Aws::String &bucketName, const Aws::String
&ownerID,
                                const Aws::String &granteePermission,
                                const Aws::String &granteeType, const Aws::String
&granteeID,
                                const Aws::String &granteeEmailAddress,
                                const Aws::String &granteeURI, const
Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

```

```
Aws::S3::Model::Owner owner;
owner.SetID(ownerID);

Aws::S3::Model::Grantee grantee;
grantee.SetType(setGranteeType(granteeType));

if (!granteeEmailAddress.empty()) {
    grantee.SetEmailAddress(granteeEmailAddress);
}

if (!granteeID.empty()) {
    grantee.SetID(granteeID);
}

if (!granteeURI.empty()) {
    grantee.SetURI(granteeURI);
}

Aws::S3::Model::Grant grant;
grant.SetGrantee(grantee);
grant.SetPermission(setGranteePermission(granteePermission));

Aws::Vector<Aws::S3::Model::Grant> grants;
grants.push_back(grant);

Aws::S3::Model::AccessControlPolicy acp;
acp.SetOwner(owner);
acp.SetGrants(grants);

Aws::S3::Model::PutBucketAclRequest request;
request.SetAccessControlPolicy(acp);
request.SetBucket(bucketName);

Aws::S3::Model::PutBucketAclOutcome outcome =
    s3Client.PutBucketAcl(request);

if (!outcome.IsSuccess()) {
    const Aws::S3::S3Error &error = outcome.GetError();

    std::cerr << "Error: putBucketAcl: " << error.GetExceptionName()
               << " - " << error.GetMessage() << std::endl;
} else {
    std::cout << "Successfully added an ACL to the bucket '" << bucketName
```

```

        << "'.'" << std::endl;
    }

    return outcome.IsSuccess();
}

//! Routine which converts a human-readable string to a built-in type enumeration.
/*!
 \param access: Human readable string.
 \return Permission: A Permission enum.
 */

Aws::S3::Model::Permission setGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a human-readable string to a built-in type enumeration.
/*!
 \param type: Human readable string.
 \return Type: Type enumeration
 */

Aws::S3::Model::Type setGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")
        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")
        return Aws::S3::Model::Type::CanonicalUser;
    if (type == "Group")
        return Aws::S3::Model::Type::Group;
    return Aws::S3::Model::Type::NOT_SET;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [PutBucketAcl](#)을 참조하세요.

## PutBucketPolicy

다음 코드 예시는 PutBucketPolicy의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::S3::putBucketPolicy(const Aws::String &bucketName,
                                const Aws::String &policyBody,
                                const Aws::S3::S3ClientConfiguration &clientConfig)
{
    Aws::S3::S3Client s3Client(clientConfig);

    std::shared_ptr<Aws::StringStream> request_body =
        Aws::MakeShared<Aws::StringStream>("");
    *request_body << policyBody;

    Aws::S3::Model::PutBucketPolicyRequest request;
    request.SetBucket(bucketName);
    request.SetBody(request_body);

    Aws::S3::Model::PutBucketPolicyOutcome outcome =
        s3Client.PutBucketPolicy(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putBucketPolicy: "
                  << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Set the following policy body for the bucket '" <<
                  bucketName << "':" << std::endl << std::endl;
        std::cout << policyBody << std::endl;
    }

    return outcome.IsSuccess();
}

//! Build a policy JSON string.
```

```

/ *!
\param userArn: Aws user Amazon Resource Name (ARN).
    For more information, see https://docs.aws.amazon.com/IAM/latest/UserGuide/
reference_identifiers.html#identifiers-arns.
\param bucketName: Name of a bucket.
\return String: Policy as JSON string.
*/

Aws::String getPolicyString(const Aws::String &userArn,
                           const Aws::String &bucketName) {
    return
        "{\n"
        "  \"Version\": \"2012-10-17\", \n"
        "  \"Statement\": [\n"
        "    {\n"
        "      \"Sid\": \"1\", \n"
        "      \"Effect\": \"Allow\", \n"
        "      \"Principal\": {\n"
        "        \"AWS\": \""
        + userArn +
        "\"\n"
        "      }, \n"
        "      \"Action\": [ \"s3:getObject\" ], \n"
        "      \"Resource\": [ \"arn:aws:s3:::"
        + bucketName +
        "\"/*\" ] \n"
        "    } \n"
        "  ] \n"
        "}";
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [PutBucketPolicy](#)를 참조하세요.

## PutBucketWebsite

다음 코드 예시는 PutBucketWebsite의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::S3::putWebsiteConfig(const Aws::String &bucketName,
                                   const Aws::String &indexPath, const Aws::String
                                   &errorPage,
                                   const Aws::S3::S3ClientConfiguration
                                   &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::IndexDocument indexDocument;
    indexDocument.SetSuffix(indexPath);

    Aws::S3::Model::ErrorDocument errorDocument;
    errorDocument.SetKey(errorPage);

    Aws::S3::Model::WebsiteConfiguration websiteConfiguration;
    websiteConfiguration.SetIndexDocument(indexDocument);
    websiteConfiguration.SetErrorDocument(errorDocument);

    Aws::S3::Model::PutBucketWebsiteRequest request;
    request.SetBucket(bucketName);
    request.SetWebsiteConfiguration(websiteConfiguration);

    Aws::S3::Model::PutBucketWebsiteOutcome outcome =
        client.PutBucketWebsite(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: PutBucketWebsite: "
                  << outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Success: Set website configuration for bucket '"
                  << bucketName << "'." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [PutBucketWebsite](#)를 참조하세요.

## PutObject

다음 코드 예시는 PutObject의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::S3::putObject(const Aws::String &bucketName,
                           const Aws::String &fileName,
                           const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucketName);
    //We are using the name of the file as the key for the object in the bucket.
    //However, this is just a string and can be set according to your retrieval
    needs.
    request.SetKey(fileName);

    std::shared_ptr<Aws::IOStream> inputData =
        Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                       fileName.c_str(),
                                       std::ios_base::in |
std::ios_base::binary);

    if (!*inputData) {
        std::cerr << "Error unable to read file " << fileName << std::endl;
        return false;
    }

    request.SetBody(inputData);

    Aws::S3::Model::PutObjectOutcome outcome =
```

```

        s3Client.PutObject(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error: putObject: " <<
            outcome.GetError().GetMessage() << std::endl;
    } else {
        std::cout << "Added object '" << fileName << "' to bucket '"
            << bucketName << "'.";
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [PutObject](#)를 참조하세요.

## PutObjectAcl

다음 코드 예시는 PutObjectAcl의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

bool AwsDoc::S3::putObjectAcl(const Aws::String &bucketName, const Aws::String
    &objectKey, const Aws::String &ownerID,
                                const Aws::String &granteePermission, const
    Aws::String &granteeType,
                                const Aws::String &granteeID, const Aws::String
    &granteeEmailAddress,
                                const Aws::String &granteeURI, const
    Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client s3Client(clientConfig);

    Aws::S3::Model::Owner owner;
    owner.SetID(ownerID);

    Aws::S3::Model::Grantee grantee;

```

```
grantee.SetType(setGranteeType(granteeType));

if (!granteeEmailAddress.empty()) {
    grantee.SetEmailAddress(granteeEmailAddress);
}

if (!granteeID.empty()) {
    grantee.SetID(granteeID);
}

if (!granteeURI.empty()) {
    grantee.SetURI(granteeURI);
}

Aws::S3::Model::Grant grant;
grant.SetGrantee(grantee);
grant.SetPermission(setGranteePermission(granteePermission));

Aws::Vector<Aws::S3::Model::Grant> grants;
grants.push_back(grant);

Aws::S3::Model::AccessControlPolicy acp;
acp.SetOwner(owner);
acp.SetGrants(grants);

Aws::S3::Model::PutObjectAclRequest request;
request.SetAccessControlPolicy(acp);
request.SetBucket(bucketName);
request.SetKey(objectKey);

Aws::S3::Model::PutObjectAclOutcome outcome =
    s3Client.PutObjectAcl(request);

if (!outcome.IsSuccess()) {
    auto error = outcome.GetError();
    std::cerr << "Error: putObjectAcl: " << error.GetExceptionName()
        << " - " << error.GetMessage() << std::endl;
} else {
    std::cout << "Successfully added an ACL to the object '" << objectKey
        << "' in the bucket '" << bucketName << ".'" << std::endl;
}

return outcome.IsSuccess();
}
```

```

//! Routine which converts a human-readable string to a built-in type enumeration.
/*!
\param access: Human readable string.
\return Permission: Permission enumeration.
*/
Aws::S3::Model::Permission setGranteePermission(const Aws::String &access) {
    if (access == "FULL_CONTROL")
        return Aws::S3::Model::Permission::FULL_CONTROL;
    if (access == "WRITE")
        return Aws::S3::Model::Permission::WRITE;
    if (access == "READ")
        return Aws::S3::Model::Permission::READ;
    if (access == "WRITE_ACP")
        return Aws::S3::Model::Permission::WRITE_ACP;
    if (access == "READ_ACP")
        return Aws::S3::Model::Permission::READ_ACP;
    return Aws::S3::Model::Permission::NOT_SET;
}

//! Routine which converts a human-readable string to a built-in type enumeration.
/*!
\param type: Human readable string.
\return Type: Type enumeration.
*/
Aws::S3::Model::Type setGranteeType(const Aws::String &type) {
    if (type == "Amazon customer by email")
        return Aws::S3::Model::Type::AmazonCustomerByEmail;
    if (type == "Canonical user")
        return Aws::S3::Model::Type::CanonicalUser;
    if (type == "Group")
        return Aws::S3::Model::Type::Group;
    return Aws::S3::Model::Type::NOT_SET;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [PutObjectAcl](#)을 참조하세요.

## UploadPart

다음 코드 예시는 UploadPart의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Upload a part to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param partNumber:
    \param checksumAlgorithm: Checksum algorithm, ignored when NOT_SET.
    \param calculatedHash: A data integrity hash to set, depending on the checksum
algorithm,
                        ignored when it is an empty string.
    \param body: An shared_ptr IOSTream of the data to be uploaded.
    \param client: The S3 client instance used to perform the upload operation.
    \return UploadPartOutcome: The outcome.
*/

Aws::S3::Model::UploadPartOutcome AwsDoc::S3::uploadPart(const Aws::String &bucket,
                                                         const Aws::String &key,
                                                         const Aws::String
                                                         &uploadID,
                                                         int partNumber,
                                                         Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm,
                                                         const Aws::String
                                                         &calculatedHash,
                                                         const
                                                         std::shared_ptr<Aws::IOStream> &body,
                                                         const Aws::S3::S3Client
                                                         &client) {
    Aws::S3::Model::UploadPartRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetPartNumber(partNumber);
    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {

```

```
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }
    request.SetBody(body);

    if (!calculatedHash.empty()) {
        switch (checksumAlgorithm) {
            case Aws::S3::Model::ChecksumAlgorithm::NOT_SET:
                request.SetContentMD5(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32:
                request.SetChecksumCRC32(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32C:
                request.SetChecksumCRC32C(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA1:
                request.SetChecksumSHA1(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA256:
                request.SetChecksumSHA256(calculatedHash);
                break;
        }
    }

    return client.UploadPart(request);
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [UploadPart](#)를 참조하세요.

## 시나리오

### 미리 서명된 URL 생성

다음 코드 예제는 Amazon S3에 대해 미리 서명된 URL을 생성하고 객체를 업로드하는 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

미리 서명된 URL을 생성하여 객체를 다운로드합니다.

```

//! Routine which demonstrates creating a pre-signed URL to download an object from
an
//! Amazon Simple Storage Service (Amazon S3) bucket.
/*!
\param bucketName: Name of the bucket.
\param key: Name of an object key.
\param expirationSeconds: Expiration in seconds for pre-signed URL.
\param clientConfig: Aws client configuration.
\return Aws::String: A pre-signed URL.
*/
Aws::String AwsDoc::S3::generatePreSignedGetObjectUrl(const Aws::String &bucketName,
                                                         const Aws::String &key,
                                                         uint64_t expirationSeconds,
                                                         const
                                                         Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    return client.GeneratePresignedUrl(bucketName, key,
    Aws::Http::HttpMethod::HTTP_GET,
    expirationSeconds);
}

```

libcurl을 사용하여 다운로드합니다.

```

static size_t myCurlWriteBack(char *buffer, size_t size, size_t nitems, void
*userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    if (nitems > 0) {
        str->write(buffer, size * nitems);
    }
    return size * nitems;
}

```

```
}

//! Utility routine to test getObject with a pre-signed URL.
/*!
  \param presignedURL: A pre-signed URL to get an object from a bucket.
  \param resultString: A string to hold the result.
  \return bool: Function succeeded.
*/
bool AwsDoc::S3::getObjectWithPresignedObjectUrl(const Aws::String &presignedURL,
                                                  Aws::String &resultString) {
    CURL *curl = curl_easy_init();
    CURLcode result;

    std::stringstream outWriteString;

    result = curl_easy_setopt(curl, CURLOPT_WRITEDATA, &outWriteString);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEDATA " << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, myCurlWriteBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_WRITEFUNCTION" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_URL, presignedURL.c_str());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_URL" << std::endl;
        return false;
    }

    result = curl_easy_perform(curl);

    if (result != CURLE_OK) {
        std::cerr << "Failed to perform CURL request" << std::endl;
        return false;
    }

    resultString = outWriteString.str();
}
```

```

    if (resultString.find("<?xml") == 0) {
        std::cerr << "Failed to get object, response:\n" << resultString <<
std::endl;
        return false;
    }

    return true;
}

```

미리 서명된 URL을 생성하여 객체를 업로드합니다.

```

//! Routine which demonstrates creating a pre-signed URL to upload an object to an
//! Amazon Simple Storage Service (Amazon S3) bucket.
/*!
    \param bucketName: Name of the bucket.
    \param key: Name of an object key.
    \param clientConfig: Aws client configuration.
    \return Aws::String: A pre-signed URL.
*/
Aws::String AwsDoc::S3::generatePreSignedPutObjectUrl(const Aws::String &bucketName,
                                                       const Aws::String &key,
                                                       uint64_t expirationSeconds,
                                                       const
    Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);
    return client.GeneratePresignedUrl(bucketName, key,
    Aws::Http::HttpMethod::HTTP_PUT,
                                   expirationSeconds);
}

```

libcurl을 사용하여 업로드합니다.

```

static size_t myCurlReadBack(char *buffer, size_t size, size_t nitems, void
*userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    str->read(buffer, size * nitems);

    return str->gcount();
}

```

```

static size_t myCurlWriteBack(char *buffer, size_t size, size_t nitems, void
*userdata) {
    Aws::StringStream *str = (Aws::StringStream *) userdata;

    if (nitems > 0) {
        str->write(buffer, size * nitems);
    }
    return size * nitems;
}

//! Utility routine to test putObject with a pre-signed URL.
/*!
    \param presignedURL: A pre-signed URL to put an object in a bucket.
    \param data: Body of the putObject request.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::PutStringWithPresignedObjectURL(const Aws::String &presignedURL,
                                                  const Aws::String &data) {

    CURL *curl = curl_easy_init();
    CURLcode result;

    Aws::StringStream readStringStream;
    readStringStream << data;
    result = curl_easy_setopt(curl, CURLOPT_READFUNCTION, myCurlReadBack);

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_READFUNCTION" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_READDATA, &readStringStream);
    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_READDATA" << std::endl;
        return false;
    }

    result = curl_easy_setopt(curl, CURLOPT_INFILESIZE_LARGE,
                              (curl_off_t) data.size());

    if (result != CURLE_OK) {
        std::cerr << "Failed to set CURLOPT_INFILESIZE_LARGE" << std::endl;
        return false;
    }
}

```

```
result = curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, myCurlWriteBack);

if (result != CURLE_OK) {
    std::cerr << "Failed to set CURLOPT_WRITEFUNCTION" << std::endl;
    return false;
}

std::stringstream outWriteString;

result = curl_easy_setopt(curl, CURLOPT_WRITEDATA, &outWriteString);

if (result != CURLE_OK) {
    std::cerr << "Failed to set CURLOPT_WRITEDATA " << std::endl;
    return false;
}

result = curl_easy_setopt(curl, CURLOPT_URL, presignedURL.c_str());

if (result != CURLE_OK) {
    std::cerr << "Failed to set CURLOPT_URL" << std::endl;
    return false;
}

result = curl_easy_setopt(curl, CURLOPT_UPLOAD, 1L);

if (result != CURLE_OK) {
    std::cerr << "Failed to set CURLOPT_PUT" << std::endl;
    return false;
}

result = curl_easy_perform(curl);

if (result != CURLE_OK) {
    std::cerr << "Failed to perform CURL request" << std::endl;
    return false;
}

std::string outString = outWriteString.str();
if (outString.empty()) {
    std::cout << "Successfully put object." << std::endl;
    return true;
} else {
    std::cout << "A server error was encountered, output:\n" << outString
```

```

        << std::endl;
    return false;
}
}

```

## 사진을 관리하기 위한 서버리스 애플리케이션 만들기

다음 코드 예시에서는 사용자가 레이블을 사용하여 사진을 관리할 수 있는 서버리스 애플리케이션을 생성하는 방법을 보여줍니다.

### SDK for C++

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 [AWS 커뮤니티](#)의 게시물을 참조하세요.

이 예제에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

### Amazon S3 객체 무결성 작업

다음 코드 예시에서는 S3 객체 무결성 기능을 사용하는 방법을 보여줍니다.

### SDK for C++

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon S3 객체 무결성 기능을 시연하는 대화형 시나리오를 실행합니다.

```

//! Routine which runs the S3 object integrity workflow.
/*!
    \param clientConfig: Aws client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::s3ObjectIntegrityWorkflow(
    const Aws::S3::S3ClientConfiguration &clientConfiguration) {

    /*
     * Create a large file to be used for multipart uploads.
     */
    if (!createLargeFileIfNotExists()) {
        std::cerr << "Workflow exiting because large file creation failed." <<
std::endl;
        return false;
    }

    Aws::String bucketName = TEST_BUCKET_PREFIX;
    bucketName += Aws::Utils::UUID::RandomUUID();
    bucketName = Aws::Utils::StringUtils::ToLower(bucketName.c_str());

    bucketName.resize(std::min(bucketName.size(), MAX_BUCKET_NAME_LENGTH));

    introductoryExplanations(bucketName);

    if (!AwsDoc::S3::createBucket(bucketName, clientConfiguration)) {
        std::cerr << "Workflow exiting because bucket creation failed." <<
std::endl;
        return false;
    }

    Aws::S3::S3ClientConfiguration s3ClientConfiguration(clientConfiguration);
    std::shared_ptr<Aws::S3::S3Client> client =
    Aws::MakeShared<Aws::S3::S3Client>("S3Client", s3ClientConfiguration);

    printAsterisksLine();
    std::cout << "Choose from one of the following checksum algorithms."
        << std::endl;

    for (HASH_METHOD hashMethod = DEFAULT; hashMethod <= SHA256; ++hashMethod) {
        std::cout << " " << hashMethod << " - " << stringForHashMethod(hashMethod)
            << std::endl;
    }
}

```

```

    }

    HASH_METHOD chosenHashMethod = askQuestionForIntRange("Enter an index: ",
    DEFAULT,

                                                                SHA256);

    gUseCalculatedChecksum = !askYesNoQuestion(
        "Let the SDK calculate the checksum for you? (y/n) ");

    printAsterisksLine();

    std::cout << "The workflow will now upload a file using PutObject."
        << std::endl;
    std::cout << "Object integrity will be verified using the "
        << stringForHashMethod(chosenHashMethod) << " algorithm."
        << std::endl;
    if (gUseCalculatedChecksum) {
        std::cout
            << "A checksum computed by this workflow will be used for object
integrity verification,"
            << std::endl;
        std::cout << "except for the TransferManager upload." << std::endl;
    } else {
        std::cout
            << "A checksum computed by the SDK will be used for object integrity
verification."
            << std::endl;
    }

    pressEnterToContinue();
    printAsterisksLine();

    std::shared_ptr<Aws::IOStream> inputData =
        Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
            TEST_FILE,
            std::ios_base::in |
            std::ios_base::binary);

    if (!*inputData) {
        std::cerr << "Error unable to read file " << TEST_FILE << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

```

```

Hasher hasher;
HASH_METHOD putObjectHashMethod = chosenHashMethod;
if (putObjectHashMethod == DEFAULT) {
    putObjectHashMethod = MD5; // MD5 is the default hash method for PutObject.

    std::cout << "The default checksum algorithm for PutObject is "
                << stringForHashMethod(putObjectHashMethod)
                << std::endl;
}

// Demonstrate in code how the hash is computed.
if (!hasher.calculateObjectHash(*inputData, putObjectHashMethod)) {
    std::cerr << "Error calculating hash for file " << TEST_FILE << std::endl;
    cleanUp(bucketName, clientConfiguration);
    return false;
}
Aws::String key = stringForHashMethod(putObjectHashMethod);
key += "_";
key += TEST_FILE_KEY;
Aws::String localHash = hasher.getBase64HashString();

// Upload the object with PutObject
if (!putObjectWithHash(bucketName, key, localHash, putObjectHashMethod,
                      inputData, chosenHashMethod == DEFAULT,
                      *client)) {
    std::cerr << "Error putting file " << TEST_FILE << " to bucket "
                << bucketName << " with key " << key << std::endl;
    cleanUp(bucketName, clientConfiguration);
    return false;
}

Aws::String retrievedHash;
if (!retrieveObjectHash(bucketName, key,
                      putObjectHashMethod, retrievedHash,
                      nullptr, *client)) {
    std::cerr << "Error getting file " << TEST_FILE << " from bucket "
                << bucketName << " with key " << key << std::endl;
    cleanUp(bucketName, clientConfiguration);
    return false;
}

explainPutObjectResults();
verifyHashingResults(retrievedHash, hasher,

```

```

        "PutObject upload", putObjectHashMethod);

    printAsterisksLine();
    pressEnterToContinue();

    key = "tr_";
    key += stringForHashMethod(chosenHashMethod) + "_" + MULTI_PART_TEST_FILE;

    introductoryTransferManagerUploadExplanations(key);

    HASH_METHOD transferManagerHashMethod = chosenHashMethod;
    if (transferManagerHashMethod == DEFAULT) {
        transferManagerHashMethod = CRC32; // The default hash method for the
        TransferManager is CRC32.

        std::cout << "The default checksum algorithm for TransferManager is "
                    << stringForHashMethod(transferManagerHashMethod)
                    << std::endl;
    }

    // Upload the large file using the transfer manager.
    if (!doTransferManagerUpload(bucketName, key, transferManagerHashMethod,
        chosenHashMethod == DEFAULT,
                                client)) {
        std::cerr << "Exiting because of an error in doTransferManagerUpload." <<
std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    std::vector<Aws::String> retrievedTransferManagerPartHashes;
    Aws::String retrievedTransferManagerFinalHash;

    // Retrieve all the hashes for the TransferManager upload.
    if (!retrieveObjectHash(bucketName, key,
                            transferManagerHashMethod,
                            retrievedTransferManagerFinalHash,
                            &retrievedTransferManagerPartHashes, *client)) {
        std::cerr << "Exiting because of an error in retrieveObjectHash for
TransferManager." << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

```

```

    AwsDoc::S3::Hasher locallyCalculatedFinalHash;
    std::vector<Aws::String> locallyCalculatedPartHashes;

    // Calculate the hashes locally to demonstrate how TransferManager hashes are
    computed.
    if (!calculatePartHashesForFile(transferManagerHashMethod, MULTI_PART_TEST_FILE,
                                    UPLOAD_BUFFER_SIZE,
                                    locallyCalculatedFinalHash,
                                    locallyCalculatedPartHashes)) {
        std::cerr << "Exiting because of an error in calculatePartHashesForFile." <<
std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    verifyHashingResults(retrievedTransferManagerFinalHash,
                        locallyCalculatedFinalHash, "TransferManager upload",
                        transferManagerHashMethod,
                        retrievedTransferManagerPartHashes,
                        locallyCalculatedPartHashes);

    printAsterisksLine();

    key = "mp_";
    key += stringForHashMethod(chosenHashMethod) + "_" + MULTI_PART_TEST_FILE;

    multiPartUploadExplanations(key, chosenHashMethod);

    pressEnterToContinue();

    std::shared_ptr<Aws::IOStream> largeFileInputData =
        Aws::MakeShared<Aws::FStream>("SampleAllocationTag",
                                    MULTI_PART_TEST_FILE,
                                    std::ios_base::in |
                                    std::ios_base::binary);

    if (!largeFileInputData->good()) {
        std::cerr << "Error unable to read file " << TEST_FILE << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    HASH_METHOD multipartUploadHashMethod = chosenHashMethod;

```

```

    if (multipartUploadHashMethod == DEFAULT) {
        multipartUploadHashMethod = MD5; // The default hash method for multipart
        uploads is MD5.

        std::cout << "The default checksum algorithm for multipart upload is "
                    << stringForHashMethod(putObjectHashMethod)
                    << std::endl;
    }

    AwsDoc::S3::Hasher hashData;
    std::vector<Aws::String> partHashes;

    if (!doMultipartUpload(bucketName, key,
                           multipartUploadHashMethod,
                           largeFileInputData, chosenHashMethod == DEFAULT,
                           hashData,
                           partHashes,
                           *client)) {
        std::cerr << "Exiting because of an error in doMultipartUpload." <<
std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    std::cout << "Finished multipart upload of with hash method " <<
                stringForHashMethod(multipartUploadHashMethod) << std::endl;

    std::cout << "Now we will retrieve the checksums from the server." << std::endl;

    retrievedHash.clear();
    std::vector<Aws::String> retrievedPartHashes;
    if (!retrieveObjectHash(bucketName, key,
                           multipartUploadHashMethod,
                           retrievedHash, &retrievedPartHashes, *client)) {
        std::cerr << "Exiting because of an error in retrieveObjectHash for
multipart." << std::endl;
        cleanUp(bucketName, clientConfiguration);
        return false;
    }

    verifyHashingResults(retrievedHash, hashData, "MultiPart upload",
                        multipartUploadHashMethod,
                        retrievedPartHashes, partHashes);

```

```

    printAsterisksLine();

    if (askYesNoQuestion("Would you like to delete the resources created in this
workflow? (y/n)")) {
        return cleanUp(bucketName, clientConfiguration);
    } else {
        std::cout << "The bucket " << bucketName << " was not deleted." <<
std::endl;
        return true;
    }
}

//! Routine which uploads an object to an S3 bucket with different object integrity
hashing methods.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param hashData: The hash value that will be associated with the uploaded object.
    \param hashMethod: The hashing algorithm to use when calculating the hash value.
    \param body: The data content of the object being uploaded.
    \param useDefaultHashMethod: A flag indicating whether to use the default hash
method or the one specified in the hashMethod parameter.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::putObjectWithHash(const Aws::String &bucket, const Aws::String
&key,

                                const Aws::String &hashData,
                                AwsDoc::S3::HASH_METHOD hashMethod,
                                const std::shared_ptr<Aws::IOStream> &body,
                                bool useDefaultHashMethod,
                                const Aws::S3::S3Client &client) {
    Aws::S3::Model::PutObjectRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    if (!useDefaultHashMethod) {
        if (hashMethod != MD5) {

request.SetChecksumAlgorithm(getChecksumAlgorithmForHashMethod(hashMethod));
        }
    }

    if (gUseCalculatedChecksum) {
        switch (hashMethod) {

```

```

        case AwsDoc::S3::MD5:
            request.SetContentMD5(hashData);
            break;
        case AwsDoc::S3::SHA1:
            request.SetChecksumSHA1(hashData);
            break;
        case AwsDoc::S3::SHA256:
            request.SetChecksumSHA256(hashData);
            break;
        case AwsDoc::S3::CRC32:
            request.SetChecksumCRC32(hashData);
            break;
        case AwsDoc::S3::CRC32C:
            request.SetChecksumCRC32C(hashData);
            break;
        default:
            std::cerr << "Unknown hash method." << std::endl;
            return false;
    }
}

request.SetBody(body);
Aws::S3::Model::PutObjectOutcome outcome = client.PutObject(request);
body->seekg(0, body->beg);
if (outcome.IsSuccess()) {
    std::cout << "Object successfully uploaded." << std::endl;
} else {
    std::cerr << "Error uploading object." <<
        outcome.GetError().GetMessage() << std::endl;
}
return outcome.IsSuccess();
}

// ! Routine which retrieves the hash value of an object stored in an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object is stored.
    \param key: The unique identifier (key) of the object within the S3 bucket.
    \param hashMethod: The hashing algorithm used to calculate the hash value of the
    object.
    \param[out] hashData: The retrieved hash.
    \param[out] partHashes: The part hashes if available.
    \param client: The S3 client instance used to retrieve the object.
    \return bool: Function succeeded.
*/

```

```

bool AwsDoc::S3::retrieveObjectHash(const Aws::String &bucket, const Aws::String
&key,
                                     AwsDoc::S3::HASH_METHOD hashMethod,
                                     Aws::String &hashData,
                                     std::vector<Aws::String> *partHashes,
                                     const Aws::S3::S3Client &client) {
    Aws::S3::Model::GetObjectAttributesRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (hashMethod == MD5) {
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::ETag);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
client.GetObjectAttributes(
    request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
            hashData = result.GetETag();
        } else {
            std::cerr << "Error retrieving object etag attributes." <<
outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    } else { // hashMethod != MD5
        Aws::Vector<Aws::S3::Model::ObjectAttributes> attributes;
        attributes.push_back(Aws::S3::Model::ObjectAttributes::Checksum);
        request.SetObjectAttributes(attributes);

        Aws::S3::Model::GetObjectAttributesOutcome outcome =
client.GetObjectAttributes(
    request);
        if (outcome.IsSuccess()) {
            const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
            switch (hashMethod) {
                case AwsDoc::S3::DEFAULT: // NOLINT(*-branch-clone)
                    break; // Default is not supported.
#pragma clang diagnostic push
#pragma ide diagnostic ignored "UnreachableCode"
                case AwsDoc::S3::MD5:

```

```

        break; // MD5 is not supported.
#pragma clang diagnostic pop
        case AwsDoc::S3::SHA1:
            hashData = result.GetChecksum().GetChecksumSHA1();
            break;
        case AwsDoc::S3::SHA256:
            hashData = result.GetChecksum().GetChecksumSHA256();
            break;
        case AwsDoc::S3::CRC32:
            hashData = result.GetChecksum().GetChecksumCRC32();
            break;
        case AwsDoc::S3::CRC32C:
            hashData = result.GetChecksum().GetChecksumCRC32C();
            break;
        default:
            std::cerr << "Unknown hash method." << std::endl;
            return false;
    }
} else {
    std::cerr << "Error retrieving object checksum attributes." <<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}

if (nullptr != partHashes) {
    attributes.clear();
    attributes.push_back(Aws::S3::Model::ObjectAttributes::ObjectParts);
    request.SetObjectAttributes(attributes);
    outcome = client.GetObjectAttributes(request);
    if (outcome.IsSuccess()) {
        const Aws::S3::Model::GetObjectAttributesResult &result =
outcome.GetResult();
        const Aws::Vector<Aws::S3::Model::ObjectPart> parts =
result.GetObjectParts().GetParts();
        for (const Aws::S3::Model::ObjectPart &part: parts) {
            switch (hashMethod) {
                case AwsDoc::S3::DEFAULT: // Default is not supported.
NOLINT(*-branch-clone)
                    break;
                case AwsDoc::S3::MD5: // MD5 is not supported.
                    break;
                case AwsDoc::S3::SHA1:
                    partHashes->push_back(part.GetChecksumSHA1());
                    break;
            }
        }
    }
}

```

```

        case AwsDoc::S3::SHA256:
            partHashes->push_back(part.GetChecksumSHA256());
            break;
        case AwsDoc::S3::CRC32:
            partHashes->push_back(part.GetChecksumCRC32());
            break;
        case AwsDoc::S3::CRC32C:
            partHashes->push_back(part.GetChecksumCRC32C());
            break;
        default:
            std::cerr << "Unknown hash method." << std::endl;
            return false;
    }
}
} else {
    std::cerr << "Error retrieving object attributes for object parts."
<<
        outcome.GetError().GetMessage() << std::endl;
    return false;
}
}
}

return true;
}

//! Verifies the hashing results between the retrieved and local hashes.
/*!
 \param retrievedHash The hash value retrieved from the remote source.
 \param localHash The hash value calculated locally.
 \param uploadtype The type of upload (e.g., "multipart", "single-part").
 \param hashMethod The hashing method used (e.g., MD5, SHA-256).
 \param retrievedPartHashes (Optional) The list of hashes for the individual parts
 retrieved from the remote source.
 \param localPartHashes (Optional) The list of hashes for the individual parts
 calculated locally.
 */
void AwsDoc::S3::verifyHashingResults(const Aws::String &retrievedHash,
                                     const Hasher &localHash,
                                     const Aws::String &uploadtype,
                                     HASH_METHOD hashMethod,
                                     const std::vector<Aws::String>
&retrievedPartHashes,

```

```

const std::vector<Aws::String>
&localPartHashes) {
    std::cout << "For " << uploadtype << " retrieved hash is " << retrievedHash <<
std::endl;
    if (!retrievedPartHashes.empty()) {
        std::cout << retrievedPartHashes.size() << " part hash(es) were also
retrieved."
                << std::endl;
        for (auto &retrievedPartHash: retrievedPartHashes) {
            std::cout << " Part hash " << retrievedPartHash << std::endl;
        }
    }
    Aws::String hashString;
    if (hashMethod == MD5) {
        hashString = localHash.getHexHashString();
        if (!localPartHashes.empty()) {
            hashString += "-" + std::to_string(localPartHashes.size());
        }
    } else {
        hashString = localHash.getBase64HashString();
    }

    bool allMatch = true;
    if (hashString != retrievedHash) {
        std::cerr << "For " << uploadtype << ", the main hashes do not match" <<
std::endl;
        std::cerr << "Local hash- " << hashString << "" << std::endl;
        std::cerr << "Remote hash - " << retrievedHash << "" << std::endl;
        allMatch = false;
    }

    if (hashMethod != MD5) {
        if (localPartHashes.size() != retrievedPartHashes.size()) {
            std::cerr << "For " << uploadtype << ", the number of part hashes do not
match" << std::endl;
            std::cerr << "Local number of hashes- " << localPartHashes.size() <<
""
                    << std::endl;
            std::cerr << "Remote number of hashes - "
                    << retrievedPartHashes.size()
                    << "" << std::endl;
        }

        for (int i = 0; i < localPartHashes.size(); ++i) {

```

```

        if (localPartHashes[i] != retrievedPartHashes[i]) {
            std::cerr << "For " << uploadtype << ", the part hashes do not match
for part " << i + 1
                << "." << std::endl;
            std::cerr << "Local hash- '" << localPartHashes[i] << "'"
                << std::endl;
            std::cerr << "Remote hash - '" << retrievedPartHashes[i] << "'"
                << std::endl;
            allMatch = false;
        }
    }
}

if (allMatch) {
    std::cout << "For " << uploadtype << ", locally and remotely calculated
hashes all match!" << std::endl;
}
}

static void transferManagerErrorCallback(const Aws::Transfer::TransferManager *,
                                         const std::shared_ptr<const
    Aws::Transfer::TransferHandle> &,
                                         const
    Aws::Client::AWSError<Aws::S3::S3Errors> &err) {
    std::cerr << "Error during transfer: '" << err.GetMessage() << "'" << std::endl;
}

static void transferManagerStatusCallback(const Aws::Transfer::TransferManager *,
                                         const std::shared_ptr<const
    Aws::Transfer::TransferHandle> &handle) {
    if (handle->GetStatus() == Aws::Transfer::TransferStatus::IN_PROGRESS) {
        std::cout << "Bytes transferred: " << handle->GetBytesTransferred() <<
std::endl;
    }
}

}

//! Routine which uploads an object to an S3 bucket using the AWS C++ SDK's Transfer
Manager.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param hashMethod: The hashing algorithm to use when calculating the hash value.

```

```

    \param useDefaultHashMethod: A flag indicating whether to use the default hash
    method or the one specified in the hashMethod parameter.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/
bool
AwsDoc::S3::doTransferManagerUpload(const Aws::String &bucket, const Aws::String
&key,

                                   AwsDoc::S3::HASH_METHOD hashMethod,
                                   bool useDefaultHashMethod,
                                   const std::shared_ptr<Aws::S3::S3Client>

&client) {
    std::shared_ptr<Aws::Utils::Threading::PooledThreadExecutor> executor =
    Aws::MakeShared<Aws::Utils::Threading::PooledThreadExecutor>(
        "executor", 25);
    Aws::Transfer::TransferManagerConfiguration transfer_config(executor.get());
    transfer_config.s3Client = client;
    transfer_config.bufferSize = UPLOAD_BUFFER_SIZE;
    if (!useDefaultHashMethod) {
        if (hashMethod == MD5) {
            transfer_config.computeContentMD5 = true;
        } else {
            transfer_config.checksumAlgorithm = getChecksumAlgorithmForHashMethod(
                hashMethod);
        }
    }
    transfer_config.errorCallback = transferManagerErrorCallback;
    transfer_config.transferStatusUpdatedCallback = transferManagerStatusCallback;

    std::shared_ptr<Aws::Transfer::TransferManager> transfer_manager =
    Aws::Transfer::TransferManager::Create(
        transfer_config);

    std::cout << "Uploading the file..." << std::endl;
    std::shared_ptr<Aws::Transfer::TransferHandle> uploadHandle = transfer_manager-
    >UploadFile(MULTI_PART_TEST_FILE,

               bucket, key,

               "text/plain",

               Aws::Map<Aws::String, Aws::String>());
    uploadHandle->WaitUntilFinished();
    bool success =

```

```

        uploadHandle->GetStatus() == Aws::Transfer::TransferStatus::COMPLETED;
    if (!success) {
        Aws::Client::AWSError<Aws::S3::S3Errors> err = uploadHandle->GetLastError();
        std::cerr << "File upload failed: " << err.GetMessage() << std::endl;
    }

    return success;
}

//! Routine which calculates the hash values for each part of a file being uploaded
to an S3 bucket.
/*!
    \param hashMethod: The hashing algorithm to use when calculating the hash values.
    \param fileName: The path to the file for which the part hashes will be
    calculated.
    \param bufferSize: The size of the buffer to use when reading the file.
    \param[out] hashDataResult: The Hasher object that will store the concatenated
    hash value.
    \param[out] partHashes: The vector that will store the calculated hash values for
    each part of the file.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::calculatePartHashesForFile(AwsDoc::S3::HASH_METHOD hashMethod,
                                             const Aws::String &fileName,
                                             size_t bufferSize,
                                             AwsDoc::S3::Hasher &hashDataResult,
                                             std::vector<Aws::String> &partHashes) {
    std::ifstream fileStream(fileName.c_str(), std::ifstream::binary);
    fileStream.seekg(0, std::ifstream::end);
    size_t objectSize = fileStream.tellg();
    fileStream.seekg(0, std::ifstream::beg);
    std::vector<unsigned char> totalHashBuffer;
    size_t uploadedBytes = 0;

    while (uploadedBytes < objectSize) {
        std::vector<unsigned char> buffer(bufferSize);
        std::streamsize bytesToRead =
static_cast<std::streamsize>(std::min(buffer.size(), objectSize - uploadedBytes));
        fileStream.read((char *) buffer.data(), bytesToRead);
        Aws::Utils::Stream::PreallocatedStreamBuf
preallocatedStreamBuf(buffer.data(),
        bytesToRead);
    }
}

```

```

        std::shared_ptr<Aws::IOStream> body =
            Aws::MakeShared<Aws::IOStream>("SampleAllocationTag",
                                           &preallocatedStreamBuf);

        Hasher hasher;
        if (!hasher.calculateObjectHash(*body, hashMethod)) {
            std::cerr << "Error calculating hash." << std::endl;
            return false;
        }
        Aws::String base64HashString = hasher.getBase64HashString();
        partHashes.push_back(base64HashString);

        Aws::Utils::ByteBuffer hashBuffer = hasher.getByteBufferHash();

        totalHashBuffer.insert(totalHashBuffer.end(),
                                hashBuffer.GetUnderlyingData(),
                                hashBuffer.GetUnderlyingData() +
                                hashBuffer.GetLength());

        uploadedBytes += bytesToRead;
    }

    return hashDataResult.calculateObjectHash(totalHashBuffer, hashMethod);
}

//! Create a multipart upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param client: The S3 client instance used to perform the upload operation.
    \return Aws::String: Upload ID or empty string if failed.
*/
Aws::String
AwsDoc::S3::createMultipartUpload(const Aws::String &bucket, const Aws::String &key,
                                   Aws::S3::Model::ChecksumAlgorithm
                                   checksumAlgorithm,
                                   const Aws::S3::S3Client &client) {
    Aws::S3::Model::CreateMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);

    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }
}

```

```

    Aws::S3::Model::CreateMultipartUploadOutcome outcome =
        client.CreateMultipartUpload(request);

    Aws::String uploadID;
    if (outcome.IsSuccess()) {
        uploadID = outcome.GetResult().GetUploadId();
    } else {
        std::cerr << "Error creating multipart upload: " <<
outcome.GetError().GetMessage() << std::endl;
    }

    return uploadID;
}

//! Upload a part to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param partNumber:
    \param checksumAlgorithm: Checksum algorithm, ignored when NOT_SET.
    \param calculatedHash: A data integrity hash to set, depending on the checksum
algorithm,
                        ignored when it is an empty string.
    \param body: An shared_ptr IOSTream of the data to be uploaded.
    \param client: The S3 client instance used to perform the upload operation.
    \return UploadPartOutcome: The outcome.
*/

Aws::S3::Model::UploadPartOutcome AwsDoc::S3::uploadPart(const Aws::String &bucket,
                                                         const Aws::String &key,
                                                         const Aws::String
&uploadID,
                                                         int partNumber,

                                                         Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm,
                                                         const Aws::String
&calculatedHash,
                                                         const
std::shared_ptr<Aws::IOStream> &body,
                                                         const Aws::S3::S3Client
&client) {
    Aws::S3::Model::UploadPartRequest request;
    request.SetBucket(bucket);

```

```

    request.SetKey(key);
    request.SetUploadId(uploadID);
    request.SetPartNumber(partNumber);
    if (checksumAlgorithm != Aws::S3::Model::ChecksumAlgorithm::NOT_SET) {
        request.SetChecksumAlgorithm(checksumAlgorithm);
    }
    request.SetBody(body);

    if (!calculatedHash.empty()) {
        switch (checksumAlgorithm) {
            case Aws::S3::Model::ChecksumAlgorithm::NOT_SET:
                request.SetContentMD5(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32:
                request.SetChecksumCRC32(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::CRC32C:
                request.SetChecksumCRC32C(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA1:
                request.SetChecksumSHA1(calculatedHash);
                break;
            case Aws::S3::Model::ChecksumAlgorithm::SHA256:
                request.SetChecksumSHA256(calculatedHash);
                break;
        }
    }

    return client.UploadPart(request);
}

//! Abort a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/

bool AwsDoc::S3::abortMultipartUpload(const Aws::String &bucket,
                                       const Aws::String &key,
                                       const Aws::String &uploadID,
                                       const Aws::S3::S3Client &client) {

```

```

    Aws::S3::Model::AbortMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);

    Aws::S3::Model::AbortMultipartUploadOutcome outcome =
        client.AbortMultipartUpload(request);

    if (outcome.IsSuccess()) {
        std::cout << "Multipart upload aborted." << std::endl;
    } else {
        std::cerr << "Error aborting multipart upload: " <<
outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

//! Complete a multipart upload to an S3 bucket.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param uploadID: An upload ID string.
    \param parts: A vector of CompleteParts.
    \param client: The S3 client instance used to perform the upload operation.
    \return CompleteMultipartUploadOutcome: The request outcome.
*/
Aws::S3::Model::CompleteMultipartUploadOutcome
AwsDoc::S3::completeMultipartUpload(const Aws::String &bucket,

const Aws::String &key,

const Aws::String &uploadID,

const Aws::Vector<Aws::S3::Model::CompletedPart> &parts,

const Aws::S3::S3Client &client) {
    Aws::S3::Model::CompletedMultipartUpload completedMultipartUpload;
    completedMultipartUpload.SetParts(parts);

    Aws::S3::Model::CompleteMultipartUploadRequest request;
    request.SetBucket(bucket);
    request.SetKey(key);
    request.SetUploadId(uploadID);

```

```

    request.SetMultipartUpload(completedMultipartUpload);

    Aws::S3::Model::CompleteMultipartUploadOutcome outcome =
        client.CompleteMultipartUpload(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error completing multipart upload: " <<
outcome.GetError().GetMessage() << std::endl;
    }
    return outcome;
}

//! Routine which performs a multi-part upload.
/*!
    \param bucket: The name of the S3 bucket where the object will be uploaded.
    \param key: The unique identifier (key) for the object within the S3 bucket.
    \param hashMethod: The hashing algorithm to use when calculating the hash value.
    \param ioStream: An IOStream for the data to be uploaded.
    \param useDefaultHashMethod: A flag indicating whether to use the default hash
method or the one specified in the hashMethod parameter.
    \param[out] hashDataResult: The Hasher object that will store the concatenated
hash value.
    \param[out] partHashes: The vector that will store the calculated hash values
for each part of the file.
    \param client: The S3 client instance used to perform the upload operation.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::doMultipartUpload(const Aws::String &bucket,
                                   const Aws::String &key,
                                   AwsDoc::S3::HASH_METHOD hashMethod,
                                   const std::shared_ptr<Aws::IOStream> &ioStream,
                                   bool useDefaultHashMethod,
                                   AwsDoc::S3::Hasher &hashDataResult,
                                   std::vector<Aws::String> &partHashes,
                                   const Aws::S3::S3Client &client) {

    // Get object size.
    ioStream->seekg(0, ioStream->end);
    size_t objectSize = ioStream->tellg();
    ioStream->seekg(0, ioStream->beg);

    Aws::S3::Model::ChecksumAlgorithm checksumAlgorithm =
    Aws::S3::Model::ChecksumAlgorithm::NOT_SET;
    if (!useDefaultHashMethod) {
        if (hashMethod != MD5) {

```

```

        checksumAlgorithm = getChecksumAlgorithmForHashMethod(hashMethod);
    }
}
Aws::String uploadID = createMultipartUpload(bucket, key, checksumAlgorithm,
client);
if (uploadID.empty()) {
    return false;
}

std::vector<unsigned char> totalHashBuffer;
bool uploadSucceeded = true;
std::streamsize uploadedBytes = 0;
int partNumber = 1;
Aws::Vector<Aws::S3::Model::CompletedPart> parts;
while (uploadedBytes < objectSize) {
    std::cout << "Uploading part " << partNumber << "." << std::endl;

    std::vector<unsigned char> buffer(UPLOAD_BUFFER_SIZE);
    std::streamsize bytesToRead =
static_cast<std::streamsize>(std::min(buffer.size(),

objectSize - uploadedBytes));
    ioStream->read((char *) buffer.data(), bytesToRead);
    Aws::Utils::Stream::PreallocatedStreamBuf
preallocatedStreamBuf(buffer.data(),

bytesToRead);
    std::shared_ptr<Aws::IOStream> body =
        Aws::MakeShared<Aws::IOStream>("SampleAllocationTag",
            &preallocatedStreamBuf);

    Hasher hasher;
    if (!hasher.calculateObjectHash(*body, hashMethod)) {
        std::cerr << "Error calculating hash." << std::endl;
        uploadSucceeded = false;
        break;
    }

    Aws::String base64HashString = hasher.getBase64HashString();
    partHashes.push_back(base64HashString);

    Aws::Utils::ByteBuffer hashBuffer = hasher.getByteBufferHash();

```

```

        totalHashBuffer.insert(totalHashBuffer.end(),
hashBuffer.GetUnderlyingData(),
                                hashBuffer.GetUnderlyingData() +
hashBuffer.GetLength());

        Aws::String calculatedHash;
        if (gUseCalculatedChecksum) {
            calculatedHash = base64HashString;
        }
        Aws::S3::Model::UploadPartOutcome uploadPartOutcome = uploadPart(bucket,
key, uploadID, partNumber,

checksumAlgorithm, base64HashString, body,

                                                                    client);

        if (uploadPartOutcome.IsSuccess()) {
            const Aws::S3::Model::UploadPartResult &uploadPartResult =
uploadPartOutcome.GetResult();
            Aws::S3::Model::CompletedPart completedPart;
            completedPart.SetETag(uploadPartResult.GetETag());
            completedPart.SetPartNumber(partNumber);
            switch (hashMethod) {
                case AwsDoc::S3::MD5:
                    break; // Do nothing.
                case AwsDoc::S3::SHA1:

completedPart.SetChecksumSHA1(uploadPartResult.GetChecksumSHA1());
                    break;
                case AwsDoc::S3::SHA256:

completedPart.SetChecksumSHA256(uploadPartResult.GetChecksumSHA256());
                    break;
                case AwsDoc::S3::CRC32:

completedPart.SetChecksumCRC32(uploadPartResult.GetChecksumCRC32());
                    break;
                case AwsDoc::S3::CRC32C:

completedPart.SetChecksumCRC32C(uploadPartResult.GetChecksumCRC32C());
                    break;
                default:
                    std::cerr << "Unhandled hash method for completedPart." <<
std::endl;
                    break;
            }
        }
    }

```

```

        parts.push_back(completedPart);
    } else {
        std::cerr << "Error uploading part. " <<
            uploadPartOutcome.GetError().GetMessage() << std::endl;
        uploadSucceeded = false;
        break;
    }

    uploadedBytes += bytesToRead;
    partNumber++;
}

if (!uploadSucceeded) {
    abortMultipartUpload(bucket, key, uploadID, client);
    return false;
} else {

    Aws::S3::Model::CompleteMultipartUploadOutcome
completeMultipartUploadOutcome = completeMultipartUpload(bucket,

                                key,

                                uploadID,

                                parts,

                                client);

    if (completeMultipartUploadOutcome.IsSuccess()) {
        std::cout << "Multipart upload completed." << std::endl;
        if (!hashDataResult.calculateObjectHash(totalHashBuffer, hashMethod)) {
            std::cerr << "Error calculating hash." << std::endl;
            return false;
        }
    } else {
        std::cerr << "Error completing multipart upload." <<
            completeMultipartUploadOutcome.GetError().GetMessage()
            << std::endl;
    }

    return completeMultipartUploadOutcome.IsSuccess();
}
}

```

```

///  

///  

///  

    \param: hashMethod: A HASH_METHOD constant.  

    \return: String: A string description of the hash method.  

*/  

Aws::String AwsDoc::S3::stringForHashMethod(AwsDoc::S3::HASH_METHOD hashMethod) {  

    switch (hashMethod) {  

        case AwsDoc::S3::DEFAULT:  

            return "Default";  

        case AwsDoc::S3::MD5:  

            return "MD5";  

        case AwsDoc::S3::SHA1:  

            return "SHA1";  

        case AwsDoc::S3::SHA256:  

            return "SHA256";  

        case AwsDoc::S3::CRC32:  

            return "CRC32";  

        case AwsDoc::S3::CRC32C:  

            return "CRC32C";  

        default:  

            return "Unknown";  

    }  

}  

  

///  

///  

///  

    \param: hashMethod: A HASH_METHOD constant.  

    \return: ChecksumAlgorithm: The ChecksumAlgorithm enum.  

*/  

Aws::S3::Model::ChecksumAlgorithm  

AwsDoc::S3::getChecksumAlgorithmForHashMethod(AwsDoc::S3::HASH_METHOD hashMethod) {  

    Aws::S3::Model::ChecksumAlgorithm result =  

    Aws::S3::Model::ChecksumAlgorithm::NOT_SET;  

    switch (hashMethod) {  

        case AwsDoc::S3::DEFAULT:  

            std::cerr << "getChecksumAlgorithmForHashMethod- DEFAULT is not valid."  

            << std::endl;  

            break; // Default is not supported.  

        case AwsDoc::S3::MD5:  

            break; // Ignore MD5.  

        case AwsDoc::S3::SHA1:  

            result = Aws::S3::Model::ChecksumAlgorithm::SHA1;  

            break;  

    }  

}

```

```

        case AwsDoc::S3::SHA256:
            result = Aws::S3::Model::ChecksumAlgorithm::SHA256;
            break;
        case AwsDoc::S3::CRC32:
            result = Aws::S3::Model::ChecksumAlgorithm::CRC32;
            break;
        case AwsDoc::S3::CRC32C:
            result = Aws::S3::Model::ChecksumAlgorithm::CRC32C;
            break;
        default:
            std::cerr << "Unknown hash method." << std::endl;
            break;
    }

    return result;
}

//! Routine which cleans up after the example is complete.
/*!
    \param bucket: The name of the S3 bucket where the object was uploaded.
    \param clientConfiguration: The client configuration for the S3 client.
    \return bool: Function succeeded.
*/
bool AwsDoc::S3::cleanUp(const Aws::String &bucketName,
                        const Aws::S3::S3ClientConfiguration &clientConfiguration)
{
    Aws::Vector<Aws::String> keysResult;
    bool result = true;
    if (AwsDoc::S3::listObjects(bucketName, keysResult, clientConfiguration)) {
        if (!keysResult.empty()) {
            result = AwsDoc::S3::deleteObjects(keysResult, bucketName,
                                                clientConfiguration);
        }
    } else {
        result = false;
    }

    return result && AwsDoc::S3::deleteBucket(bucketName, clientConfiguration);
}

//! Console interaction introducing the workflow.
/*!

```

```

    \param bucketName: The name of the S3 bucket to use.
*/
void AwsDoc::S3::introductoryExplanations(const Aws::String &bucketName) {

    std::cout
        << "Welcome to the Amazon Simple Storage Service (Amazon S3) object
integrity workflow."
        << std::endl;
    printAsterisksLine();
    std::cout
        << "This workflow demonstrates how Amazon S3 uses checksum values to
verify the integrity of data\n";
    std::cout << "uploaded to Amazon S3 buckets" << std::endl;
    std::cout
        << "The AWS SDK for C++ automatically handles checksums.\n";
    std::cout
        << "By default it calculates a checksum that is uploaded with an object.
\n"
        << "The default checksum algorithm for PutObject and MultiPart upload is
an MD5 hash.\n"
        << "The default checksum algorithm for TransferManager uploads is a
CRC32 checksum."
        << std::endl;
    std::cout
        << "You can override the default behavior, requiring one of the
following checksums,\n";
    std::cout << "MD5, CRC32, CRC32C, SHA-1 or SHA-256." << std::endl;
    std::cout << "You can also set the checksum hash value, instead of letting the
SDK calculate the value."
        << std::endl;
    std::cout
        << "For more information, see https://docs.aws.amazon.com/AmazonS3/
latest/userguide/checking-object-integrity.html."
        << std::endl;

    std::cout
        << "This workflow will locally compute checksums for files uploaded to
an Amazon S3 bucket,\n";
    std::cout << "even when the SDK also computes the checksum." << std::endl;
    std::cout
        << "This is done to provide demonstration code for how the checksums are
calculated."
        << std::endl;
}

```

```

        std::cout << "A bucket named '" << bucketName << "' will be created for the
        object uploads."
                << std::endl;
}

//! Console interaction which explains the PutObject results.
/*!
*/
void AwsDoc::S3::explainPutObjectResults() {

    std::cout << "The upload was successful.\n";
    std::cout << "If the checksums had not matched, the upload would have failed."
                << std::endl;
    std::cout
        << "The checksums calculated by the server have been retrieved using the
        GetObjectAttributes."
        << std::endl;
    std::cout
        << "The locally calculated checksums have been verified against the
        retrieved checksums."
        << std::endl;
}

//! Console interaction explaining transfer manager uploads.
/*!
    \param objectKey: The key for the object being uploaded.
*/
void AwsDoc::S3::introductoryTransferManagerUploadExplanations(
    const Aws::String &objectKey) {
    std::cout
        << "Now the workflow will demonstrate object integrity for
        TransferManager multi-part uploads."
        << std::endl;
    std::cout
        << "The AWS C++ SDK has a TransferManager class which simplifies
        multipart uploads."
        << std::endl;
    std::cout
        << "The following code lets the TransferManager handle much of the
        checksum configuration."
        << std::endl;

    std::cout << "An object with the key '" << objectKey
                << " will be uploaded by the TransferManager using a "

```

```

        << BUFFER_SIZE_IN_MEGABYTES << " MB buffer." << std::endl;
    if (gUseCalculatedChecksum) {
        std::cout << "For TransferManager uploads, this demo always lets the SDK
calculate the hash value."
        << std::endl;
    }

    pressEnterToContinue();
    printAsterisksLine();
}

//! Console interaction explaining multi-part uploads.
/*!
 \param objectKey: The key for the object being uploaded.
 \param chosenHashMethod: The hash method selected by the user.
 */
void AwsDoc::S3::multiPartUploadExplanations(const Aws::String &objectKey,
                                              HASH_METHOD chosenHashMethod) {
    std::cout
        << "Now we will provide an in-depth demonstration of multi-part
uploading by calling the multi-part upload APIs directly."
        << std::endl;
    std::cout << "These are the same APIs used by the TransferManager when uploading
large files."
        << std::endl;
    std::cout
        << "In the following code, the checksums are also calculated locally and
then compared."
        << std::endl;
    std::cout
        << "For multi-part uploads, a checksum is uploaded with each part. The
final checksum is a concatenation of"
        << std::endl;
    std::cout << "the checksums for each part." << std::endl;
    std::cout
        << "This is explained in the user guide, https://docs.aws.amazon.com/
AmazonS3/latest/userguide/checking-object-integrity.html,\"\"
        << " in the section \"Using part-level checksums for multipart uploads
\".\" << std::endl;

    std::cout << "Starting multipart upload of with hash method " <<
        stringForHashMethod(chosenHashMethod) << " uploading to with object
key\n"
        << "\"" << objectKey << "\",\" << std::endl;

```

```
}

//! Create a large file for doing multi-part uploads.
/*!
*/
bool AwsDoc::S3::createLargeFileIfNotExists() {
    // Generate a large file by writing this source file multiple times to a new
    file.
    if (std::filesystem::exists(MULTI_PART_TEST_FILE)) {
        return true;
    }

    std::ofstream newFile(MULTI_PART_TEST_FILE, std::ios::out

                                | std::ios::binary);

    if (!newFile) {
        std::cerr << "createLargeFileIfNotExists- Error creating file " <<
MULTI_PART_TEST_FILE <<
        std::endl;
        return false;
    }

    std::ifstream input(TEST_FILE, std::ios::in

                                | std::ios::binary);

    if (!input) {
        std::cerr << "Error opening file " << TEST_FILE <<
        std::endl;
        return false;
    }
    std::stringstream buffer;
    buffer << input.rdbuf();

    input.close();

    while (newFile.tellp() < LARGE_FILE_SIZE && !newFile.bad()) {
        buffer.seekg(std::stringstream::beg);
        newFile << buffer.rdbuf();
    }

    newFile.close();
}
```

```
    return true;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 다음 주제를 참조하세요.
  - [AbortMultipartUpload](#)
  - [CompleteMultipartUpload](#)
  - [CreateMultipartUpload](#)
  - [DeleteObject](#)
  - [GetObjectAttributes](#)
  - [PutObject](#)
  - [UploadPart](#)

## SDK for C++를 사용한 Secrets Manager 예제

다음 코드 예제에서는 Secrets Manager와 AWS SDK for C++ 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

#### GetSecretValue

다음 코드 예시는 GetSecretValue의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예제 리포지토리](#)에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Retrieve an AWS Secrets Manager encrypted secret.
/*!
    \param secretID: The ID for the secret.
    \return bool: Function succeeded.
*/
bool AwsDoc::SecretsManager::getSecretValue(const Aws::String &secretID,
                                             const Aws::Client::ClientConfiguration
                                             &clientConfiguration) {
    Aws::SecretsManager::SecretsManagerClient
    secretsManagerClient(clientConfiguration);

    Aws::SecretsManager::Model::GetSecretValueRequest request;
    request.SetSecretId(secretID);

    Aws::SecretsManager::Model::GetSecretValueOutcome getSecretValueOutcome =
    secretsManagerClient.GetSecretValue(
        request);
    if (getSecretValueOutcome.IsSuccess()) {
        std::cout << "Secret is: "
                    << getSecretValueOutcome.GetResult().GetSecretString() <<
std::endl;
    }
    else {
        std::cerr << "Failed with Error: " << getSecretValueOutcome.GetError()
                    << std::endl;
    }

    return getSecretValueOutcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetSecretValue](#)를 참조하세요.

## SDK for C++를 사용한 Amazon SES 예제

다음 코드 예제에서는 AWS SDK for C++ Amazon SES에서 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](#)
- [시나리오](#)

### 작업

#### CreateReceiptFilter

다음 코드 예시는 CreateReceiptFilter의 사용 방법을 보여줍니다.

SDK for C++

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
//! Create an Amazon Simple Email Service (Amazon SES) receipt filter..  
/*!  
    \param receiptFilterName: The name for the receipt filter.  
    \param cidr: IP address or IP address range in Classless Inter-Domain Routing  
(CIDR) notation.  
    \param policy: Block or allow enum of type ReceiptFilterPolicy.  
    \param clientConfiguration: AWS client configuration.  
    \return bool: Function succeeded.
```

```

*/
bool AwsDoc::SES::createReceiptFilter(const Aws::String &receiptFilterName,
                                      const Aws::String &cidr,
                                      Aws::SES::Model::ReceiptFilterPolicy policy,
                                      const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);
    Aws::SES::Model::CreateReceiptFilterRequest createReceiptFilterRequest;
    Aws::SES::Model::ReceiptFilter receiptFilter;
    Aws::SES::Model::ReceiptIpFilter receiptIpFilter;
    receiptIpFilter.SetCidr(cidr);
    receiptIpFilter.SetPolicy(policy);
    receiptFilter.SetName(receiptFilterName);
    receiptFilter.SetIpFilter(receiptIpFilter);
    createReceiptFilterRequest.SetFilter(receiptFilter);
    Aws::SES::Model::CreateReceiptFilterOutcome createReceiptFilterOutcome =
sesClient.CreateReceiptFilter(
    createReceiptFilterRequest);
    if (createReceiptFilterOutcome.IsSuccess()) {
        std::cout << "Successfully created receipt filter." << std::endl;
    }
    else {
        std::cerr << "Error creating receipt filter: " <<
            createReceiptFilterOutcome.GetError().GetMessage() << std::endl;
    }

    return createReceiptFilterOutcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CreateReceiptFilter](#)를 참조하세요.

## CreateReceiptRule

다음 코드 예시는 CreateReceiptRule의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Create an Amazon Simple Email Service (Amazon SES) receipt rule.
/*!
    \param receiptRuleName: The name for the receipt rule.
    \param s3BucketName: The name of the S3 bucket for incoming mail.
    \param s3ObjectKeyPrefix: The prefix for the objects in the S3 bucket.
    \param ruleSetName: The name of the rule set where the receipt rule is added.
    \param recipients: Aws::Vector of recipients.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::SES::createReceiptRule(const Aws::String &receiptRuleName,
                                     const Aws::String &s3BucketName,
                                     const Aws::String &s3ObjectKeyPrefix,
                                     const Aws::String &ruleSetName,
                                     const Aws::Vector<Aws::String> &recipients,
                                     const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::CreateReceiptRuleRequest createReceiptRuleRequest;

    Aws::SES::Model::S3Action s3Action;
    s3Action.SetBucketName(s3BucketName);
    s3Action.SetObjectKeyPrefix(s3ObjectKeyPrefix);

    Aws::SES::Model::ReceiptAction receiptAction;
    receiptAction.SetS3Action(s3Action);

    Aws::SES::Model::ReceiptRule receiptRule;
    receiptRule.SetName(receiptRuleName);
    receiptRule.WithRecipients(recipients);

    Aws::Vector<Aws::SES::Model::ReceiptAction> receiptActionList;
    receiptActionList.emplace_back(receiptAction);
    receiptRule.SetActions(receiptActionList);

    createReceiptRuleRequest.SetRuleSetName(ruleSetName);
    createReceiptRuleRequest.SetRule(receiptRule);

    auto outcome = sesClient.CreateReceiptRule(createReceiptRuleRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully created receipt rule." << std::endl;
    }
}

```

```

    }
    else {
        std::cerr << "Error creating receipt rule. " <<
outcome.GetError().GetMessage()
        << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CreateReceiptRule](#)을 참조하세요.

## CreateReceiptRuleSet

다음 코드 예시는 CreateReceiptRuleSet의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Create an Amazon Simple Email Service (Amazon SES) receipt rule set.
/*!
    \param ruleSetName: The name of the rule set.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */
bool AwsDoc::SES::createReceiptRuleSet(const Aws::String &ruleSetName,
                                       const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::CreateReceiptRuleSetRequest createReceiptRuleSetRequest;

    createReceiptRuleSetRequest.SetRuleSetName(ruleSetName);

```

```

    Aws::SES::Model::CreateReceiptRuleSetOutcome outcome =
    sesClient.CreateReceiptRuleSet(
        createReceiptRuleSetRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully created receipt rule set." << std::endl;
    }
    else {
        std::cerr << "Error creating receipt rule set. "
            << outcome.GetError().GetMessage()
            << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CreateReceiptRuleSet](#)를 참조하세요.

## CreateTemplate

다음 코드 예시는 CreateTemplate의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Create an Amazon Simple Email Service (Amazon SES) template.
/*!
    \param templateName: The name of the template.
    \param htmlPart: The HTML body of the email.
    \param subjectPart: The subject line of the email.
    \param textPart: The plain text version of the email.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::SES::createTemplate(const Aws::String &templateName,

```

```

        const Aws::String &htmlPart,
        const Aws::String &subjectPart,
        const Aws::String &textPart,
        const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::CreateTemplateRequest createTemplateRequest;
    Aws::SES::Model::Template aTemplate;

    aTemplate.SetTemplateName(templateName);
    aTemplate.SetHtmlPart(htmlPart);
    aTemplate.SetSubjectPart(subjectPart);
    aTemplate.SetTextPart(textPart);

    createTemplateRequest.SetTemplate(aTemplate);

    Aws::SES::Model::CreateTemplateOutcome outcome = sesClient.CreateTemplate(
        createTemplateRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully created template." << templateName << "."
            << std::endl;
    }
    else {
        std::cerr << "Error creating template. " << outcome.GetError().GetMessage()
            << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CreateTemplate](#)을 참조하세요.

## DeleteIdentity

다음 코드 예시는 DeleteIdentity의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Delete the specified identity (an email address or a domain).
/*!
  \param identity: The identity to delete.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::SES::deleteIdentity(const Aws::String &identity,
                                const Aws::Client::ClientConfiguration
                                &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteIdentityRequest deleteIdentityRequest;

    deleteIdentityRequest.SetIdentity(identity);

    Aws::SES::Model::DeleteIdentityOutcome outcome = sesClient.DeleteIdentity(
        deleteIdentityRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted identity." << std::endl;
    }
    else {
        std::cerr << "Error deleting identity. " << outcome.GetError().GetMessage()
            << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteIdentity](#)를 참조하세요.

## DeleteReceiptFilter

다음 코드 예시는 DeleteReceiptFilter의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
///  
// Delete an Amazon Simple Email Service (Amazon SES) receipt filter.  
/*!  
  \param receiptFilterName: The name for the receipt filter.  
  \param clientConfiguration: AWS client configuration.  
  \return bool: Function succeeded.  
*/  
bool AwsDoc::SES::deleteReceiptFilter(const Aws::String &receiptFilterName,  
                                       const Aws::Client::ClientConfiguration  
                                       &clientConfiguration) {  
    Aws::SES::SESClient sesClient(clientConfiguration);  
  
    Aws::SES::Model::DeleteReceiptFilterRequest deleteReceiptFilterRequest;  
  
    deleteReceiptFilterRequest.SetFilterName(receiptFilterName);  
  
    Aws::SES::Model::DeleteReceiptFilterOutcome outcome =  
    sesClient.DeleteReceiptFilter(  
        deleteReceiptFilterRequest);  
  
    if (outcome.IsSuccess()) {  
        std::cout << "Successfully deleted receipt filter." << std::endl;  
    }  
    else {  
        std::cerr << "Error deleting receipt filter. "  
                   << outcome.GetError().GetMessage()  
                   << std::endl;  
    }  
  
    return outcome.IsSuccess();  
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteReceiptFilter](#)를 참조하세요.

## DeleteReceiptRule

다음 코드 예시는 DeleteReceiptRule의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Delete an Amazon Simple Email Service (Amazon SES) receipt rule.
/*!
    \param receiptRuleName: The name for the receipt rule.
    \param receiptRuleSetName: The name for the receipt rule set.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::SES::deleteReceiptRule(const Aws::String &receiptRuleName,
                                     const Aws::String &receiptRuleSetName,
                                     const Aws::Client::ClientConfiguration
                                     &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteReceiptRuleRequest deleteReceiptRuleRequest;

    deleteReceiptRuleRequest.SetRuleName(receiptRuleName);
    deleteReceiptRuleRequest.SetRuleSetName(receiptRuleSetName);

    Aws::SES::Model::DeleteReceiptRuleOutcome outcome = sesClient.DeleteReceiptRule(
        deleteReceiptRuleRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted receipt rule." << std::endl;
    }
    else {

```

```

        std::cout << "Error deleting receipt rule. " <<
outcome.GetError().GetMessage()
        << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteReceiptRule](#)을 참조하세요.

## DeleteReceiptRuleSet

다음 코드 예시는 DeleteReceiptRuleSet의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Delete an Amazon Simple Email Service (Amazon SES) receipt rule set.
/*!
    \param receiptRuleSetName: The name for the receipt rule set.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */
bool AwsDoc::SES::deleteReceiptRuleSet(const Aws::String &receiptRuleSetName,
                                       const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteReceiptRuleSetRequest deleteReceiptRuleSetRequest;

    deleteReceiptRuleSetRequest.SetRuleSetName(receiptRuleSetName);

    Aws::SES::Model::DeleteReceiptRuleSetOutcome outcome =
    sesClient.DeleteReceiptRuleSet(

```

```

        deleteReceiptRuleSetRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted receipt rule set." << std::endl;
    }

    else {
        std::cerr << "Error deleting receipt rule set. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteReceiptRuleSet](#)를 참조하세요.

## DeleteTemplate

다음 코드 예시는 DeleteTemplate의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Delete an Amazon Simple Email Service (Amazon SES) template.
/*!
    \param templateName: The name for the template.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::SES::deleteTemplate(const Aws::String &templateName,
                                const Aws::Client::ClientConfiguration
                                &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::DeleteTemplateRequest deleteTemplateRequest;

```

```

deleteTemplateRequest.SetTemplateName(templateName);

Aws::SES::Model::DeleteTemplateOutcome outcome = sesClient.DeleteTemplate(
    deleteTemplateRequest);

if (outcome.IsSuccess()) {
    std::cout << "Successfully deleted template." << std::endl;
}
else {
    std::cerr << "Error deleting template. " << outcome.GetError().GetMessage()
        << std::endl;
}

return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteTemplate](#)을 참조하세요.

## GetTemplate

다음 코드 예시는 GetTemplate의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Get a template's attributes.
/*!
    \param templateName: The name for the template.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
 */
bool AwsDoc::SES::getTemplate(const Aws::String &templateName,
                             const Aws::Client::ClientConfiguration
    &clientConfiguration) {

```

```

    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::GetTemplateRequest getTemplateRequest;

    getTemplateRequest.SetTemplateName(templateName);

    Aws::SES::Model::GetTemplateOutcome outcome = sesClient.GetTemplate(
        getTemplateRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully got template." << std::endl;
    }

    else {
        std::cerr << "Error getting template. " << outcome.GetError().GetMessage()
            << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetTemplate](#)을 참조하세요.

## ListIdentities

다음 코드 예시는 ListIdentities의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

///! List the identities associated with this account.
/*!
    \param identityType: The identity type enum. "NOT_SET" is a valid option.
    \param identities; A vector to receive the retrieved identities.
    \param clientConfiguration: AWS client configuration.

```

```

    \return bool: Function succeeded.
    */
bool AwsDoc::SES::listIdentities(Aws::SES::Model::IdentityType identityType,
                                Aws::Vector<Aws::String> &identities,
                                const Aws::Client::ClientConfiguration
                                &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::ListIdentitiesRequest listIdentitiesRequest;

    if (identityType != Aws::SES::Model::IdentityType::NOT_SET) {
        listIdentitiesRequest.SetIdentityType(identityType);
    }

    Aws::String nextToken; // Used for paginated results.
    do {
        if (!nextToken.empty()) {
            listIdentitiesRequest.SetNextToken(nextToken);
        }
        Aws::SES::Model::ListIdentitiesOutcome outcome = sesClient.ListIdentities(
            listIdentitiesRequest);

        if (outcome.IsSuccess()) {
            const auto &retrievedIdentities = outcome.GetResult().GetIdentities();
            if (!retrievedIdentities.empty()) {
                identities.insert(identities.cend(), retrievedIdentities.cbegin(),
                                retrievedIdentities.cend());
            }
            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cout << "Error listing identities. " <<
outcome.GetError().GetMessage()
                << std::endl;
            return false;
        }
    } while (!nextToken.empty());

    return true;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ListIdentities](#)를 참조하세요.

## ListReceiptFilters

다음 코드 예시는 ListReceiptFilters의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! List the receipt filters associated with this account.
/*!
    \param filters; A vector of "ReceiptFilter" to receive the retrieved filters.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
 */
bool
AwsDoc::SES::listReceiptFilters(Aws::Vector<Aws::SES::Model::ReceiptFilter>
    &filters,
                                const Aws::Client::ClientConfiguration
    &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);
    Aws::SES::Model::ListReceiptFiltersRequest listReceiptFiltersRequest;

    Aws::SES::Model::ListReceiptFiltersOutcome outcome =
    sesClient.ListReceiptFilters(
        listReceiptFiltersRequest);
    if (outcome.IsSuccess()) {
        auto &retrievedFilters = outcome.GetResult().GetFilters();
        if (!retrievedFilters.empty()) {
            filters.insert(filters.cend(), retrievedFilters.cbegin(),
                           retrievedFilters.cend());
        }
    }
    else {
        std::cerr << "Error retrieving IP address filters: "
                   << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ListReceiptFilters](#)를 참조하세요.

## SendEmail

다음 코드 예시는 SendEmail의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Send an email to a list of recipients.
/*!
    \param recipients: Vector of recipient email addresses.
    \param subject: Email subject.
    \param htmlBody: Email body as HTML. At least one body data is required.
    \param textBody: Email body as plain text. At least one body data is required.
    \param senderEmailAddress: Email address of sender. Ignored if empty string.
    \param ccAddresses: Vector of cc addresses. Ignored if empty.
    \param replyToAddress: Reply to email address. Ignored if empty string.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::SES::sendEmail(const Aws::Vector<Aws::String> &recipients,
                           const Aws::String &subject,
                           const Aws::String &htmlBody,
                           const Aws::String &textBody,
                           const Aws::String &senderEmailAddress,
                           const Aws::Vector<Aws::String> &ccAddresses,
                           const Aws::String &replyToAddress,
                           const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::Destination destination;
    if (!ccAddresses.empty()) {
        destination.WithCcAddresses(ccAddresses);
    }
}

```

```
}
if (!recipients.empty()) {
    destination.WithToAddresses(recipients);
}

Aws::SES::Model::Body message_body;
if (!htmlBody.empty()) {
    message_body.SetHtml(
        Aws::SES::Model::Content().WithCharset("UTF-8").WithData(htmlBody));
}

if (!textBody.empty()) {
    message_body.SetText(
        Aws::SES::Model::Content().WithCharset("UTF-8").WithData(textBody));
}

Aws::SES::Model::Message message;
message.SetBody(message_body);
message.SetSubject(
    Aws::SES::Model::Content().WithCharset("UTF-8").WithData(subject));

Aws::SES::Model::SendEmailRequest sendEmailRequest;
sendEmailRequest.SetDestination(destination);
sendEmailRequest.SetMessage(message);
if (!senderEmailAddress.empty()) {
    sendEmailRequest.SetSource(senderEmailAddress);
}
if (!replyToAddress.empty()) {
    sendEmailRequest.AddReplyToAddresses(replyToAddress);
}

auto outcome = sesClient.SendEmail(sendEmailRequest);

if (outcome.IsSuccess()) {
    std::cout << "Successfully sent message with ID "
                << outcome.GetResult().GetMessageId()
                << "." << std::endl;
}
else {
    std::cerr << "Error sending message. " << outcome.GetError().GetMessage()
                << std::endl;
}

return outcome.IsSuccess();
```

```
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [SendEmail](#)을 참조하세요.

## SendTemplatedEmail

다음 코드 예시는 SendTemplatedEmail의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
///! Send a templated email to a list of recipients.
/*!
  \param recipients; Vector of recipient email addresses.
  \param templateName: The name of the template to use.
  \param templateData: Map of key-value pairs for replacing text in template.
  \param senderEmailAddress: Email address of sender. Ignored if empty string.
  \param ccAddresses: Vector of cc addresses. Ignored if empty.
  \param replyToAddress: Reply to email address. Ignored if empty string.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
*/
bool AwsDoc::SES::sendTemplatedEmail(const Aws::Vector<Aws::String> &recipients,
                                     const Aws::String &templateName,
                                     const Aws::Map<Aws::String, Aws::String>
                                     &templateData,
                                     const Aws::String &senderEmailAddress,
                                     const Aws::Vector<Aws::String> &ccAddresses,
                                     const Aws::String &replyToAddress,
                                     const Aws::Client::ClientConfiguration
                                     &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::Destination destination;
    if (!ccAddresses.empty()) {
        destination.WithCcAddresses(ccAddresses);
    }
}
```

```

    }
    if (!recipients.empty()) {
        destination.WithToAddresses(recipients);
    }

    Aws::SES::Model::SendTemplatedEmailRequest sendTemplatedEmailRequest;
    sendTemplatedEmailRequest.SetDestination(destination);
    sendTemplatedEmailRequest.SetTemplate(templateName);

    std::ostringstream templateDataStream;
    templateDataStream << "{";
    size_t dataCount = 0;
    for (auto &pair: templateData) {
        templateDataStream << "\"" << pair.first << "":\"" << pair.second << "\"";
        dataCount++;
        if (dataCount < templateData.size()) {
            templateDataStream << ",";
        }
    }
    templateDataStream << "}";

    sendTemplatedEmailRequest.SetTemplateData(templateDataStream.str());

    if (!senderEmailAddress.empty()) {
        sendTemplatedEmailRequest.SetSource(senderEmailAddress);
    }
    if (!replyToAddress.empty()) {
        sendTemplatedEmailRequest.AddReplyToAddresses(replyToAddress);
    }

    auto outcome = sesClient.SendTemplatedEmail(sendTemplatedEmailRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully sent templated message with ID "
                  << outcome.GetResult().GetMessageId()
                  << "." << std::endl;
    }
    else {
        std::cerr << "Error sending templated message. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();

```

```
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [SendTemplatedEmail](#)을 참조하세요.

## UpdateTemplate

다음 코드 예시는 UpdateTemplate의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
//! Update an Amazon Simple Email Service (Amazon SES) template.
/*!
    \param templateName: The name of the template.
    \param htmlPart: The HTML body of the email.
    \param subjectPart: The subject line of the email.
    \param textPart: The plain text version of the email.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::SES::updateTemplate(const Aws::String &templateName,
                                const Aws::String &htmlPart,
                                const Aws::String &subjectPart,
                                const Aws::String &textPart,
                                const Aws::Client::ClientConfiguration
                                &clientConfiguration) {
    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::Template templateValues;

    templateValues.SetTemplateName(templateName);
    templateValues.SetSubjectPart(subjectPart);
    templateValues.SetHtmlPart(htmlPart);
    templateValues.SetTextPart(textPart);
```

```

    Aws::SES::Model::UpdateTemplateRequest updateTemplateRequest;
    updateTemplateRequest.SetTemplate(templateValues);

    Aws::SES::Model::UpdateTemplateOutcome outcome =
    sesClient.UpdateTemplate(updateTemplateRequest);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated template." << std::endl;
    } else {
        std::cerr << "Error updating template. " << outcome.GetError().GetMessage()
            << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [UpdateTemplate](#)을 참조하세요.

## VerifyEmailIdentity

다음 코드 예시는 VerifyEmailIdentity의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Add an email address to the list of identities associated with this account and
//! initiate verification.
/*!
    \param emailAddress; The email address to add.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::SES::verifyEmailIdentity(const Aws::String &emailAddress,
                                     const Aws::Client::ClientConfiguration
    &clientConfiguration)
{

```

```

    Aws::SES::SESClient sesClient(clientConfiguration);

    Aws::SES::Model::VerifyEmailIdentityRequest verifyEmailIdentityRequest;

    verifyEmailIdentityRequest.SetEmailAddress(emailAddress);

    Aws::SES::Model::VerifyEmailIdentityOutcome outcome =
    sesClient.VerifyEmailIdentity(verifyEmailIdentityRequest);

    if (outcome.IsSuccess())
    {
        std::cout << "Email verification initiated." << std::endl;
    }

    else
    {
        std::cerr << "Error initiating email verification. " <<
        outcome.GetError().GetMessage()
            << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ SDK API 참조의 [VerifyEmailIdentity](#)를 참조하세요.

## 시나리오

### Aurora 서버리스 작업 항목 트래커 만들기

다음 코드 예제에서는 Amazon Aurora Serverless 데이터베이스에서 작업 항목을 추적하고 Amazon Simple Email Service(Amazon SES)를 사용하여 보고서를 보내는 웹 애플리케이션을 만드는 방법을 보여줍니다.

#### SDK for C++

Amazon Aurora Serverless 데이터베이스에 저장된 작업 항목을 추적하고 보고하는 웹 애플리케이션을 만드는 방법을 보여줍니다.

Amazon Aurora Serverless 데이터를 쿼리하고 React 애플리케이션에서 사용하도록 C++ REST API를 설정하는 방법에 대한 지침과 전체 소스 코드는 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- Aurora
- Amazon RDS
- Amazon RDS 데이터 서비스
- Amazon SES

## SDK for C++를 사용한 Amazon SNS 예제

다음 코드 예제에서는 AWS SDK for C++ Amazon SNS에서 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

- [시작하기](#)
- [작업](#)
- [시나리오](#)

## 시작하기

Hello Amazon SNS

다음 코드 예제에서는 Amazon SNS 사용을 시작하는 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

## CMakeLists.txt CMake 파일의 코드입니다.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS sns)

# Set this project's name.
project("hello_sns")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
  for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
      "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
  endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
  # Copy relevant AWS SDK for C++ libraries into the current binary directory for
  running and debugging.

  # set(BIN_SUB_DIR "/Debug") # If you are building from the command line you may
  need to uncomment this
  # and set the proper subdirectory to the executables' location.

  AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
    ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
  hello_sns.cpp)

target_link_libraries(${PROJECT_NAME}
```

```
${AWSSDK_LINK_LIBRARIES})
```

hello\_sns.cpp 소스 파일의 코드입니다.

```
#include <aws/core/Aws.h>
#include <aws/sns/SNSClient.h>
#include <aws/sns/model/ListTopicsRequest.h>
#include <iostream>

/*
 * A "Hello SNS" starter application which initializes an Amazon Simple
 * Notification
 * Service (Amazon SNS) client and lists the SNS topics in the current account.
 *
 * main function
 *
 * Usage: 'hello_sns'
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::SNS::SNSClient snsClient(clientConfig);

        Aws::Vector<Aws::SNS::Model::Topic> allTopics;
        Aws::String nextToken; // Next token is used to handle a paginated response.
        do {
            Aws::SNS::Model::ListTopicsRequest request;

            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }

            const Aws::SNS::Model::ListTopicsOutcome outcome = snsClient.ListTopics(
```

```

        request);

    if (outcome.IsSuccess()) {
        const Aws::Vector<Aws::SNS::Model::Topic> &paginatedTopics =
            outcome.GetResult().GetTopics();
        if (!paginatedTopics.empty()) {
            allTopics.insert(allTopics.cend(), paginatedTopics.cbegin(),
                             paginatedTopics.cend());
        }
    }
    else {
        std::cerr << "Error listing topics " <<
outcome.GetError().GetMessage()
        << std::endl;
        return 1;
    }

    nextToken = outcome.GetResult().GetNextToken();
} while (!nextToken.empty());

std::cout << "Hello Amazon SNS! You have " << allTopics.size() << " topic"
        << (allTopics.size() == 1 ? "" : "s") << " in your account."
        << std::endl;

if (!allTopics.empty()) {
    std::cout << "Here are your topic ARNs." << std::endl;
    for (const Aws::SNS::Model::Topic &topic: allTopics) {
        std::cout << "  * " << topic.GetTopicArn() << std::endl;
    }
}

}

Aws::ShutdownAPI(options); // Should only be called once.
return 0;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ListTopics](#)를 참조하세요.

## 작업

### CreateTopic

다음 코드 예시는 CreateTopic의 사용 방법을 보여줍니다.

SDK for C++

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Create an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
    \param topicName: An Amazon SNS topic name.
    \param topicARNResult: String to return the Amazon Resource Name (ARN) for the
    topic.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::SNS::createTopic(const Aws::String &topicName,
                              Aws::String &topicARNResult,
                              const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::CreateTopicRequest request;
    request.SetName(topicName);

    const Aws::SNS::Model::CreateTopicOutcome outcome =
    snsClient.CreateTopic(request);

    if (outcome.IsSuccess()) {
        topicARNResult = outcome.GetResult().GetTopicArn();
        std::cout << "Successfully created an Amazon SNS topic " << topicName
                    << " with topic ARN '" << topicARNResult
                    << "'." << std::endl;
    }
    else {

```

```

        std::cerr << "Error creating topic " << topicName << ":" <<
            outcome.GetError().GetMessage() << std::endl;
        topicARNResult.clear();
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CreateTopic](#)을 참조하세요.

## DeleteTopic

다음 코드 예시는 DeleteTopic의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Delete an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
    \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::SNS::deleteTopic(const Aws::String &topicARN,
                              const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::DeleteTopicRequest request;
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::DeleteTopicOutcome outcome =
snsClient.DeleteTopic(request);

    if (outcome.IsSuccess()) {

```

```

        std::cout << "Successfully deleted the Amazon SNS topic " << topicARN <<
std::endl;
    }
    else {
        std::cerr << "Error deleting topic " << topicARN << ":" <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteTopic](#)을 참조하세요.

## GetSMSAttributes

다음 코드 예시는 GetSMSAttributes의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Retrieve the default settings for sending SMS messages from your AWS account by
using
//! Amazon Simple Notification Service (Amazon SNS).
//!
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool
AwsDoc::SNS::getSMSType(const Aws::Client::ClientConfiguration &clientConfiguration)
{
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::GetSMSAttributesRequest request;
    //Set the request to only retrieve the DefaultSMSType setting.
    //Without the following line, GetSMSAttributes would retrieve all settings.
    request.AddAttributes("DefaultSMSType");
}

```

```

    const Aws::SNS::Model::GetSMSAttributesOutcome outcome =
snsClient.GetSMSAttributes(
    request);

    if (outcome.IsSuccess()) {
        const Aws::Map<Aws::String, Aws::String> attributes =
            outcome.GetResult().GetAttributes();
        if (!attributes.empty()) {
            for (auto const &att: attributes) {
                std::cout << att.first << ": " << att.second << std::endl;
            }
        }
        else {
            std::cout
                << "AwsDoc::SNS::getSMSType - an empty map of attributes was
retrieved."
                << std::endl;
        }
    }
    else {
        std::cerr << "Error while getting SMS Type: '"
            << outcome.GetError().GetMessage()
            << "'" << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetSMSAttributes](#)를 참조하세요.

## GetTopicAttributes

다음 코드 예시는 GetTopicAttributes의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Retrieve the properties of an Amazon Simple Notification Service (Amazon SNS)
topic.
/*!
\param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::SNS::getTopicAttributes(const Aws::String &topicARN,
                                     const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);
    Aws::SNS::Model::GetTopicAttributesRequest request;
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::GetTopicAttributesOutcome outcome =
snsClient.GetTopicAttributes(
    request);

    if (outcome.IsSuccess()) {
        std::cout << "Topic Attributes:" << std::endl;
        for (auto const &attribute: outcome.GetResult().GetAttributes()) {
            std::cout << "  * " << attribute.first << " : " << attribute.second
                << std::endl;
        }
    }
    else {
        std::cerr << "Error while getting Topic attributes "
            << outcome.GetError().GetMessage()
            << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetTopicAttributes](#)를 참조하세요.

## ListSubscriptions

다음 코드 예시는 ListSubscriptions의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Retrieve a list of Amazon Simple Notification Service (Amazon SNS)
subscriptions.
/*!
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::SNS::listSubscriptions(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::String nextToken; // Next token is used to handle a paginated response.
    bool result = true;
    Aws::Vector<Aws::SNS::Model::Subscription> subscriptions;
    do {
        Aws::SNS::Model::ListSubscriptionsRequest request;

        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        const Aws::SNS::Model::ListSubscriptionsOutcome outcome =
snsClient.ListSubscriptions(
    request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::SNS::Model::Subscription> &newSubscriptions =
                outcome.GetResult().GetSubscriptions();
            subscriptions.insert(subscriptions.cend(), newSubscriptions.begin(),
                                newSubscriptions.end());
        }
        else {
            std::cerr << "Error listing subscriptions "
                << outcome.GetError().GetMessage()
                <<

```

```

        std::endl;
        result = false;
        break;
    }

    nextToken = outcome.GetResult().GetNextToken();
} while (!nextToken.empty());

if (result) {
    if (subscriptions.empty()) {
        std::cout << "No subscriptions found" << std::endl;
    }
    else {
        std::cout << "Subscriptions list:" << std::endl;
        for (auto const &subscription: subscriptions) {
            std::cout << "  * " << subscription.GetSubscriptionArn() <<
std::endl;
        }
    }
}
return result;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ListSubscriptions](#)를 참조하세요.

## ListTopics

다음 코드 예시는 ListTopics의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Retrieve a list of Amazon Simple Notification Service (Amazon SNS) topics.
/*!
    \param clientConfiguration: AWS client configuration.

```

```

    \return bool: Function succeeded.
    */
bool
AwsDoc::SNS::listTopics(const Aws::Client::ClientConfiguration &clientConfiguration)
{
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::String nextToken; // Next token is used to handle a paginated response.
    bool result = true;
    do {
        Aws::SNS::Model::ListTopicsRequest request;

        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        const Aws::SNS::Model::ListTopicsOutcome outcome = snsClient.ListTopics(
            request);

        if (outcome.IsSuccess()) {
            std::cout << "Topics list:" << std::endl;
            for (auto const &topic: outcome.GetResult().GetTopics()) {
                std::cout << "  * " << topic.GetTopicArn() << std::endl;
            }
        }
        else {
            std::cerr << "Error listing topics " << outcome.GetError().GetMessage()
<<
                std::endl;
            result = false;
            break;
        }

        nextToken = outcome.GetResult().GetNextToken();
    } while (!nextToken.empty());

    return result;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ListTopics](#)를 참조하세요.

## Publish

다음 코드 예시는 Publish의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Send a message to an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
    \param message: The message to publish.
    \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::SNS::publishToTopic(const Aws::String &message,
                                const Aws::String &topicARN,
                                const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

    if (outcome.IsSuccess()) {
        std::cout << "Message published successfully with id '"
                    << outcome.GetResult().GetMessageId() << "'." << std::endl;
    }
    else {
        std::cerr << "Error while publishing message "
                    << outcome.GetError().GetMessage()
                    << std::endl;
    }

    return outcome.IsSuccess();
}

```

## 속성을 사용하여 메시지 게시

```

static const Aws::String TONE_ATTRIBUTE("tone");
static const Aws::Vector<Aws::String> TONES = {"cheerful", "funny",
"serious",
                                     "sincere"};

Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::SNS::SNSClient snsClient(clientConfiguration);

Aws::SNS::Model::PublishRequest request;
request.SetTopicArn(topicARN);
Aws::String message = askQuestion("Enter a message text to publish. ");
request.SetMessage(message);

if (filteringMessages && askYesNoQuestion(
    "Add an attribute to this message? (y/n) ")) {
    for (size_t i = 0; i < TONES.size(); ++i) {
        std::cout << "    " << (i + 1) << ". " << TONES[i] << std::endl;
    }
    int selection = askQuestionForIntRange(
        "Enter a number for an attribute. ",
        1, static_cast<int>(TONES.size()));
    Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
    messageAttributeValue.SetDataType("String");
    messageAttributeValue.SetStringValue(TONES[selection - 1]);
    request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
}

Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

if (outcome.IsSuccess()) {
    std::cout << "Your message was successfully published." << std::endl;
}
else {
    std::cerr << "Error with TopicsAndQueues::Publish. "
               << outcome.GetError().GetMessage()
               << std::endl;
}

```

```

        cleanUp(topicARN,
                queueURLS,
                subscriptionARNS,
                snsClient,
                sqsClient);

    return false;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [Publish](#)를 참조하세요.

## SetSMSAttributes

다음 코드 예시는 SetSMSAttributes의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon SNS를 사용하여 DefaultSMSType 속성을 설정하는 방법입니다.

```

//! Set the default settings for sending SMS messages.
/*!
    \param smsType: The type of SMS message that you will send by default.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::SNS::setSMSType(const Aws::String &smsType,
                             const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SetSMSAttributesRequest request;
    request.AddAttributes("DefaultSMSType", smsType);
}

```

```

    const Aws::SNS::Model::SetSMSAttributesOutcome outcome =
snsClient.SetSMSAttributes(
    request);

    if (outcome.IsSuccess()) {
        std::cout << "SMS Type set successfully " << std::endl;
    }
    else {
        std::cerr << "Error while setting SMS Type: '"
            << outcome.GetError().GetMessage()
            << "'" << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [SetSMSAttributes](#)를 참조하세요.

## Subscribe

다음 코드 예시는 Subscribe의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

이메일 주소로 주제 구독.

```

//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
delivery to an email address.
/*!
    \param topicARN: An SNS topic Amazon Resource Name (ARN).
    \param emailAddress: An email address.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::SNS::subscribeEmail(const Aws::String &topicARN,

```

```

        const Aws::String &emailAddress,
        const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("email");
    request.SetEndpoint(emailAddress);

    const Aws::SNS::Model::SubscribeOutcome outcome = snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
outcome.GetResult().GetSubscriptionArn()
        << "'" << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " << outcome.GetError().GetMessage()
        << std::endl;
    }

    return outcome.IsSuccess();
}

```

모바일 애플리케이션으로 주제 구독.

```

//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
delivery to a mobile app.
/*!
    \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
    \param endpointARN: The ARN for a mobile app or device endpoint.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool
AwsDoc::SNS::subscribeApp(const Aws::String &topicARN,
                        const Aws::String &endpointARN,
                        const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

```

```

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("application");
    request.SetEndpoint(endpointARN);

    const Aws::SNS::Model::SubscribeOutcome outcome = snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
outcome.GetResult().GetSubscriptionArn()
        << "'" << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " << outcome.GetError().GetMessage()
        << std::endl;
    }

    return outcome.IsSuccess();
}

```

Lambda 함수에서 주제를 구독합니다.

```

//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
delivery to an AWS Lambda function.
/*!
    \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
    \param lambdaFunctionARN: The ARN for an AWS Lambda function.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
*/
bool AwsDoc::SNS::subscribeLambda(const Aws::String &topicARN,
                                   const Aws::String &lambdaFunctionARN,
                                   const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("lambda");

```

```

request.SetEndpoint(lambdaFunctionARN);

const Aws::SNS::Model::SubscribeOutcome outcome = snsClient.Subscribe(request);

if (outcome.IsSuccess()) {
    std::cout << "Subscribed successfully." << std::endl;
    std::cout << "Subscription ARN '" <<
outcome.GetResult().GetSubscriptionArn()
    << "'" << std::endl;
}
else {
    std::cerr << "Error while subscribing " << outcome.GetError().GetMessage()
    << std::endl;
}

return outcome.IsSuccess();
}

```

SQS 대기열로 주제 구독.

```

Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("sqs");
    request.SetEndpoint(queueARN);

    Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
        std::cout << "The queue '" << queueName
    << "' has been subscribed to the topic '"
    << "'" << topicName << "'" << std::endl;
        std::cout << "with the subscription ARN '" << subscriptionARN << "'"
    << std::endl;
    }
}

```

```

        subscriptionARNS.push_back(subscriptionARN);
    }
    else {
        std::cerr << "Error with TopicsAndQueues::Subscribe. "
                    << outcome.GetError().GetMessage()
                    << std::endl;

        cleanUp(topicARN,
                queueURLS,
                subscriptionARNS,
                snsClient,
                sqsClient);

        return false;
    }

```

필터를 사용하여 주제를 구독합니다.

```

static const Aws::String TONE_ATTRIBUTE("tone");
static const Aws::Vector<Aws::String> TONES = {"cheerful", "funny",
"serious",
                                                "sincere"};

Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::SNS::SNSClient snsClient(clientConfig);

Aws::SNS::Model::SubscribeRequest request;
request.SetTopicArn(topicARN);
request.SetProtocol("sqs");
request.SetEndpoint(queueARN);
if (isFifoTopic) {
    if (first) {
        std::cout << "Subscriptions to a FIFO topic can have filters."
                    << std::endl;
        std::cout
            << "If you add a filter to this subscription, then only
the filtered messages "
            << "will be received in the queue." << std::endl;
        std::cout << "For information about message filtering, "

```

```

        << "see https://docs.aws.amazon.com/sns/latest/dg/sns-
message-filtering.html"
        << std::endl;
        std::cout << "For this example, you can filter messages by a \""
        << TONE_ATTRIBUTE << "\" attribute." << std::endl;
    }

    std::ostringstream ostream;
    ostream << "Filter messages for \"" << queueName
        << "\"'s subscription to the topic \""
        << topicName << "\"? (y/n)";

    // Add filter if user answers yes.
    if (askYesNoQuestion(ostream.str())) {
        Aws::String jsonPolicy = getFilterPolicyFromUser();
        if (!jsonPolicy.empty()) {
            filteringMessages = true;

            std::cout << "This is the filter policy for this
subscription."
                << std::endl;
            std::cout << jsonPolicy << std::endl;

            request.AddAttributes("FilterPolicy", jsonPolicy);
        }
        else {
            std::cout
                << "Because you did not select any attributes, no
filter "
                << "will be added to this subscription." <<
std::endl;
        }
    }
} // if (isFifoTopic)
Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

if (outcome.IsSuccess()) {
    Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
    std::cout << "The queue '" << queueName
        << "' has been subscribed to the topic '"
        << "'" << topicName << "'" << std::endl;
    std::cout << "with the subscription ARN '" << subscriptionARN << ".'"

```

```

        << std::endl;
        subscriptionARNS.push_back(subscriptionARN);
    }
    else {
        std::cerr << "Error with TopicsAndQueues::Subscribe. "
        << outcome.GetError().GetMessage()
        << std::endl;

        cleanUp(topicARN,
                queueURLS,
                subscriptionARNS,
                snsClient,
                sqsClient);

        return false;
    }

    //! Routine that lets the user select attributes for a subscription filter policy.
    /*!
    \sa getFilterPolicyFromUser()
    \return Aws::String: The filter policy as JSON.
    */
    Aws::String AwsDoc::TopicsAndQueues::getFilterPolicyFromUser() {
        std::cout
            << "You can filter messages by one or more of the following \""
            << TONE_ATTRIBUTE << "\" attributes." << std::endl;

        std::vector<Aws::String> filterSelections;
        int selection;
        do {
            for (size_t j = 0; j < TONES.size(); ++j) {
                std::cout << " " << (j + 1) << ". " << TONES[j]
                << std::endl;
            }
            selection = askQuestionForIntRange(
                "Enter a number (or enter zero to stop adding more). ",
                0, static_cast<int>(TONES.size()));

            if (selection != 0) {
                const Aws::String &selectedTone(TONES[selection - 1]);
                // Add the tone to the selection if it is not already added.
                if (std::find(filterSelections.begin(),
                    filterSelections.end(),
                    selectedTone)

```

```

        == filterSelections.end()) {
            filterSelections.push_back(selectedTone);
        }
    }
} while (selection != 0);

Aws::String result;
if (!filterSelections.empty()) {
    std::ostringstream jsonPolicyStream;
    jsonPolicyStream << "{ \"\" << TONE_ATTRIBUTE << "\": [";

    for (size_t j = 0; j < filterSelections.size(); ++j) {
        jsonPolicyStream << "\"" << filterSelections[j] << "\"";
        if (j < filterSelections.size() - 1) {
            jsonPolicyStream << ",";
        }
    }
    jsonPolicyStream << "] }";

    result = jsonPolicyStream.str();
}

return result;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [Subscribe](#)를 참조하세요.

## Unsubscribe

다음 코드 예시는 Unsubscribe의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

//! Delete a subscription to an Amazon Simple Notification Service (Amazon SNS)
topic.
/*!
\param subscriptionARN: The Amazon Resource Name (ARN) for an Amazon SNS topic
subscription.
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::SNS::unsubscribe(const Aws::String &subscriptionARN,
                             const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::UnsubscribeRequest request;
    request.SetSubscriptionArn(subscriptionARN);

    const Aws::SNS::Model::UnsubscribeOutcome outcome =
snsClient.Unsubscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Unsubscribed successfully " << std::endl;
    }
    else {
        std::cerr << "Error while unsubscribing " << outcome.GetError().GetMessage()
        << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 [AWS SDK for C++ API 참조](#)의 Unsubscribe를 참조하세요.

## 시나리오

### 사진을 관리하기 위한 서버리스 애플리케이션 만들기

다음 코드 예시에서는 사용자가 레이블을 사용하여 사진을 관리할 수 있는 서버리스 애플리케이션을 생성하는 방법을 보여줍니다.

## SDK for C++

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제의 출처에 대한 자세한 내용은 [AWS 커뮤니티](#)의 게시물을 참조하세요.

이 예제에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

## SMS 문자 메시지 게시

다음 코드 예제에서는 Amazon SNS를 사용하여 SMS 메시지를 게시하는 방법을 보여줍니다.

## SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Publish SMS: use Amazon Simple Notification Service (Amazon SNS) to send an SMS
 * text message to a phone number.
 * Note: This requires additional AWS configuration prior to running example.
 *
 * NOTE: When you start using Amazon SNS to send SMS messages, your AWS account is
 * in the SMS sandbox and you can only
 * use verified destination phone numbers. See https://docs.aws.amazon.com/sns/
 * latest/dg/sns-sms-sandbox.html.
 * NOTE: If destination is in the US, you also have an additional restriction that
 * you have use a dedicated
```

```

* origination ID (phone number). You can request an origination number using
Amazon Pinpoint for a fee.
* See https://aws.amazon.com/blogs/compute/provisioning-and-using-10dlc-origination-numbers-with-amazon-sns/
* for more information.
*
* <phone_number_value> input parameter uses E.164 format.
* For example, in United States, this input value should be of the form:
+12223334444
*/

//! Send an SMS text message to a phone number.
/*!
\param message: The message to publish.
\param phoneNumber: The phone number of the recipient in E.164 format.
\param clientConfiguration: AWS client configuration.
\return bool: Function succeeded.
*/
bool AwsDoc::SNS::publishSms(const Aws::String &message,
                             const Aws::String &phoneNumber,
                             const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetPhoneNumber(phoneNumber);

    const Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

    if (outcome.IsSuccess()) {
        std::cout << "Message published successfully with message id, '"
                    << outcome.GetResult().GetMessageId() << "'."
                    << std::endl;
    }
    else {
        std::cerr << "Error while publishing message "
                    << outcome.GetError().GetMessage()
                    << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [Publish](#)를 참조하세요.

## 대기열에 메시지 게시

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 주제(FIFO 또는 비 FIFO)를 생성합니다.
- 필터 적용 옵션을 사용하여 여러 개의 대기열로 주제를 구독합니다.
- 주제에 메시지를 게시합니다.
- 대기열에서 받은 메시지를 폴링합니다.

## SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예제 리포지토리](#)에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

//! Workflow for messaging with topics and queues using Amazon SNS and Amazon SQS.
/*!
 \param clientConfig Aws client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::TopicsAndQueues::messagingWithTopicsAndQueues(
    const Aws::Client::ClientConfiguration &clientConfiguration) {
    std::cout << "Welcome to messaging with topics and queues." << std::endl;
    printAsterisksLine();
    std::cout << "In this workflow, you will create an SNS topic and subscribe "
        << NUMBER_OF_QUEUES <<
        " SQS queues to the topic." << std::endl;
    std::cout
        << "You can select from several options for configuring the topic and
the subscriptions for the "
```

```

        << NUMBER_OF_QUEUES << " queues." << std::endl;
std::cout << "You can then post to the topic and see the results in the queues."
        << std::endl;

Aws::SNS::SNSClient snsClient(clientConfiguration);

printAsterisksLine();

std::cout << "SNS topics can be configured as FIFO (First-In-First-Out)."
        << std::endl;
std::cout
        << "FIFO topics deliver messages in order and support deduplication and
message filtering."
        << std::endl;
bool isFifoTopic = askYesNoQuestion(
        "Would you like to work with FIFO topics? (y/n) ");

bool contentBasedDeduplication = false;
Aws::String topicName;
if (isFifoTopic) {
    printAsterisksLine();
    std::cout << "Because you have chosen a FIFO topic, deduplication is
supported."
        << std::endl;
    std::cout
        << "Deduplication IDs are either set in the message or automatically
generated "
        << "from content using a hash function." << std::endl;
    std::cout
        << "If a message is successfully published to an SNS FIFO topic, any
message "
        << "published and determined to have the same deduplication ID, "
        << std::endl;
    std::cout
        << "within the five-minute deduplication interval, is accepted but
not delivered."
        << std::endl;
    std::cout
        << "For more information about deduplication, "
        << "see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-
dedup.html."
        << std::endl;
    contentBasedDeduplication = askYesNoQuestion(

```

```

        "Use content-based deduplication instead of entering a deduplication
ID? (y/n) ");
    }

    printAsterisksLine();

    Aws::SQS::SQSClient sqsClient(clientConfiguration);
    Aws::Vector<Aws::String> queueURLS;
    Aws::Vector<Aws::String> subscriptionARNs;

    Aws::String topicARN;
    {
        topicName = askQuestion("Enter a name for your SNS topic. ");

        // 1. Create an Amazon SNS topic, either FIFO or non-FIFO.
        Aws::SNS::Model::CreateTopicRequest request;

        if (isFifoTopic) {
            request.AddAttributes("FifoTopic", "true");
            if (contentBasedDeduplication) {
                request.AddAttributes("ContentBasedDeduplication", "true");
            }
            topicName = topicName + FIFO_SUFFIX;

            std::cout
                << "Because you have selected a FIFO topic, '.fifo' must be
appended to the topic name."
                << std::endl;
        }

        request.SetName(topicName);

        Aws::SNS::Model::CreateTopicOutcome outcome =
snsClient.CreateTopic(request);

        if (outcome.IsSuccess()) {
            topicARN = outcome.GetResult().GetTopicArn();
            std::cout << "Your new topic with the name '" << topicName
                << "' and the topic Amazon Resource Name (ARN) " << std::endl;
            std::cout << "'" << topicARN << "' has been created." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::CreateTopic. "

```

```

        << outcome.GetError().GetMessage()
        << std::endl;

        cleanUp(topicARN,
                queueURLS,
                subscriptionARNS,
                snsClient,
                sqsClient);

        return false;
    }
}

printAsterisksLine();

std::cout << "Now you will create " << NUMBER_OF_QUEUES
           << " SQS queues to subscribe to the topic." << std::endl;
Aws::Vector<Aws::String> queueNames;
bool filteringMessages = false;
bool first = true;
for (int i = 1; i <= NUMBER_OF_QUEUES; ++i) {
    Aws::String queueURL;
    Aws::String queueName;
    {
        printAsterisksLine();
        std::ostringstream ostringstream;
        ostringstream << "Enter a name for " << (first ? "an" : "the next")
                      << " SQS queue. ";
        queueName = askQuestion(ostringstream.str());

        // 2. Create an SQS queue.
        Aws::SQS::Model::CreateQueueRequest request;
        if (isFifoTopic) {
            request.AddAttributes(Aws::SQS::Model::QueueAttributeName::FifoQueue,
                                "true");
            queueName = queueName + FIFO_SUFFIX;

            if (first) // Only explain this once.
            {
                std::cout
                    << "Because you are creating a FIFO SQS queue, '.fifo'
must "
                    << "be appended to the queue name." << std::endl;
            }
        }
    }
}

```

```

    }
}

request.SetQueueName(queueName);
queueNames.push_back(queueName);

Aws::SQS::Model::CreateQueueOutcome outcome =
    sqsClient.CreateQueue(request);

if (outcome.IsSuccess()) {
    queueURL = outcome.GetResult().GetQueueUrl();
    std::cout << "Your new SQS queue with the name '" << queueName
                << "' and the queue URL " << std::endl;
    std::cout << "'" << queueURL << "' has been created." << std::endl;
}
else {
    std::cerr << "Error with SQS::CreateQueue. "
                << outcome.GetError().GetMessage()
                << std::endl;

    cleanUp(topicARN,
            queueURLS,
            subscriptionARNS,
            snsClient,
            sqsClient);

    return false;
}
}
queueURLS.push_back(queueURL);

if (first) // Only explain this once.
{
    std::cout
        << "The queue URL is used to retrieve the queue ARN, which is "
        << "used to create a subscription." << std::endl;
}

Aws::String queueARN;
{
    // 3. Get the SQS queue ARN attribute.
    Aws::SQS::Model::GetQueueAttributesRequest request;
    request.SetQueueUrl(queueURL);

```

```

request.AddAttributeNames(Aws::SQS::Model::QueueAttributeName::QueueArn);

    Aws::SQS::Model::GetQueueAttributesOutcome outcome =
        sqsClient.GetQueueAttributes(request);

    if (outcome.IsSuccess()) {
        const Aws::Map<Aws::SQS::Model::QueueAttributeName, Aws::String>
&attributes =
            outcome.GetResult().GetAttributes();
        const auto &iter = attributes.find(
            Aws::SQS::Model::QueueAttributeName::QueueArn);
        if (iter != attributes.end()) {
            queueARN = iter->second;
            std::cout << "The queue ARN '" << queueARN
                << "' has been retrieved."
                << std::endl;
        }
        else {
            std::cerr
                << "Error ARN attribute not returned by
GetQueueAttribute."
                << std::endl;

            cleanUp(topicARN,
                queueURLs,
                subscriptionARNs,
                snsClient,
                sqsClient);

            return false;
        }
    }
    else {
        std::cerr << "Error with SQS::GetQueueAttributes. "
            << outcome.GetError().GetMessage()
            << std::endl;

        cleanUp(topicARN,
            queueURLs,
            subscriptionARNs,
            snsClient,
            sqsClient);
    }
}

```

```

        return false;
    }
}

if (first) {
    std::cout
        << "An IAM policy must be attached to an SQS queue, enabling it
to receive "
        "messages from an SNS topic." << std::endl;
}

{
    // 4. Set the SQS queue policy attribute with a policy enabling the
receipt of SNS messages.
    Aws::SQS::Model::SetQueueAttributesRequest request;
    request.SetQueueUrl(queueURL);
    Aws::String policy = createPolicyForQueue(queueARN, topicARN);
    request.AddAttributes(Aws::SQS::Model::QueueAttributeName::Policy,
        policy);

    Aws::SQS::Model::SetQueueAttributesOutcome outcome =
        sqsClient.SetQueueAttributes(request);

    if (outcome.IsSuccess()) {
        std::cout << "The attributes for the queue '" << queueName
            << "' were successfully updated." << std::endl;
    }
    else {
        std::cerr << "Error with SQS::SetQueueAttributes. "
            << outcome.GetError().GetMessage()
            << std::endl;

        cleanUp(topicARN,
            queueURLS,
            subscriptionARNS,
            snsClient,
            sqsClient);

        return false;
    }
}

printAsterisksLine();

```

```

{
    // 5. Subscribe the SQS queue to the SNS topic.
    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("sqs");
    request.SetEndpoint(queueARN);
    if (isFifoTopic) {
        if (first) {
            std::cout << "Subscriptions to a FIFO topic can have filters."
                << std::endl;
            std::cout
                << "If you add a filter to this subscription, then only
the filtered messages "
                << "will be received in the queue." << std::endl;
            std::cout << "For information about message filtering, "
                << "see https://docs.aws.amazon.com/sns/latest/dg/sns-
message-filtering.html"
                << std::endl;
            std::cout << "For this example, you can filter messages by a \""
                << TONE_ATTRIBUTE << "\" attribute." << std::endl;
        }

        std::ostringstream ostream;
        ostream << "Filter messages for \"" << queueName
            << "\"'s subscription to the topic \""
            << topicName << "\"? (y/n)";

        // Add filter if user answers yes.
        if (askYesNoQuestion(ostream.str())) {
            Aws::String jsonPolicy = getFilterPolicyFromUser();
            if (!jsonPolicy.empty()) {
                filteringMessages = true;

                std::cout << "This is the filter policy for this
subscription."
                    << std::endl;
                std::cout << jsonPolicy << std::endl;

                request.AddAttributes("FilterPolicy", jsonPolicy);
            }
            else {
                std::cout
                    << "Because you did not select any attributes, no
filter "

```

```

        << "will be added to this subscription." <<

std::endl;
    }
} // if (isFifoTopic)
    Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
        std::cout << "The queue '" << queueName
            << "' has been subscribed to the topic '"
            << "'" << topicName << "'" << std::endl;
        std::cout << "with the subscription ARN '" << subscriptionARN << "."
            << std::endl;
        subscriptionARNS.push_back(subscriptionARN);
    }
    else {
        std::cerr << "Error with TopicsAndQueues::Subscribe. "
            << outcome.GetError().GetMessage()
            << std::endl;

        cleanUp(topicARN,
            queueURLS,
            subscriptionARNS,
            snsClient,
            sqsClient);

        return false;
    }
}

    first = false;
}

first = true;
do {
    printAsterisksLine();

    // 6. Publish a message to the SNS topic.
    Aws::SNS::Model::PublishRequest request;
    request.SetTopicArn(topicARN);
    Aws::String message = askQuestion("Enter a message text to publish. ");

```

```

    request.SetMessage(message);
    if (isFifoTopic) {
        if (first) {
            std::cout
                << "Because you are using a FIFO topic, you must set a
message group ID."
                << std::endl;
            std::cout
                << "All messages within the same group will be received in
the "
                << "order they were published." << std::endl;
        }
        Aws::String messageGroupId = askQuestion(
            "Enter a message group ID for this message. ");
        request.SetMessageGroupId(messageGroupId);
        if (!contentBasedDeduplication) {
            if (first) {
                std::cout
                    << "Because you are not using content-based
deduplication, "
                    << "you must enter a deduplication ID." << std::endl;
            }
            Aws::String deduplicationID = askQuestion(
                "Enter a deduplication ID for this message. ");
            request.SetMessageDeduplicationId(deduplicationID);
        }
    }

    if (filteringMessages && askYesNoQuestion(
        "Add an attribute to this message? (y/n) ")) {
        for (size_t i = 0; i < TONES.size(); ++i) {
            std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
        }
        int selection = askQuestionForIntRange(
            "Enter a number for an attribute. ",
            1, static_cast<int>(TONES.size()));
        Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
        messageAttributeValue.SetDataType("String");
        messageAttributeValue.SetStringValue(TONES[selection - 1]);
        request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
    }

    Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

```

```

        if (outcome.IsSuccess()) {
            std::cout << "Your message was successfully published." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::Publish. "
                << outcome.GetError().GetMessage()
                << std::endl;

            cleanUp(topicARN,
                queueURLS,
                subscriptionARNs,
                snsClient,
                sqsClient);

            return false;
        }

        first = false;
    } while (askYesNoQuestion("Post another message? (y/n) "));

    printAsterisksLine();

    std::cout << "Now the SQS queue will be polled to retrieve the messages."
        << std::endl;
    askQuestion("Press any key to continue...", alwaysTrueTest);

    for (size_t i = 0; i < queueURLS.size(); ++i) {
        // 7. Poll an SQS queue for its messages.
        std::vector<Aws::String> messages;
        std::vector<Aws::String> receiptHandles;
        while (true) {
            Aws::SQS::Model::ReceiveMessageRequest request;
            request.SetMaxNumberOfMessages(10);
            request.SetQueueUrl(queueURLS[i]);

            // Setting WaitTimeSeconds to non-zero enables long polling.
            // For information about long polling, see
            // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
            SQSDeveloperGuide/sqs-short-and-long-polling.html
            request.SetWaitTimeSeconds(1);
            Aws::SQS::Model::ReceiveMessageOutcome outcome =
                sqsClient.ReceiveMessage(request);

            if (outcome.IsSuccess()) {

```

```

        const Aws::Vector<Aws::SQS::Model::Message> &newMessages =
outcome.GetResult().GetMessages();
        if (newMessages.empty()) {
            break;
        }
        else {
            for (const Aws::SQS::Model::Message &message: newMessages) {
                messages.push_back(message.GetBody());
                receiptHandles.push_back(message.GetReceiptHandle());
            }
        }
    }
    else {
        std::cerr << "Error with SQS::ReceiveMessage. "
            << outcome.GetError().GetMessage()
            << std::endl;

        cleanUp(topicARN,
            queueURLS,
            subscriptionARNS,
            snsClient,
            sqsClient);

        return false;
    }
}

printAsterisksLine();

if (messages.empty()) {
    std::cout << "No messages were ";
}
else if (messages.size() == 1) {
    std::cout << "One message was ";
}
else {
    std::cout << messages.size() << " messages were ";
}
std::cout << "received by the queue '" << queueNames[i]
    << "'." << std::endl;
for (const Aws::String &message: messages) {
    std::cout << " Message : '" << message << "'."
        << std::endl;
}

```

```

// 8. Delete a batch of messages from an SQS queue.
if (!receiptHandles.empty()) {
    Aws::SQS::Model::DeleteMessageBatchRequest request;
    request.SetQueueUrl(queueURLS[i]);
    int id = 1; // Ids must be unique within a batch delete request.
    for (const Aws::String &receiptHandle: receiptHandles) {
        Aws::SQS::Model::DeleteMessageBatchRequestEntry entry;
        entry.SetId(std::to_string(id));
        ++id;
        entry.SetReceiptHandle(receiptHandle);
        request.AddEntries(entry);
    }

    Aws::SQS::Model::DeleteMessageBatchOutcome outcome =
        sqsClient.DeleteMessageBatch(request);

    if (outcome.IsSuccess()) {
        std::cout << "The batch deletion of messages was successful."
                    << std::endl;
    }
    else {
        std::cerr << "Error with SQS::DeleteMessageBatch. "
                    << outcome.GetError().GetMessage()
                    << std::endl;
        cleanUp(topicARN,
                queueURLS,
                subscriptionARNs,
                snsClient,
                sqsClient);

        return false;
    }
}

return cleanUp(topicARN,
               queueURLS,
               subscriptionARNs,
               snsClient,
               sqsClient,
               true); // askUser
}

```

```

bool AwsDoc::TopicsAndQueues::cleanUp(const Aws::String &topicARN,
                                       const Aws::Vector<Aws::String> &queueURLS,
                                       const Aws::Vector<Aws::String>
                                       &subscriptionARNs,
                                       const Aws::SNS::SNSClient &snsClient,
                                       const Aws::SQS::SQSClient &sqsClient,
                                       bool askUser) {

    bool result = true;
    printAsterisksLine();
    if (!queueURLS.empty() && askUser &&
        askYesNoQuestion("Delete the SQS queues? (y/n) ")) {

        for (const auto &queueURL: queueURLS) {
            // 9. Delete an SQS queue.
            Aws::SQS::Model::DeleteQueueRequest request;
            request.SetQueueUrl(queueURL);

            Aws::SQS::Model::DeleteQueueOutcome outcome =
                sqsClient.DeleteQueue(request);

            if (outcome.IsSuccess()) {
                std::cout << "The queue with URL '" << queueURL
                    << "' was successfully deleted." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteQueue. "
                    << outcome.GetError().GetMessage()
                    << std::endl;
                result = false;
            }
        }

        for (const auto &subscriptionARN: subscriptionARNs) {
            // 10. Unsubscribe an SNS subscription.
            Aws::SNS::Model::UnsubscribeRequest request;
            request.SetSubscriptionArn(subscriptionARN);

            Aws::SNS::Model::UnsubscribeOutcome outcome =
                snsClient.Unsubscribe(request);

            if (outcome.IsSuccess()) {
                std::cout << "Unsubscribe of subscription ARN '" << subscriptionARN
                    << "' was successful." << std::endl;
            }
        }
    }
}

```

```

        }
        else {
            std::cerr << "Error with TopicsAndQueues::Unsubscribe. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }
}

printAsterisksLine();
if (!topicARN.empty() && askUser &&
    askYesNoQuestion("Delete the SNS topic? (y/n) ")) {

    // 11. Delete an SNS topic.
    Aws::SNS::Model::DeleteTopicRequest request;
    request.SetTopicArn(topicARN);

    Aws::SNS::Model::DeleteTopicOutcome outcome =
snsClient.DeleteTopic(request);

    if (outcome.IsSuccess()) {
        std::cout << "The topic with ARN '" << topicARN
                  << "' was successfully deleted." << std::endl;
    }
    else {
        std::cerr << "Error with TopicsAndQueues::DeleteTopicRequest. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        result = false;
    }
}

return result;
}

//! Create an IAM policy that gives an SQS queue permission to receive messages from
an SNS topic.
/*!
\sa createPolicyForQueue()
\param queueARN: The SQS queue Amazon Resource Name (ARN).
\param topicARN: The SNS topic ARN.
\return Aws::String: The policy as JSON.
*/

```

```

Aws::String AwsDoc::TopicsAndQueues::createPolicyForQueue(const Aws::String
&queueARN,
                                                    const Aws::String
&topicARN) {
    std::ostringstream policyStream;
    policyStream << R"({
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "sns.amazonaws.com"
                },
                "Action": "sqs:SendMessage",
                "Resource": ")" << queueARN << R"(",
                "Condition": {
                    "ArnEquals": {
                        "aws:SourceArn": ")" << topicARN << R"("
                    }
                }
            }
        ]
    })";

    return policyStream.str();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 다음 주제를 참조하세요.

- [CreateQueue](#)
- [CreateTopic](#)
- [DeleteMessageBatch](#)
- [DeleteQueue](#)
- [DeleteTopic](#)
- [GetQueueAttributes](#)
- [Publish](#)
- [ReceiveMessage](#)
- [SetQueueAttributes](#)
- [Subscribe](#)
- [Unsubscribe](#)

# SDK for C++를 사용한 Amazon SQS 예제

다음 코드 예제에서는 AWS SDK for C++ Amazon SQS를 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

- [시작하기](#)
- [작업](#)
- [시나리오](#)

## 시작하기

Hello Amazon SNS

다음 코드 예제에서는 Amazon SQS 사용을 시작하는 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

CMakeLists.txt CMake 파일의 코드입니다.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS sqs)
```

```

# Set this project's name.
project("hello_sqs")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed libraries
  for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
      "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
  endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if(WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
  # Copy relevant AWS SDK for C++ libraries into the current binary directory for
  running and debugging.

  # set(BIN_SUB_DIR "/Debug") # If you are building from the command line you may
  need to uncomment this
  # and set the proper subdirectory to the executables' location.

  AWSSDK_COPY_DYN_LIBS(SERVICE_COMPONENTS ""
    ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif()

add_executable(${PROJECT_NAME}
  hello_sqs.cpp)

target_link_libraries(${PROJECT_NAME}
  ${AWSSDK_LINK_LIBRARIES})

```

hello\_sqs.cpp 소스 파일의 코드입니다.

```
#include <aws/core/Aws.h>
```

```

#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/ListQueuesRequest.h>
#include <iostream>

/*
 * A "Hello SQS" starter application that initializes an Amazon Simple Queue
 * Service
 * (Amazon SQS) client and lists the SQS queues in the current account.
 *
 * main function
 *
 * Usage: 'hello_sqs'
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::Sqs::SQSClient sqsClient(clientConfig);

        Aws::Vector<Aws::String> allQueueUrls;
        Aws::String nextToken; // Next token is used to handle a paginated response.
        do {
            Aws::Sqs::Model::ListQueuesRequest request;

            Aws::Sqs::Model::ListQueuesOutcome outcome =
sqsClient.ListQueues(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::String> &pageOfQueueUrls =
outcome.GetResult().GetQueueUrls();
                if (!pageOfQueueUrls.empty()) {
                    allQueueUrls.insert(allQueueUrls.cend(),
pageOfQueueUrls.cbegin(),
                                pageOfQueueUrls.cend());
                }
            }
        }
    }
}

```

```

        else {
            std::cerr << "Error with SQS::ListQueues. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            break;
        }
        nextToken = outcome.GetResult().GetNextToken();
    } while (!nextToken.empty());

    std::cout << "Hello Amazon SQS! You have " << allQueueUrls.size() << "
queue"
              << (allQueueUrls.size() == 1 ? "" : "s") << " in your account."
              << std::endl;

    if (!allQueueUrls.empty()) {
        std::cout << "Here are your queue URLs." << std::endl;
        for (const Aws::String &queueUrl: allQueueUrls) {
            std::cout << "  * " << queueUrl << std::endl;
        }
    }
}

Aws::ShutdownAPI(options); // Should only be called once.
return 0;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ListQueues](#)를 참조하세요.

## 작업

### ChangeMessageVisibility

다음 코드 예시는 ChangeMessageVisibility의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    /** Changes the visibility timeout of a message in an Amazon Simple Queue Service
    /** (Amazon SQS) queue.
    /**
    /** \param queueUrl: An Amazon SQS queue URL.
    /** \param messageReceiptHandle: A message receipt handle.
    /** \param visibilityTimeoutSeconds: Visibility timeout in seconds.
    /** \param clientConfiguration: AWS client configuration.
    /** \return bool: Function succeeded.
    /**
    bool AwsDoc::SQS::changeMessageVisibility(
        const Aws::String &queue_url,
        const Aws::String &messageReceiptHandle,
        int visibilityTimeoutSeconds,
        const Aws::Client::ClientConfiguration &clientConfiguration) {
        Aws::SQS::SQSClient sqsClient(clientConfiguration);

        Aws::SQS::Model::ChangeMessageVisibilityRequest request;
        request.SetQueueUrl(queue_url);
        request.SetReceiptHandle(messageReceiptHandle);
        request.SetVisibilityTimeout(visibilityTimeoutSeconds);

        auto outcome = sqsClient.ChangeMessageVisibility(request);
        if (outcome.IsSuccess()) {
            std::cout << "Successfully changed visibility of message " <<
                messageReceiptHandle << " from queue " << queue_url << std::endl;
        }
        else {
            std::cout << "Error changing visibility of message from queue "
                << queue_url << ": " <<
                outcome.GetError().GetMessage() << std::endl;
        }
    }

```

```

    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ChangeMessageVisibility](#)를 참조하세요.

## CreateQueue

다음 코드 예시는 CreateQueue의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    //! Create an Amazon Simple Queue Service (Amazon SQS) queue.
    /*!
     * \param queueName: An Amazon SQS queue name.
     * \param clientConfiguration: AWS client configuration.
     * \return bool: Function succeeded.
     */
    bool AwsDoc::SQS::createQueue(const Aws::String &queueName,
                                   const Aws::Client::ClientConfiguration
                                   &clientConfiguration) {
        Aws::SQS::SQSClient sqsClient(clientConfiguration);

        Aws::SQS::Model::CreateQueueRequest request;
        request.SetQueueName(queueName);

        const Aws::SQS::Model::CreateQueueOutcome outcome =
            sqsClient.CreateQueue(request);
        if (outcome.IsSuccess()) {

```

```

        std::cout << "Successfully created queue " << queueName << " with a queue
URL "
                << outcome.GetResult().GetQueueUrl() << "." << std::endl;
    }
    else {
        std::cerr << "Error creating queue " << queueName << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [CreateQueue](#)를 참조하세요.

## DeleteMessage

다음 코드 예시는 DeleteMessage의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    /*! Delete a message from an Amazon Simple Queue Service (Amazon SQS) queue.
    */
    \param queueUrl: An Amazon SQS queue URL.
    \param messageReceiptHandle: A message receipt handle.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */
    bool AwsDoc::SQS::deleteMessage(const Aws::String &queueUrl,
                                    const Aws::String &messageReceiptHandle,
                                    const Aws::Client::ClientConfiguration
&clientConfiguration) {

```

```

    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::DeleteMessageRequest request;
    request.SetQueueUrl(queueUrl);
    request.SetReceiptHandle(messageReceiptHandle);

    const Aws::SQS::Model::DeleteMessageOutcome outcome = sqsClient.DeleteMessage(
        request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted message from queue " << queueUrl
            << std::endl;
    }
    else {
        std::cerr << "Error deleting message from queue " << queueUrl << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteMessage](#)를 참조하세요.

## DeleteMessageBatch

다음 코드 예시는 DeleteMessageBatch의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    Aws::SQS::SQSClient sqsClient(clientConfiguration);

```

```

    Aws::SQS::Model::DeleteMessageBatchRequest request;
    request.SetQueueUrl(queueURLS[i]);
    int id = 1; // Ids must be unique within a batch delete request.
    for (const Aws::String &receiptHandle: receiptHandles) {
        Aws::SQS::Model::DeleteMessageBatchRequestEntry entry;
        entry.SetId(std::to_string(id));
        ++id;
        entry.SetReceiptHandle(receiptHandle);
        request.AddEntries(entry);
    }

    Aws::SQS::Model::DeleteMessageBatchOutcome outcome =
        sqsClient.DeleteMessageBatch(request);

    if (outcome.IsSuccess()) {
        std::cout << "The batch deletion of messages was successful."
                  << std::endl;
    }
    else {
        std::cerr << "Error with SQS::DeleteMessageBatch. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        cleanUp(topicARN,
                queueURLS,
                subscriptionARNS,
                snsClient,
                sqsClient);

        return false;
    }

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteMessageBatch](#)를 참조하세요.

## DeleteQueue

다음 코드 예시는 DeleteQueue의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    //! Delete an Amazon Simple Queue Service (Amazon SQS) queue.
    /*
    \param queueURL: An Amazon SQS queue URL.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */
    bool AwsDoc::SQS::deleteQueue(const Aws::String &queueURL,
                                   const Aws::Client::ClientConfiguration
    &clientConfiguration) {
        Aws::SQS::SQSClient sqsClient(clientConfiguration);
        Aws::SQS::Model::DeleteQueueRequest request;
        request.SetQueueUrl(queueURL);

        const Aws::SQS::Model::DeleteQueueOutcome outcome =
        sqsClient.DeleteQueue(request);
        if (outcome.IsSuccess()) {
            std::cout << "Successfully deleted queue with url " << queueURL <<
            std::endl;
        }
        else {
            std::cerr << "Error deleting queue " << queueURL << ": " <<
            outcome.GetError().GetMessage() << std::endl;
        }
        return outcome.IsSuccess();
    }

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [DeleteQueue](#)를 참조하세요.

## GetQueueAttributes

다음 코드 예시는 GetQueueAttributes의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";

Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::GetQueueAttributesRequest request;
    request.SetQueueUrl(queueURL);

request.AddAttributeNames(Aws::SQS::Model::QueueAttributeName::QueueArn);

    Aws::SQS::Model::GetQueueAttributesOutcome outcome =
        sqsClient.GetQueueAttributes(request);

    if (outcome.IsSuccess()) {
        const Aws::Map<Aws::SQS::Model::QueueAttributeName, Aws::String>
&attributes =
            outcome.GetResult().GetAttributes();
        const auto &iter = attributes.find(
            Aws::SQS::Model::QueueAttributeName::QueueArn);
        if (iter != attributes.end()) {
            queueARN = iter->second;
            std::cout << "The queue ARN '" << queueARN
                << "' has been retrieved."
                << std::endl;
        }
    }
    else {
        std::cerr << "Error with SQS::GetQueueAttributes. "
            << outcome.GetError().GetMessage()
```

```

        << std::endl;

    }

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetQueueAttributes](#)를 참조하세요.

## GetQueueUrl

다음 코드 예시는 GetQueueUrl의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    //! Get the URL for an Amazon Simple Queue Service (Amazon SQS) queue.
    /*!
     * \param queueName: An Amazon SQS queue name.
     * \param clientConfiguration: AWS client configuration.
     * \return bool: Function succeeded.
     */
    bool AwsDoc::SQS::getQueueUrl(const Aws::String &queueName,
                                   const Aws::Client::ClientConfiguration
                                   &clientConfiguration) {
        Aws::SQS::SQSClient sqsClient(clientConfiguration);

        Aws::SQS::Model::GetQueueUrlRequest request;
        request.SetQueueName(queueName);

        const Aws::SQS::Model::GetQueueUrlOutcome outcome =
            sqsClient.GetQueueUrl(request);
        if (outcome.IsSuccess()) {

```

```

        std::cout << "Queue " << queueName << " has url " <<
            outcome.GetResult().GetQueueUrl() << std::endl;
    }
    else {
        std::cerr << "Error getting url for queue " << queueName << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [GetQueueUrl](#)을 참조하세요.

## ListQueues

다음 코드 예시는 ListQueues의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    /*! List the Amazon Simple Queue Service (Amazon SQS) queues within an AWS account.
    */
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */
    bool
    AwsDoc::SQS::listQueues(const Aws::Client::ClientConfiguration &clientConfiguration)
    {
        Aws::SQS::SQSClient sqsClient(clientConfiguration);

        Aws::SQS::Model::ListQueuesRequest listQueuesRequest;
    }

```

```

    Aws::String nextToken; // Used for pagination.
    Aws::Vector<Aws::String> allQueueUrls;

    do {
        if (!nextToken.empty()) {
            listQueuesRequest.SetNextToken(nextToken);
        }
        const Aws::SQS::Model::ListQueuesOutcome outcome = sqsClient.ListQueues(
            listQueuesRequest);
        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::String> &queueUrls =
outcome.GetResult().GetQueueUrls();
            allQueueUrls.insert(allQueueUrls.end(),
                                queueUrls.begin(),
                                queueUrls.end());

            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cerr << "Error listing queues: " <<
                outcome.GetError().GetMessage() << std::endl;
            return false;
        }
    } while (!nextToken.empty());

    std::cout << allQueueUrls.size() << " Amazon SQS queue(s) found." << std::endl;
    for (const auto &iter: allQueueUrls) {
        std::cout << " " << iter << std::endl;
    }

    return true;
}

```

- API에 대한 세부 정보는 AWS SDK for C++ API 참조의 [ListQueues](#)를 참조하세요.

## ReceiveMessage

다음 코드 예시는 ReceiveMessage의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    //! Receive a message from an Amazon Simple Queue Service (Amazon SQS) queue.
    /*!
    \param queueUrl: An Amazon SQS queue URL.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */
    bool AwsDoc::SQS::receiveMessage(const Aws::String &queueUrl,
                                     const Aws::Client::ClientConfiguration
&clientConfiguration) {
        Aws::SQS::SQSClient sqsClient(clientConfiguration);

        Aws::SQS::Model::ReceiveMessageRequest request;
        request.SetQueueUrl(queueUrl);
        request.SetMaxNumberOfMessages(1);

        const Aws::SQS::Model::ReceiveMessageOutcome outcome = sqsClient.ReceiveMessage(
            request);
        if (outcome.IsSuccess()) {

            const Aws::Vector<Aws::SQS::Model::Message> &messages =
                outcome.GetResult().GetMessages();
            if (!messages.empty()) {
                const Aws::SQS::Model::Message &message = messages[0];
                std::cout << "Received message:" << std::endl;
                std::cout << "  MessageId: " << message.GetMessageId() << std::endl;
                std::cout << "  ReceiptHandle: " << message.GetReceiptHandle() <<
std::endl;
                std::cout << "  Body: " << message.GetBody() << std::endl << std::endl;
            }
        }
        else {

```

```

        std::cout << "No messages received from queue " << queueUrl <<
            std::endl;
    }
}
else {
    std::cerr << "Error receiving message from queue " << queueUrl << ": "
        << outcome.GetError().GetMessage() << std::endl;
}
return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [ReceiveMessage](#)를 참조하세요.

## SendMessage

다음 코드 예시는 SendMessage의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    /*! Send a message to an Amazon Simple Queue Service (Amazon SQS) queue.
    /*!
    \param queueUrl: An Amazon SQS queue URL.
    \param messageBody: A message body.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */
    bool AwsDoc::SQS::sendMessage(const Aws::String &queueUrl,
                                   const Aws::String &messageBody,
                                   const Aws::Client::ClientConfiguration
    &clientConfiguration) {

```

```

    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::SendMessageRequest request;
    request.SetQueueUrl(queueUrl);
    request.SetMessageBody(messageBody);

    const Aws::SQS::Model::SendMessageOutcome outcome =
    sqsClient.SendMessage(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully sent message to " << queueUrl <<
            std::endl;
    }
    else {
        std::cerr << "Error sending message to " << queueUrl << ": " <<
            outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [SendMessage](#)를 참조하세요.

## SetQueueAttributes

다음 코드 예시는 SetQueueAttributes의 사용 방법을 보여줍니다.

SDK for C++

### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    //! Set the value for an attribute in an Amazon Simple Queue Service (Amazon SQS)
    queue.

```

```

/ * !
\ param queueUrl: An Amazon SQS queue URL.
\ param attributeName: An attribute name enum.
\ param attribute: The attribute value as a string.
\ param clientConfiguration: AWS client configuration.
\ return bool: Function succeeded.
* /
bool AwsDoc::SQS::setQueueAttributes(const Aws::String &queueURL,
                                     Aws::SQS::Model::QueueAttributeName
attributeName,
                                     const Aws::String &attribute,
                                     const Aws::Client::ClientConfiguration
&clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

    Aws::SQS::Model::SetQueueAttributesRequest request;
    request.SetQueueUrl(queueURL);
    request.AddAttributes(
        attributeName,
        attribute);

    const Aws::SQS::Model::SetQueueAttributesOutcome outcome =
sqsClient.SetQueueAttributes(
    request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully set the attribute " <<

    Aws::SQS::Model::QueueAttributeNameMapper::GetNameForQueueAttributeName(
        attributeName)
        << " with value " << attribute << " in queue " <<
queueURL << "." << std::endl;
    }
    else {
        std::cout << "Error setting attribute for queue " <<
queueURL << ": " << outcome.GetError().GetMessage() <<
std::endl;
    }

    return outcome.IsSuccess();
}

```

배달 못한 편지 대기열을 구성합니다.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    //! Connect an Amazon Simple Queue Service (Amazon SQS) queue to an associated
    //! dead-letter queue.
    /*!
    \param srcQueueUrl: An Amazon SQS queue URL.
    \param deadLetterQueueARN: The Amazon Resource Name (ARN) of an Amazon SQS dead-
    letter queue.
    \param maxReceiveCount: The max receive count of a message before it is sent to
    the dead-letter queue.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */
    bool AwsDoc::SQS::setDeadLetterQueue(const Aws::String &srcQueueUrl,
                                         const Aws::String &deadLetterQueueARN,
                                         int maxReceiveCount,
                                         const Aws::Client::ClientConfiguration
    &clientConfiguration) {
        Aws::String redrivePolicy = MakeRedrivePolicy(deadLetterQueueARN,
    maxReceiveCount);

        Aws::SQS::SQSClient sqsClient(clientConfiguration);

        Aws::SQS::Model::SetQueueAttributesRequest request;
        request.SetQueueUrl(srcQueueUrl);
        request.AddAttributes(
            Aws::SQS::Model::QueueAttributeName::RedrivePolicy,
            redrivePolicy);

        const Aws::SQS::Model::SetQueueAttributesOutcome outcome =
            sqsClient.SetQueueAttributes(request);
        if (outcome.IsSuccess()) {
            std::cout << "Successfully set dead letter queue for queue " <<
                srcQueueUrl << " to " << deadLetterQueueARN << std::endl;
        }
        else {
            std::cerr << "Error setting dead letter queue for queue " <<
                srcQueueUrl << ": " << outcome.GetError().GetMessage() <<
                std::endl;
        }
    }

```

```

    return outcome.IsSuccess();
}

//! Make a redrive policy for a dead-letter queue.
/*!
    \param queueArn: An Amazon SQS ARN for the dead-letter queue.
    \param maxReceiveCount: The max receive count of a message before it is sent to
    the dead-letter queue.
    \return Aws::String: Policy as JSON string.
    */
Aws::String MakeRedrivePolicy(const Aws::String &queueArn, int maxReceiveCount) {
    Aws::Utils::Json::JsonValue redrive_arn_entry;
    redrive_arn_entry.AsString(queueArn);

    Aws::Utils::Json::JsonValue max_msg_entry;
    max_msg_entry.AsInteger(maxReceiveCount);

    Aws::Utils::Json::JsonValue policy_map;
    policy_map.WithObject("deadLetterTargetArn", redrive_arn_entry);
    policy_map.WithObject("maxReceiveCount", max_msg_entry);

    return policy_map.View().WriteReadable();
}

```

긴 폴링을 사용하도록 Amazon SQS 대기열을 구성합니다.

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region (overrides config file).
    // clientConfig.region = "us-east-1";

    //! Set the wait time for an Amazon Simple Queue Service (Amazon SQS) queue poll.
    /*!
    \param queueUrl: An Amazon SQS queue URL.
    \param pollTimeSeconds: The receive message wait time in seconds.
    \param clientConfiguration: AWS client configuration.
    \return bool: Function succeeded.
    */
bool AwsDoc::SQS::setQueueLongPollingAttribute(const Aws::String &queueURL,
                                                const Aws::String &pollTimeSeconds,
                                                const
    Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::SQS::SQSClient sqsClient(clientConfiguration);

```

```

    Aws::SQS::Model::SetQueueAttributesRequest request;
    request.SetQueueUrl(queueURL);
    request.AddAttributes(
        Aws::SQS::Model::QueueAttributeName::ReceiveMessageWaitTimeSeconds,
        pollTimeSeconds);

    const Aws::SQS::Model::SetQueueAttributesOutcome outcome =
    sqsClient.SetQueueAttributes(
        request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated long polling time for queue " <<
            queueURL << " to " << pollTimeSeconds << std::endl;
    }
    else {
        std::cout << "Error updating long polling time for queue " <<
            queueURL << ": " << outcome.GetError().GetMessage() <<
            std::endl;
    }

    return outcome.IsSuccess();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [SetQueueAttributes](#)를 참조하세요.

## 시나리오

### 대기열에 메시지 게시

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 주제(FIFO 또는 비 FIFO)를 생성합니다.
- 필터 적용 옵션을 사용하여 여러 개의 대기열로 주제를 구독합니다.
- 주제에 메시지를 게시합니다.
- 대기열에서 받은 메시지를 폴링합니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예제 리포지토리](#)에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```

        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    //! Workflow for messaging with topics and queues using Amazon SNS and Amazon SQS.
    /*!
    \param clientConfig Aws client configuration.
    \return bool: Successful completion.
    */
    bool AwsDoc::TopicsAndQueues::messagingWithTopicsAndQueues(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
        std::cout << "Welcome to messaging with topics and queues." << std::endl;
        printAsterisksLine();
        std::cout << "In this workflow, you will create an SNS topic and subscribe "
            << NUMBER_OF_QUEUES <<
            " SQS queues to the topic." << std::endl;
        std::cout
            << "You can select from several options for configuring the topic and
the subscriptions for the "
            << NUMBER_OF_QUEUES << " queues." << std::endl;
        std::cout << "You can then post to the topic and see the results in the queues."
            << std::endl;

        Aws::SNS::SNSClient snsClient(clientConfiguration);

        printAsterisksLine();

        std::cout << "SNS topics can be configured as FIFO (First-In-First-Out)."
            << std::endl;
        std::cout
            << "FIFO topics deliver messages in order and support deduplication and
message filtering."
            << std::endl;
        bool isFifoTopic = askYesNoQuestion(

```

```

        "Would you like to work with FIFO topics? (y/n) ");

    bool contentBasedDeduplication = false;
    Aws::String topicName;
    if (isFifoTopic) {
        printAsterisksLine();
        std::cout << "Because you have chosen a FIFO topic, deduplication is
supported."
                    << std::endl;
        std::cout
            << "Deduplication IDs are either set in the message or automatically
generated "
            << "from content using a hash function." << std::endl;
        std::cout
            << "If a message is successfully published to an SNS FIFO topic, any
message "
            << "published and determined to have the same deduplication ID, "
            << std::endl;
        std::cout
            << "within the five-minute deduplication interval, is accepted but
not delivered."
            << std::endl;
        std::cout
            << "For more information about deduplication, "
            << "see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-
dedup.html."
            << std::endl;
        contentBasedDeduplication = askYesNoQuestion(
            "Use content-based deduplication instead of entering a deduplication
ID? (y/n) ");
    }

    printAsterisksLine();

    Aws::SQS::SQSClient sqsClient(clientConfiguration);
    Aws::Vector<Aws::String> queueURLS;
    Aws::Vector<Aws::String> subscriptionARNs;

    Aws::String topicARN;
    {
        topicName = askQuestion("Enter a name for your SNS topic. ");

        // 1. Create an Amazon SNS topic, either FIFO or non-FIFO.
        Aws::SNS::Model::CreateTopicRequest request;

```

```

    if (isFifoTopic) {
        request.AddAttributes("FifoTopic", "true");
        if (contentBasedDeduplication) {
            request.AddAttributes("ContentBasedDeduplication", "true");
        }
        topicName = topicName + FIFO_SUFFIX;

        std::cout
            << "Because you have selected a FIFO topic, '.fifo' must be
appended to the topic name."
            << std::endl;
    }

    request.SetName(topicName);

    Aws::SNS::Model::CreateTopicOutcome outcome =
snsClient.CreateTopic(request);

    if (outcome.IsSuccess()) {
        topicARN = outcome.GetResult().GetTopicArn();
        std::cout << "Your new topic with the name '" << topicName
            << "' and the topic Amazon Resource Name (ARN) " << std::endl;
        std::cout << "'" << topicARN << "' has been created." << std::endl;
    }
    else {
        std::cerr << "Error with TopicsAndQueues::CreateTopic. "
            << outcome.GetError().GetMessage()
            << std::endl;

        cleanUp(topicARN,
            queueURLS,
            subscriptionARNS,
            snsClient,
            sqsClient);

        return false;
    }
}

printAsterisksLine();

std::cout << "Now you will create " << NUMBER_OF_QUEUES

```

```

        << " SQS queues to subscribe to the topic." << std::endl;
    Aws::Vector<Aws::String> queueNames;
    bool filteringMessages = false;
    bool first = true;
    for (int i = 1; i <= NUMBER_OF_QUEUES; ++i) {
        Aws::String queueURL;
        Aws::String queueName;
        {
            printAsterisksLine();
            std::ostringstream ostringstream;
            ostringstream << "Enter a name for " << (first ? "an" : "the next")
                << " SQS queue. ";
            queueName = askQuestion(ostringstream.str());

            // 2. Create an SQS queue.
            Aws::SQS::Model::CreateQueueRequest request;
            if (isFifoTopic) {

request.AddAttributes(Aws::SQS::Model::QueueAttributeName::FifoQueue,
                        "true");
                queueName = queueName + FIFO_SUFFIX;

                if (first) // Only explain this once.
                {
                    std::cout
                        << "Because you are creating a FIFO SQS queue, '.fifo'
must "
                        << "be appended to the queue name." << std::endl;
                }
            }

            request.SetQueueName(queueName);
            queueNames.push_back(queueName);

            Aws::SQS::Model::CreateQueueOutcome outcome =
                sqsClient.CreateQueue(request);

            if (outcome.IsSuccess()) {
                queueURL = outcome.GetResult().GetQueueUrl();
                std::cout << "Your new SQS queue with the name '" << queueName
                    << "' and the queue URL " << std::endl;
                std::cout << "'" << queueURL << "' has been created." << std::endl;
            }
            else {

```

```

        std::cerr << "Error with SQS::CreateQueue. "
                   << outcome.GetError().GetMessage()
                   << std::endl;

        cleanUp(topicARN,
                queueURLS,
                subscriptionARNs,
                snsClient,
                sqsClient);

        return false;
    }
}
queueURLS.push_back(queueURL);

if (first) // Only explain this once.
{
    std::cout
        << "The queue URL is used to retrieve the queue ARN, which is "
        << "used to create a subscription." << std::endl;
}

Aws::String queueARN;
{
    // 3. Get the SQS queue ARN attribute.
    Aws::SQS::Model::GetQueueAttributesRequest request;
    request.SetQueueUrl(queueURL);

    request.AddAttributeNames(Aws::SQS::Model::QueueAttributeName::QueueArn);

    Aws::SQS::Model::GetQueueAttributesOutcome outcome =
        sqsClient.GetQueueAttributes(request);

    if (outcome.IsSuccess()) {
        const Aws::Map<Aws::SQS::Model::QueueAttributeName, Aws::String>
&attributes =
            outcome.GetResult().GetAttributes();
        const auto &iter = attributes.find(
            Aws::SQS::Model::QueueAttributeName::QueueArn);
        if (iter != attributes.end()) {
            queueARN = iter->second;
            std::cout << "The queue ARN '" << queueARN
                      << "' has been retrieved."
                      << std::endl;
        }
    }
}

```

```

        }
        else {
            std::cerr
                << "Error ARN attribute not returned by
GetQueueAttribute."
                << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
    }
    else {
        std::cerr << "Error with SQS::GetQueueAttributes. "
            << outcome.GetError().GetMessage()
            << std::endl;

        cleanUp(topicARN,
                queueURLS,
                subscriptionARNS,
                snsClient,
                sqsClient);

        return false;
    }
}

if (first) {
    std::cout
        << "An IAM policy must be attached to an SQS queue, enabling it
to receive "
        << "messages from an SNS topic." << std::endl;
}

{
    // 4. Set the SQS queue policy attribute with a policy enabling the
receipt of SNS messages.
    Aws::SQS::Model::SetQueueAttributesRequest request;
    request.SetQueueUrl(queueURL);
    Aws::String policy = createPolicyForQueue(queueARN, topicARN);

```

```

        request.AddAttributes(Aws::SQS::Model::QueueAttributeName::Policy,
                               policy);

        Aws::SQS::Model::SetQueueAttributesOutcome outcome =
            sqsClient.SetQueueAttributes(request);

        if (outcome.IsSuccess()) {
            std::cout << "The attributes for the queue '" << queueName
                       << "' were successfully updated." << std::endl;
        }
        else {
            std::cerr << "Error with SQS::SetQueueAttributes. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
    }

    printAsterisksLine();

    {
        // 5. Subscribe the SQS queue to the SNS topic.
        Aws::SNS::Model::SubscribeRequest request;
        request.SetTopicArn(topicARN);
        request.SetProtocol("sqs");
        request.SetEndpoint(queueARN);
        if (isFifoTopic) {
            if (first) {
                std::cout << "Subscriptions to a FIFO topic can have filters."
                          << std::endl;

                std::cout
                    << "If you add a filter to this subscription, then only
the filtered messages "
                    << "will be received in the queue." << std::endl;
                std::cout << "For information about message filtering, "
                          << "see https://docs.aws.amazon.com/sns/latest/dg/sns-
message-filtering.html"

```

```

        << std::endl;
        std::cout << "For this example, you can filter messages by a \""
        << TONE_ATTRIBUTE << "\" attribute." << std::endl;
    }

    std::ostringstream ostream;
    ostream << "Filter messages for \"" << queueName
    << "\"'s subscription to the topic \""
    << topicName << "\"? (y/n)";

    // Add filter if user answers yes.
    if (askYesNoQuestion(ostream.str())) {
        Aws::String jsonPolicy = getFilterPolicyFromUser();
        if (!jsonPolicy.empty()) {
            filteringMessages = true;

            std::cout << "This is the filter policy for this
subscription."
                        << std::endl;
            std::cout << jsonPolicy << std::endl;

            request.AddAttributes("FilterPolicy", jsonPolicy);
        }
        else {
            std::cout
                << "Because you did not select any attributes, no
filter "
                << "will be added to this subscription." <<
std::endl;
        }
    } // if (isFifoTopic)
    Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
        std::cout << "The queue '" << queueName
        << "' has been subscribed to the topic '"
        << "'" << topicName << "'" << std::endl;
        std::cout << "with the subscription ARN '" << subscriptionARN << "."
        << std::endl;
        subscriptionARNS.push_back(subscriptionARN);
    }
}

```

```

    }
    else {
        std::cerr << "Error with TopicsAndQueues::Subscribe. "
                  << outcome.GetError().GetMessage()
                  << std::endl;

        cleanUp(topicARN,
                queueURLS,
                subscriptionARNS,
                snsClient,
                sqsClient);

        return false;
    }
}

first = false;
}

first = true;
do {
    printAsterisksLine();

    // 6. Publish a message to the SNS topic.
    Aws::SNS::Model::PublishRequest request;
    request.SetTopicArn(topicARN);
    Aws::String message = askQuestion("Enter a message text to publish. ");
    request.SetMessage(message);
    if (isFifoTopic) {
        if (first) {
            std::cout
                << "Because you are using a FIFO topic, you must set a
message group ID."
                << std::endl;
            std::cout
                << "All messages within the same group will be received in
the "
                << "order they were published." << std::endl;
        }
        Aws::String messageGroupID = askQuestion(
            "Enter a message group ID for this message. ");
        request.SetMessageGroupId(messageGroupID);
        if (!contentBasedDeduplication) {
            if (first) {

```

```

        std::cout
            << "Because you are not using content-based
deduplication, "
            << "you must enter a deduplication ID." << std::endl;
    }
    Aws::String deduplicationID = askQuestion(
        "Enter a deduplication ID for this message. ");
    request.SetMessageDeduplicationId(deduplicationID);
}
}

if (filteringMessages && askYesNoQuestion(
    "Add an attribute to this message? (y/n) ")) {
    for (size_t i = 0; i < TONES.size(); ++i) {
        std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
    }
    int selection = askQuestionForIntRange(
        "Enter a number for an attribute. ",
        1, static_cast<int>(TONES.size()));
    Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
    messageAttributeValue.SetDataType("String");
    messageAttributeValue.SetStringValue(TONES[selection - 1]);
    request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
}

Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

if (outcome.IsSuccess()) {
    std::cout << "Your message was successfully published." << std::endl;
}
else {
    std::cerr << "Error with TopicsAndQueues::Publish. "
        << outcome.GetError().GetMessage()
        << std::endl;

    cleanUp(topicARN,
        queueURLS,
        subscriptionARNS,
        snsClient,
        sqsClient);

    return false;
}

```

```

        first = false;
    } while (askYesNoQuestion("Post another message? (y/n) "));

    printAsterisksLine();

    std::cout << "Now the SQS queue will be polled to retrieve the messages."
               << std::endl;
    askQuestion("Press any key to continue...", alwaysTrueTest);

    for (size_t i = 0; i < queueURLS.size(); ++i) {
        // 7. Poll an SQS queue for its messages.
        std::vector<Aws::String> messages;
        std::vector<Aws::String> receiptHandles;
        while (true) {
            Aws::SQS::Model::ReceiveMessageRequest request;
            request.SetMaxNumberOfMessages(10);
            request.SetQueueUrl(queueURLS[i]);

            // Setting WaitTimeSeconds to non-zero enables long polling.
            // For information about long polling, see
            // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
            request.SetWaitTimeSeconds(1);
            Aws::SQS::Model::ReceiveMessageOutcome outcome =
                sqsClient.ReceiveMessage(request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::SQS::Model::Message> &newMessages =
outcome.GetResult().GetMessages();
                if (newMessages.empty()) {
                    break;
                }
                else {
                    for (const Aws::SQS::Model::Message &message: newMessages) {
                        messages.push_back(message.GetBody());
                        receiptHandles.push_back(message.GetReceiptHandle());
                    }
                }
            }
            else {
                std::cerr << "Error with SQS::ReceiveMessage. "
                           << outcome.GetError().GetMessage()
                           << std::endl;
            }
        }
    }
}

```

```

        cleanUp(topicARN,
                queueURLS,
                subscriptionARNS,
                snsClient,
                sqsClient);

        return false;
    }
}

printAsterisksLine();

if (messages.empty()) {
    std::cout << "No messages were ";
}
else if (messages.size() == 1) {
    std::cout << "One message was ";
}
else {
    std::cout << messages.size() << " messages were ";
}
std::cout << "received by the queue '" << queueNames[i]
            << "'." << std::endl;
for (const Aws::String &message: messages) {
    std::cout << " Message : '" << message << "'."
            << std::endl;
}

// 8. Delete a batch of messages from an SQS queue.
if (!receiptHandles.empty()) {
    Aws::SQS::Model::DeleteMessageBatchRequest request;
    request.SetQueueUrl(queueURLS[i]);
    int id = 1; // Ids must be unique within a batch delete request.
    for (const Aws::String &receiptHandle: receiptHandles) {
        Aws::SQS::Model::DeleteMessageBatchRequestEntry entry;
        entry.SetId(std::to_string(id));
        ++id;
        entry.SetReceiptHandle(receiptHandle);
        request.AddEntries(entry);
    }

    Aws::SQS::Model::DeleteMessageBatchOutcome outcome =
        sqsClient.DeleteMessageBatch(request);
}

```

```

        if (outcome.IsSuccess()) {
            std::cout << "The batch deletion of messages was successful."
                      << std::endl;
        }
        else {
            std::cerr << "Error with SQS::DeleteMessageBatch. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
    }
}

return cleanUp(topicARN,
                queueURLS,
                subscriptionARNS,
                snsClient,
                sqsClient,
                true); // askUser
}

bool AwsDoc::TopicsAndQueues::cleanUp(const Aws::String &topicARN,
                                       const Aws::Vector<Aws::String> &queueURLS,
                                       const Aws::Vector<Aws::String>
                                       &subscriptionARNS,
                                       const Aws::SNS::SNSClient &snsClient,
                                       const Aws::SQS::SQSClient &sqsClient,
                                       bool askUser) {

    bool result = true;
    printAsterisksLine();
    if (!queueURLS.empty() && askUser &&
        askYesNoQuestion("Delete the SQS queues? (y/n) ")) {

        for (const auto &queueURL: queueURLS) {
            // 9. Delete an SQS queue.
            Aws::SQS::Model::DeleteQueueRequest request;
            request.SetQueueUrl(queueURL);

```

```

        Aws::SQS::Model::DeleteQueueOutcome outcome =
            sqsClient.DeleteQueue(request);

        if (outcome.IsSuccess()) {
            std::cout << "The queue with URL '" << queueURL
                << "' was successfully deleted." << std::endl;
        }
        else {
            std::cerr << "Error with SQS::DeleteQueue. "
                << outcome.GetError().GetMessage()
                << std::endl;
            result = false;
        }
    }

    for (const auto &subscriptionARN: subscriptionARNS) {
        // 10. Unsubscribe an SNS subscription.
        Aws::SNS::Model::UnsubscribeRequest request;
        request.SetSubscriptionArn(subscriptionARN);

        Aws::SNS::Model::UnsubscribeOutcome outcome =
            snsClient.Unsubscribe(request);

        if (outcome.IsSuccess()) {
            std::cout << "Unsubscribe of subscription ARN '" << subscriptionARN
                << "' was successful." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::Unsubscribe. "
                << outcome.GetError().GetMessage()
                << std::endl;
            result = false;
        }
    }
}

printAsterisksLine();
if (!topicARN.empty() && askUser &&
    askYesNoQuestion("Delete the SNS topic? (y/n) ")) {

    // 11. Delete an SNS topic.
    Aws::SNS::Model::DeleteTopicRequest request;
    request.SetTopicArn(topicARN);

```

```

    Aws::SNS::Model::DeleteTopicOutcome outcome =
snsClient.DeleteTopic(request);

    if (outcome.IsSuccess()) {
        std::cout << "The topic with ARN '" << topicARN
            << "' was successfully deleted." << std::endl;
    }
    else {
        std::cerr << "Error with TopicsAndQueues::DeleteTopicRequest. "
            << outcome.GetError().GetMessage()
            << std::endl;
        result = false;
    }
}

return result;
}

//! Create an IAM policy that gives an SQS queue permission to receive messages from
an SNS topic.
/*!
\sa createPolicyForQueue()
\param queueARN: The SQS queue Amazon Resource Name (ARN).
\param topicARN: The SNS topic ARN.
\return Aws::String: The policy as JSON.
*/
Aws::String AwsDoc::TopicsAndQueues::createPolicyForQueue(const Aws::String
&queueARN,
                                                    const Aws::String
&topicARN) {
    std::ostringstream policyStream;
    policyStream << R"({
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "sns.amazonaws.com"
                },
                "Action": "sqs:SendMessage",
                "Resource": ")" << queueARN << R"(",
                "Condition": {
                    "ArnEquals": {
                        "aws:SourceArn": ")" << topicARN << R"("

```

```

    }
    }
    ]
  })";

  return policyStream.str();
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 다음 주제를 참조하세요.

- [CreateQueue](#)
- [CreateTopic](#)
- [DeleteMessageBatch](#)
- [DeleteQueue](#)
- [DeleteTopic](#)
- [GetQueueAttributes](#)
- [Publish](#)
- [ReceiveMessage](#)
- [SetQueueAttributes](#)
- [Subscribe](#)
- [Unsubscribe](#)

## AWS STS SDK for C++를 사용한 예제

다음 코드 예제에서는와 AWS SDK for C++ 함께를 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 AssumeRole의 사용 방법을 보여줍니다.

SDK for C++

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
bool AwsDoc::STS::assumeRole(const Aws::String &roleArn,
                             const Aws::String &roleSessionName,
                             const Aws::String &externalId,
                             Aws::Auth::AWSCredentials &credentials,
                             const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::STS::STSClient sts(clientConfig);
    Aws::STS::Model::AssumeRoleRequest sts_req;

    sts_req.SetRoleArn(roleArn);
    sts_req.SetRoleSessionName(roleSessionName);
    sts_req.SetExternalId(externalId);

    const Aws::STS::Model::AssumeRoleOutcome outcome = sts.AssumeRole(sts_req);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error assuming IAM role. " <<
            outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Credentials successfully retrieved." << std::endl;
        const Aws::STS::Model::AssumeRoleResult result = outcome.GetResult();
        const Aws::STS::Model::Credentials &temp_credentials =
            result.GetCredentials();

        // Store temporary credentials in return argument.
        // Note: The credentials object returned by assumeRole differs
        // from the AWSCredentials object used in most situations.
        credentials.SetAWSAccessKeyId(temp_credentials.GetAccessKeyId());
    }
}
```

```
        credentials.SetAWSSecretKey(temp_credentials.GetSecretAccessKey());
        credentials.SetSessionToken(temp_credentials.GetSessionToken());
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 [AssumeRole](#)을 참조하세요.

## SDK for C++를 사용한 Amazon Transcribe 스트리밍 예제

다음 코드 예제에서는 Amazon Transcribe Streaming과 AWS SDK for C++ 함께를 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](#)
- [시나리오](#)

## 작업

### StartStreamTranscription

다음 코드 예시는 StartStreamTranscription의 사용 방법을 보여줍니다.

## SDK for C++

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
int main() {
    Aws::SDKOptions options;

    Aws::InitAPI(options);
    {
        //TODO(User): Set to the region of your AWS account.
        const Aws::String region = Aws::Region::US_WEST_2;

        //Load a profile that has been granted AmazonTranscribeFullAccess AWS
        managed permission policy.
        Aws::Client::ClientConfiguration config;
#ifdef _WIN32
        // ATTENTION: On Windows with the AWS C++ SDK, this example only runs if the
        SDK is built
        // with the curl library.
        // For more information, see the accompanying ReadMe.
        // For more information, see "Building the SDK for Windows with curl".
        // https://docs.aws.amazon.com/sdk-for-cpp/v1/developer-guide/setup-
        windows.html
        //TODO(User): Update to the location of your .crt file.
        config.caFile = "C:/curl/bin/curl-ca-bundle.crt";
#endif
        config.region = region;

        TranscribeStreamingServiceClient client(config);
        StartStreamTranscriptionHandler handler;
        handler.SetOnErrorCallback(
            [](const Aws::Client::AWSError<TranscribeStreamingServiceErrors>
            &error) {
                std::cerr << "ERROR: " + error.GetMessage() << std::endl;
            });
        //SetTranscriptEventCallback called for every 'chunk' of file transcribed.
        // Partial results are returned in real time.
        handler.SetTranscriptEventCallback([](const TranscriptEvent &ev) {
```

```

        for (auto &&r: ev.GetTranscript().GetResults()) {
            if (r.GetIsPartial()) {
                std::cout << "[partial] ";
            }
            else {
                std::cout << "[Final] ";
            }
            for (auto &&alt: r.GetAlternatives()) {
                std::cout << alt.GetTranscript() << std::endl;
            }
        }
    });

    StartStreamTranscriptionRequest request;
    request.SetMediaSampleRateHertz(SAMPLE_RATE);
    request.SetLanguageCode(LanguageCode::en_US);
    request.SetMediaEncoding(
        MediaEncoding::pcm); // wav and aiff files are PCM formats.
    request.SetEventStreamHandler(handler);

    auto OnStreamReady = [](AudioStream &stream) {
        Aws::FStream file(FILE_NAME, std::ios_base::in |
std::ios_base::binary);
        if (!file.is_open()) {
            std::cerr << "Failed to open " << FILE_NAME << '\n';
        }
        std::array<char, BUFFER_SIZE> buf;
        int i = 0;
        while (file) {
            file.read(&buf[0], buf.size());

            if (!file)
                std::cout << "File: only " << file.gcount() << " could be
read"
                    << std::endl;

            Aws::Vector<unsigned char> bits{buf.begin(), buf.end()};
            AudioEvent event(std::move(bits));
            if (!stream) {
                std::cerr << "Failed to create a stream" << std::endl;
                break;
            }
            //The std::basic_istream::gcount() is used to count the
characters in the given string. It returns

```

```

        //the number of characters extracted by the last read()
operation.
        if (file.gcount() > 0) {
            if (!stream.WriteAudioEvent(event)) {
                std::cerr << "Failed to write an audio event" <<
std::endl;
                break;
            }
        }
        else {
            break;
        }
        std::this_thread::sleep_for(std::chrono::milliseconds(
            25)); // Slow down because we are streaming from a file.
    }
    if (!stream.WriteAudioEvent(
        AudioEvent())) {
        // Per the spec, we have to send an empty event (an event
without a payload) at the end.
        std::cerr << "Failed to send an empty frame" << std::endl;
    }
    else {
        std::cout << "Successfully sent the empty frame" << std::endl;
    }
    stream.flush();
    stream.Close();
};

    Aws::Utils::Threading::Semaphore signaling(0 /*initialCount*/, 1 /
*maxCount*/);
    auto OnResponseCallback = [&signaling](
        const TranscribeStreamingServiceClient * /*unused*/,
        const Model::StartStreamTranscriptionRequest & /*unused*/,
        const Model::StartStreamTranscriptionOutcome &outcome,
        const std::shared_ptr<const Aws::Client::AsyncCallerContext> & /
*unused*/) {

        if (!outcome.IsSuccess()) {
            std::cerr << "Transcribe streaming error "
                << outcome.GetError().GetMessage() << std::endl;
        }

        signaling.Release();
    };
};

```

```

        std::cout << "Starting..." << std::endl;
        client.StartStreamTranscriptionAsync(request, OnStreamReady,
        OnResponseCallback,
                                   nullptr /*context*/);
        signaling.WaitOne(); // Prevent the application from exiting until we're
done.
        std::cout << "Done" << std::endl;
    }

    Aws::ShutdownAPI(options);

    return 0;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [StartStreamTranscription](#)을 참조하세요.

## 시나리오

### 오디오 파일 트랜스크립션

다음 코드 예제에서는 Amazon Transcribe 스트리밍을 사용하여 소스 오디오 파일의 트랜스크립션을 생성하는 방법을 보여줍니다.

#### SDK for C++

#### Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

int main() {
    Aws::SDKOptions options;

    Aws::InitAPI(options);
    {
        //TODO(User): Set to the region of your AWS account.
        const Aws::String region = Aws::Region::US_WEST_2;
    }
}

```

```

        //Load a profile that has been granted AmazonTranscribeFullAccess AWS
        managed permission policy.
        Aws::Client::ClientConfiguration config;
#ifdef _WIN32
        // ATTENTION: On Windows with the AWS C++ SDK, this example only runs if the
        SDK is built
        // with the curl library.
        // For more information, see the accompanying ReadMe.
        // For more information, see "Building the SDK for Windows with curl".
        // https://docs.aws.amazon.com/sdk-for-cpp/v1/developer-guide/setup-
        windows.html
        //TODO(User): Update to the location of your .crt file.
        config.caFile = "C:/curl/bin/curl-ca-bundle.crt";
#endif
        config.region = region;

        TranscribeStreamingServiceClient client(config);
        StartStreamTranscriptionHandler handler;
        handler.SetOnErrorCallback(
            [](const Aws::Client::AWSError<TranscribeStreamingServiceErrors>
&error) {
                std::cerr << "ERROR: " + error.GetMessage() << std::endl;
            });
        //SetTranscriptEventCallback called for every 'chunk' of file transcribed.
        // Partial results are returned in real time.
        handler.SetTranscriptEventCallback([](const TranscriptEvent &ev) {
            for (auto &r: ev.GetTranscript().GetResults()) {
                if (r.GetIsPartial()) {
                    std::cout << "[partial] ";
                }
                else {
                    std::cout << "[Final] ";
                }
                for (auto &alt: r.GetAlternatives()) {
                    std::cout << alt.GetTranscript() << std::endl;
                }
            }
        });

        StartStreamTranscriptionRequest request;
        request.SetMediaSampleRateHertz(SAMPLE_RATE);
        request.SetLanguageCode(LanguageCode::en_US);
        request.SetMediaEncoding(
            MediaEncoding::pcm); // wav and aiff files are PCM formats.

```

```

request.SetEventStreamHandler(handler);

auto OnStreamReady = [](AudioStream &stream) {
    Aws::FStream file(FILE_NAME, std::ios_base::in |
std::ios_base::binary);
    if (!file.is_open()) {
        std::cerr << "Failed to open " << FILE_NAME << '\n';
    }
    std::array<char, BUFFER_SIZE> buf;
    int i = 0;
    while (file) {
        file.read(&buf[0], buf.size());

        if (!file)
            std::cout << "File: only " << file.gcount() << " could be
read"
                        << std::endl;

        Aws::Vector<unsigned char> bits{buf.begin(), buf.end()};
        AudioEvent event(std::move(bits));
        if (!stream) {
            std::cerr << "Failed to create a stream" << std::endl;
            break;
        }
        //The std::basic_istream::gcount() is used to count the
characters in the given string. It returns
//the number of characters extracted by the last read()
operation.

        if (file.gcount() > 0) {
            if (!stream.WriteAudioEvent(event)) {
                std::cerr << "Failed to write an audio event" <<
std::endl;

                break;
            }
        }
        else {
            break;
        }
        std::this_thread::sleep_for(std::chrono::milliseconds(
25)); // Slow down because we are streaming from a file.
    }
    if (!stream.WriteAudioEvent(
        AudioEvent())) {

```

```

        // Per the spec, we have to send an empty event (an event
        without a payload) at the end.
        std::cerr << "Failed to send an empty frame" << std::endl;
    }
    else {
        std::cout << "Successfully sent the empty frame" << std::endl;
    }
    stream.flush();
    stream.Close();
};

    Aws::Utils::Threading::Semaphore signaling(0 /*initialCount*/, 1 /*
    *maxCount*/);
    auto OnResponseCallback = [&signaling](
        const TranscribeStreamingServiceClient * /*unused*/,
        const Model::StartStreamTranscriptionRequest & /*unused*/,
        const Model::StartStreamTranscriptionOutcome &outcome,
        const std::shared_ptr<const Aws::Client::AsyncCallerContext> & /*
    *unused*/) {

        if (!outcome.IsSuccess()) {
            std::cerr << "Transcribe streaming error "
                << outcome.GetError().GetMessage() << std::endl;
        }

        signaling.Release();
    };

    std::cout << "Starting..." << std::endl;
    client.StartStreamTranscriptionAsync(request, OnStreamReady,
    OnResponseCallback,
        nullptr /*context*/);
    signaling.WaitOne(); // Prevent the application from exiting until we're
    done.
    std::cout << "Done" << std::endl;
}

    Aws::ShutdownAPI(options);

    return 0;
}

```

- API 세부 정보는 AWS SDK for C++ API 참조의 [StartStreamTranscription](#)을 참조하세요.

# AWS SDK for C++ 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 AWS와 사용자 간의 공동 책임입니다. [공동 책임 모델](#)은 이 사항을 클라우드 내 보안 및 클라우드의 보안으로 설명합니다.

클라우드 보안 - AWS는 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 중인 AWS 서비스와 데이터의 민감도, 조직의 요구 사항 및 관련 법률 및 규정을 비롯한 기타 요인에 따라 결정됩니다.

이 AWS 제품 또는 서비스는 지원하는 특정 Amazon Web Services(AWS) 서비스를 통해 [공동 책임 모델](#)을 따릅니다. AWS 서비스 보안 정보는 [AWS 서비스 보안 설명서 페이지](#) 및 규정 [AWSAWS 준수 프로그램의 규정 준수 노력 범위에 속하는 서비스를 참조하세요](#).

## 주제

- [AWS SDK for C++의 데이터 보호](#)
- [ID 및 액세스 관리](#)
- [이 AWS 제품 또는 서비스에 대한 규정 준수 확인](#)
- [이 AWS 제품 또는 서비스에 대한 복원력](#)
- [이 AWS 제품 또는 서비스에 대한 인프라 보안](#)
- [AWS SDK for C++에서 최소 TLS 버전 적용](#)
- [Amazon S3 암호화 클라이언트 마이그레이션\(V1에서 V2로\)](#)
- [Amazon S3 암호화 클라이언트 마이그레이션\(V2에서 V3로\)](#)

## AWS SDK for C++의 데이터 보호

AWS [공동 책임 모델](#)은 AWS SDK for C++의 데이터 보호에 적용됩니다. 이 모델에서 설명하는 것처럼 AWS는 모든 AWS 클라우드를 실행하는 글로벌 인프라를 보호할 책임이 있습니다. 사용자는 인프라에서 호스팅되는 콘텐츠를 관리해야 합니다. 사용하는 AWS 서비스의 보안 구성과 관리 태스크에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 [데이터 프라이버시 FAQ](#)를

참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 [AWS 공동 책임 모델 및 GDPR](#) 블로그 게시물을 참조하세요.

데이터를 보호하려면 AWS 계정 자격 증명을 보호하고 AWS IAM Identity Center 또는 AWS Identity and Access Management(IAM)를 통해 개별 사용자 계정을 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용하세요.
- SSL/TLS를 사용하여 AWS 리소스와 통신하세요. TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- AWS CloudTrail로 API 및 사용자 활동 로깅을 설정하세요. AWS 활동 캡처에 CloudTrail 추적을 사용하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 [CloudTrail 추적 작업](#)을 참조하세요.
- AWS 암호화 솔루션을 AWS 서비스 내의 모든 기본 보안 컨트롤과 함께 사용하세요.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용하세요.
- 명령줄 인터페이스 또는 API를 통해 AWS에 액세스할 때 FIPS 140-3 검증된 암호화 모듈이 필요한 경우, FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 [Federal Information Processing Standard\(FIPS\) 140-3](#)을 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 형식 텍스트 필드에 입력하지 않는 것이 좋습니다. 여기에는 SDK for C++ 또는 콘솔 또는 API, AWS CLI 또는 AWS SDK를 사용하여 다른 AWS 서비스를 처리하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버로 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 자격 증명 정보를 URL에 포함해서는 안 됩니다.

## ID 및 액세스 관리

AWS Identity and Access Management(IAM)는 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어할 수 있도록 지원하는 AWS 서비스입니다. IAM 관리자는 어떤 사용자가 AWS 리소스를 사용할 수 있는 인증(로그인) 및 권한(권한 있음)을 받을 수 있는지 제어합니다. IAM은 추가 비용 없이 사용할 수 있는 AWS 서비스입니다.

주제

- [고객](#)

- [보안 인증을 통한 인증](#)
- [정책을 사용하여 액세스 관리](#)
- [AWS 서비스에서 IAM을 사용하는 방식](#)
- [AWS 보안 인증 및 액세스 문제 해결](#)

## 고객

AWS Identity and Access Management(IAM)을 사용하는 방법은 AWS에서 수행하는 작업에 따라 달라 집니다.

서비스 사용자 - AWS 서비스를 사용하여 작업을 수행하는 경우 필요한 보안 인증 정보와 권한을 관리 자가 제공합니다. 더 많은 AWS 기능을 사용하여 작업을 수행하게 되면 추가 권한이 필요할 수 있습 니다. 액세스 권한 관리 방법을 이해하면 관리자에게 올바른 권한을 요청하는 데 도움이 됩니다. AWS의 기능에 액세스할 수 없는 경우 [AWS 보안 인증 및 액세스 문제 해결](#) 또는 사용 중인 AWS 서비스의 사 용 설명서를 참조하세요.

서비스 관리자 - 회사에서 AWS 리소스를 책임지고 있는 담당자라면 AWS에 대한 전체 액세스 권한을 가지고 있을 것입니다. 서비스 관리자는 서비스 사용자가 액세스해야 하는 AWS 기능과 리소스를 결정 합니다. 그런 다음, IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페 이지의 정보를 검토하여 IAM의 기본 개념을 이해하세요. 회사에서 AWS와 함께 IAM을 사용하는 방법 에 대한 자세한 내용은 사용 중인 AWS 서비스의 사용 설명서를 참조하세요.

IAM 관리자 - IAM 관리자라면 AWS에 대한 액세스 권한 관리 정책 작성 방법을 자세히 알고 싶을 것입 니다. IAM에서 사용할 수 있는 AWS 보안 인증 기반 정책 예제를 보려면 사용 중인 AWS 서비스의 사 용 설명서를 참조하세요.

## 보안 인증을 통한 인증

인증은 ID 자격 증명을 사용하여 AWS에 로그인하는 방식입니다. AWS 계정 루트 사용자나 IAM 사용 자로 또는 IAM 역할을 수임하여 인증받아야 합니다.

AWS IAM Identity Center(IAM Identity Center), Single Sign-On 인증 또는 Google/Facebook 자격 증명 과 같은 자격 증명 소스의 자격 증명을 사용하여 페더레이션 자격 증명으로 로그인할 수 있습니다. 로 그인하는 방법에 대한 자세한 내용은 AWS 로그인 사용 설명서의 [AWS 계정에 로그인하는 방법](#) 섹션 을 참조하세요.

프로그래밍 방식 액세스를 위해 AWS는 요청에 암호화 방식으로 서명할 수 있는 SDK 및 CLI를 제공합 니다. 자세한 내용은 IAM 사용 설명서의 [API 요청용 AWS Signature Version 4](#) 섹션을 참조하세요.

## AWS 계정 루트 사용자

AWS 계정을 생성하는 경우에는 모든 AWS 서비스 서비스와 리소스에 대한 완전한 액세스 권한이 있는 AWS 계정 루트 사용자라는 단일 로그인 자격 증명으로 시작합니다. 일상적인 작업에 루트 사용자를 사용하지 않을 것을 강력히 권장합니다. 루트 사용자 자격 증명이 필요한 작업은 IAM 사용 설명서의 [루트 사용자 자격 증명이 필요한 작업](#) 섹션을 참조하세요.

## 페더레이션 자격 증명

가장 좋은 방법은 인간 사용자가 ID 제공업체와의 페더레이션을 사용하여 임시 보안 인증으로 AWS 서비스에 액세스하도록 하는 것입니다.

페더레이션 자격 증명은 엔터프라이즈 사용자 디렉터리, 웹 ID 제공업체 또는 Directory Service의 사용자, 자격 증명 소스의 자격 증명을 사용하여 AWS 서비스에 액세스합니다. 페더레이션 자격 증명은 임시 자격 증명을 제공하는 역할을 수임합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center를 추천합니다. 자세한 정보는 AWS IAM Identity Center 사용 설명서의 [IAM Identity Center란 무엇인가요?](#) 섹션을 참조하세요.

## IAM 사용자 및 그룹

[IAM 사용자](#)는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가진 자격 증명입니다. 장기 자격 증명에 있는 IAM 사용자 대신 임시 자격 증명을 사용하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서에서 [임시 자격 증명을 사용하여 AWS에 액세스하려면 인간 사용자가 ID 공급자와의 페더레이션을 사용하도록 요구](#)를 참조하세요.

[IAM 그룹](#)은 IAM 사용자 모음을 지정하고 대규모 사용자 집합에 대한 권한 관리를 더 쉽게 만듭니다. 자세한 내용은 IAM 사용 설명서의 [IAM 사용자 사용 사례](#) 섹션을 참조하세요.

## IAM 역할

[IAM 역할](#)은 임시 자격 증명을 제공하는 특정 권한이 있는 자격 증명입니다. [사용자에서 IAM 역할\(콘솔\)로 전환](#)하거나 AWS CLI 또는 AWS API 작업을 직접 호출하여 역할을 수임할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 [역할 수임 방법](#)을 참조하세요.

IAM 역할은 페더레이션 사용자 액세스, 임시 IAM 사용자 권한, 교차 계정 액세스, 교차 서비스 액세스 및 Amazon EC2에서 실행되는 애플리케이션에 유용합니다. 자세한 내용은 IAM 사용 설명서의 [교차 계정 리소스 액세스](#)를 참조하세요.

## 정책을 사용하여 액세스 관리

정책을 생성하고 AWS ID 또는 리소스에 연결하여 AWS에서 내 액세스를 제어합니다. 정책은 자격 증명이나 리소스와 연결될 때 해당 권한을 정의합니다. AWS는 위탁자가 요청을 보낼 때 이러한 정책을 평가합니다. 대부분의 정책은 AWS에 JSON 문서로 저장됩니다. JSON 정책 문서에 대한 자세한 내용은 IAM 사용 설명서의 [JSON 정책 개요](#) 섹션을 참조하세요.

정책을 사용하여 관리자는 어떤 위탁자가 어떤 리소스에 대해 어떤 조건에서 작업을 수행할 수 있는지 정의하여 누가 무엇을 액세스할 수 있는지 지정합니다.

기본적으로, 사용자 및 역할에는 어떠한 권한도 없습니다. IAM 관리자는 IAM 정책을 생성하고 사용자가 수임할 수 있는 역할에 이를 추가합니다. IAM 정책은 작업을 수행하기 위해 사용하는 방법과 관계없이 해당 권한을 정의합니다.

### ID 기반 정책

자격 증명 기반 정책은 자격 증명(사용자, 사용자 그룹 또는 역할)에 연결하는 JSON 권한 정책 문서입니다. 이러한 정책은 자격 증명이 수행할 수 있는 작업, 대상 리소스 및 이에 관한 조건을 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서에서 [고객 관리형 정책으로 사용자 지정 IAM 권한 정의](#)를 참조하세요.

자격 증명 기반 정책은 인라인 정책(단일 자격 증명에 직접 포함) 또는 관리형 정책(여러 자격 증명에 연결된 독립 실행형 정책)일 수 있습니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 [관리형 정책 및 인라인 정책 중에서 선택](#) 섹션을 참조하세요.

### 리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 예를 들어 IAM 역할 신뢰 정책 및 Amazon S3 버킷 정책이 있습니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 통제할 수 있습니다. 리소스 기반 정책에서 [위탁자를 지정](#)해야 합니다.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 IAM의 AWS 관리형 정책을 사용할 수 없습니다.

### 액세스 제어 목록(ACL)

액세스 제어 목록(ACL)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACLs는 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

Amazon S3, AWS WAF 및 Amazon VPC는 ACL을 지원하는 대표적인 서비스입니다. ACL에 관한 자세한 내용은 Amazon Simple Storage Service 개발자 가이드의 [액세스 제어 목록\(ACL\) 개요](#)를 참조하세요.

## 기타 정책 타입

AWS는 더 일반적인 정책 유형에 따라 부여되는 최대 권한을 설정할 수 있는 추가 정책 유형을 지원합니다.

- 권한 경계 - 자격 증명 기반 정책에서 IAM 엔터티에 부여할 수 있는 최대 권한을 설정합니다. 자세한 정보는 IAM 사용 설명서의 [IAM 엔터티의 권한 범위](#)를 참조하세요.
- 서비스 제어 정책(SCP) - AWS Organizations 내 조직 또는 조직 단위에 대한 최대 권한을 지정합니다. 자세한 내용은 AWS Organizations 사용 설명서의 [서비스 제어 정책](#)을 참조하세요.
- 리소스 제어 정책(RCP) - 계정 내 리소스에 사용할 수 있는 최대 권한을 설정합니다. 자세한 내용은 AWS Organizations 사용 설명서의 [리소스 제어 정책\(RCP\)](#)을 참조하세요.
- 세션 정책 - 역할 또는 페더레이션 사용자에게 대해 임시 세션을 생성할 때 파라미터 자격으로 전달되는 고급 정책입니다. 자세한 내용은 IAM 사용 설명서의 [세션 정책](#)을 참조하세요.

## 여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 여러 정책 유형이 관련될 때 AWS가 요청을 허용할지를 결정하는 방법을 알아보려면 IAM 사용 설명서의 [정책 평가 로직](#)을 참조하세요.

## AWS 서비스에서 IAM을 사용하는 방식

AWS 서비스에서 대부분의 IAM 기능을 사용하는 방법을 전체적으로 알아보려면 IAM 사용 설명서의 [IAM으로 작업하는 AWS 서비스](#)를 참조하세요.

특정 AWS 서비스에 IAM을 사용하는 방법을 알아보려면 관련 서비스 사용 설명서의 보안 섹션을 참조하세요.

## AWS 보안 인증 및 액세스 문제 해결

다음 정보를 사용하여 AWS 및 IAM에서 발생할 수 있는 공통적인 문제를 진단하고 수정할 수 있습니다.

### 주제

- [AWS에서 작업을 수행할 권한이 없음](#)
- [iam:PassRole을 수행하도록 인증되지 않음](#)
- [내 AWS 계정 외부의 사람이 내 AWS 리소스에 액세스할 수 있게 허용하기를 원합니다.](#)

## AWS에서 작업을 수행할 권한이 없음

작업을 수행할 권한이 없다는 오류가 수신되면, 작업을 수행할 수 있도록 정책을 업데이트해야 합니다.

다음의 예제 오류는 mateojackson IAM 사용자가 콘솔을 사용하여 가상 *my-example-widget* 리소스에 대한 세부 정보를 보려고 하지만 가상 *aws:GetWidget* 권한이 없을 때 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
aws:GetWidget on resource: my-example-widget
```

이 경우, *aws:GetWidget* 작업을 사용하여 *my-example-widget* 리소스에 액세스할 수 있도록 mateojackson 사용자 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 자격 증명을 제공한 사람입니다.

## iam:PassRole을 수행하도록 인증되지 않음

*iam:PassRole* 작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 AWS에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부 AWS 서비스에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신, 해당 서비스에 기존 역할을 전달할 수 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예 오류는 marymajor라는 IAM 사용자가 콘솔을 사용하여 AWS에서 작업을 수행하려고 하는 경우에 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 수 있는 권한을 가지고 있지 않습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

이 경우, Mary가 *iam:PassRole* 작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 보안 인증 정보를 제공하는 사람입니다.

내 AWS 계정 외부의 사람이 내 AWS 리소스에 액세스할 수 있게 허용하기를 원합니다.

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우, 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세히 알아보려면 다음을 참조하세요.

- AWS에서 이러한 기능을 지원하는지 여부를 알아보려면 [AWS 서비스에서 IAM을 사용하는 방식](#)을 참조하세요.
- 소유하고 있는 AWS 계정의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 [자신이 소유한 다른 AWS 계정의 IAM 사용자에게 대한 액세스 권한 제공](#)을 참조하세요.
- 리소스에 대한 액세스 권한을 서드 파티 AWS 계정에게 제공하는 방법을 알아보려면 IAM 사용 설명서의 [서드 파티가 소유한 AWS 계정에 대한 액세스 제공](#)을 참조하세요.
- ID 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 [외부에서 인증된 사용자에게 액세스 권한 제공\(ID 페더레이션\)](#)을 참조하세요.
- 크로스 계정 액세스에 대한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 [IAM의 크로스 계정 리소스 액세스](#)를 참조하세요.

## 이 AWS 제품 또는 서비스에 대한 규정 준수 확인

AWS 서비스가 특정 규정 준수 프로그램의 범위에 포함되는지 알아보려면 [규정 준수 프로그램 제공 범위 내 AWS 서비스](#)를 참조하고 관심 있는 규정 준수 프로그램을 선택하십시오. 일반적인 정보는 [AWS 규정 준수 프로그램](#)을 참조하세요.

AWS Artifact(을)를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 [AWS Artifact에서 보고서 다운로드](#)를 참조하세요.

AWS 서비스 사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률 및 규정에 따라 결정됩니다. AWS 서비스 사용 시 규정 준수 책임에 대한 자세한 내용은 [AWS 보안 설명서](#)를 참조하세요.

이 AWS 제품 또는 서비스는 지원하는 특정 Amazon Web Services(AWS) 서비스를 통해 [공동 책임 모델](#)을 따릅니다. AWS 서비스 보안 정보는 [AWS 서비스 보안 설명서 페이지](#) 및 [AWS 규정 준수 프로그램의 AWS 규정 준수 업무 범위에 속하는 서비스를 참조하세요](#).

## 이 AWS 제품 또는 서비스에 대한 복원력

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다.

AWS 리전에서는 물리적으로 분리되고 격리된 다수의 가용 영역을 제공하며 이러한 가용 영역은 짧은 지연 시간, 높은 처리량 및 높은 중복성을 갖춘 네트워크에 연결되어 있습니다.

가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 극복 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 리전 및 가용 영역에 대한 자세한 내용은 [AWS 글로벌 인프라](#)를 참조하세요.

이 AWS 제품 또는 서비스는 지원하는 특정 Amazon Web Services(AWS) 서비스를 통해 [공동 책임 모델](#)을 따릅니다. AWS 서비스 보안 정보는 [AWS 서비스 보안 설명서 페이지](#) 및 [AWS 규정 준수 프로그램의 AWS 규정 준수 업무 범위에 속하는 서비스를 참조하세요](#).

## 이 AWS 제품 또는 서비스에 대한 인프라 보안

이 AWS 제품 또는 서비스는 관리형 서비스를 사용하므로 AWS 글로벌 네트워크 보안으로 보호됩니다. AWS 보안 서비스와 AWS의 인프라 보호 방법에 대한 자세한 내용은 [AWS 클라우드 보안](#)을 참조하세요. 인프라 보안에 대한 모범 사례를 사용하여 AWS 환경을 설계하려면 보안 원칙 AWS Well-Architected Framework의 [인프라 보호](#)를 참조하십시오.

AWS에서 게시한 API 직접 호출을 사용하여 네트워크를 통해 이 AWS 제품 또는 서비스에 액세스합니다. 고객은 다음을 지원해야 합니다.

- Transport Layer Security(TLS) TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- DHE(Ephemeral Diffie-Hellman) 또는 ECDHE(Elliptic Curve Ephemeral Diffie-Hellman)와 같은 완전 전송 보안(PFS)이 포함된 암호 제품군 Java 7 이상의 최신 시스템은 대부분 이러한 모드를 지원합니다.

또한 요청은 액세스 키 ID 및 IAM 위탁자와 관련된 시크릿 액세스 키를 사용하여 서명해야 합니다. 또는 [AWS Security Token Service](#)(AWS STS)를 사용하여 임시 보안 토큰 인증을 생성하여 요청에 서명할 수 있습니다.

이 AWS 제품 또는 서비스는 지원하는 특정 Amazon Web Services(AWS) 서비스를 통해 [공동 책임 모델](#)을 따릅니다. AWS 서비스 보안 정보는 [AWS 서비스 보안 설명서 페이지](#) 및 [AWS 규정 준수 프로그램의 AWS 규정 준수 업무 범위에 속하는 서비스를 참조하세요](#).

## AWS SDK for C++에서 최소 TLS 버전 적용

AWS 서비스와 통신할 때 보안을 강화하려면 TLS 1.2 이상을 사용하도록 SDK for C++를 구성해야 합니다. TLS 1.3 사용을 권장합니다.

AWS SDK for C++는 교차 플랫폼 라이브러리입니다. 원하는 플랫폼에서 애플리케이션을 빌드하고 실행할 수 있습니다. 플랫폼마다 서로 다른 기본 HTTP 클라이언트에 의존할 수 있습니다.

기본적으로 macOS, Linux, Android 및 기타 비 Windows 플랫폼에서는 [libcurl](#)을 사용합니다. libcurl 버전이 7.34.0 이상인 경우 기본 HTTP 클라이언트에서 사용하는 최소 버전은 TLS 1.0입니다.

Windows의 경우 기본 라이브러리는 [WinHttp](#)입니다. Windows에서는 사용 가능한 TLS 1.0, TLS 1.1, TLS 1.2, TLS 1.3 프로토콜 중에서 실제 사용할 프로토콜을 결정합니다. [WinINet](#) 및 [IXMLHttpRequest2](#)는 Windows에서 사용할 수 있는 다른 두 가지 옵션입니다. CMake 시점과 런타임 시점에 애플리케이션이 기본 라이브러리를 대체하도록 구성할 수 있습니다. 이 두 HTTP 클라이언트의 경우에도 Windows에서 보안 프로토콜을 결정합니다.

AWS SDK for C++는 또한 기본 HTTP 클라이언트를 재정의할 수 있는 유연성도 제공합니다. 예를 들어 사용자 지정 HTTP 클라이언트 팩토리를 사용하여 libcurl을 적용하거나 원하는 HTTP 클라이언트를 사용할 수 있습니다. TLS 1.2를 최소 버전으로 사용하려면 사용 중인 HTTP 클라이언트 라이브러리를 알고 있어야 합니다.

### 모든 플랫폼에서 libcurl을 사용하여 특정 TLS 버전 적용

이 섹션에서는 AWS SDK for C++가 HTTP 프로토콜 지원을 위해 libcurl을 종속성으로 사용한다고 가정합니다. TLS 버전을 명시적으로 지정하려면 최소 libcurl 버전이 7.34.0이어야 합니다. 또한 AWS SDK for C++의 소스 코드를 수정한 다음 다시 빌드해야 할 수도 있습니다.

다음 절차에서는 이러한 작업을 수행하는 방법을 보여줍니다.

#### libcurl을 사용하여 TLS 1.2를 적용하려면

1. libcurl 설치가 버전 7.34.0 이상인지 확인합니다.
2. [GitHub](#)에서 AWS SDK for C++의 소스 코드를 다운로드합니다.
3. `aws-cpp-sdk-core/source/http/curl/CurlHttpClient.cpp` 파일을 열고 다음 코드 줄을 검색합니다.

```
#if LIBCURL_VERSION_MAJOR >= 7
#if LIBCURL_VERSION_MINOR >= 34
curl_easy_setopt(connectionHandle, CURLOPT_SSLVERSION, CURL_SSLVERSION_TLSv1);
```

```
#endif //LIBCURL_VERSION_MINOR
#endif //LIBCURL_VERSION_MAJOR
```

4. 필요한 경우 함수 호출의 마지막 파라미터를 다음과 같이 변경합니다.

```
#if LIBCURL_VERSION_MAJOR >= 7
#if LIBCURL_VERSION_MINOR >= 34
curl_easy_setopt(connectionHandle, CURLOPT_SSLVERSION, CURL_SSLVERSION_TLSv1_2);
#endif //LIBCURL_VERSION_MINOR
#endif //LIBCURL_VERSION_MAJOR
```

5. 앞서 설명한 코드 변경을 수행한 경우 <https://github.com/aws/aws-sdk-cpp#building-the-sdk>에 있는 지침에 따라 AWS SDK for C++를 빌드하고 설치합니다.
6. 애플리케이션의 서비스 클라이언트에 대해, 클라이언트 구성에서 verifySSL 옵션을 활성화합니다(해당 옵션이 아직 활성화되지 않은 경우).

## libcurl을 사용하여 TLS 1.3을 적용하려면

TLS 1.3을 적용하려면 앞 섹션의 단계를 따르되 CURL\_SSLVERSION\_TLSv1\_2 대신 CURL\_SSLVERSION\_TLSv1\_3 옵션을 설정합니다.

## Windows에서 특정 TLS 버전 적용

다음 절차에서는 WinHttp, WinInet 또는 IXMLHTTPRequest2를 사용하여 TLS 1.2 또는 TLS 1.3을 적용하는 방법을 보여줍니다.

### 사전 조건: Windows TLS 지원 결정

- <https://docs.microsoft.com/en-us/windows/win32/secauthn/protocols-in-tls-ssl-schannel-ssp>에 설명된 대로 시스템에 사용할 수 있는 TLS 프로토콜 버전 지원을 결정합니다.
- Windows 7 SP1 또는 Windows Server 2008 R2 SP1에서 실행하는 경우 <https://docs.microsoft.com/ko-kr/windows-server/security/tls/tls-registry-settings#tls-12>에 설명된 대로 레지스트리에서 TLS 1.2 지원이 활성화되어 있는지 확인해야 합니다. 이전 배포를 실행 중인 경우 운영 체제를 업그레이드해야 합니다.

### WinHttp을 사용해 TLS 1.2 또는 TLS 1.3을 적용하려면

WinHttp는 허용 가능한 보안 프로토콜을 명시적으로 설정하는 API를 제공합니다. 그러나 런타임 시에 이를 구성 가능하게 하려면 AWS SDK for C++의 소스 코드를 수정한 다음 다시 빌드해야 합니다.

1. [GitHub](#)에서 AWS SDK for C++의 소스 코드를 다운로드합니다.
2. `aws-cpp-sdk-core/source/http/windows/WinHttpSyncHttpClient.cpp` 파일을 열고 다음 코드 줄을 검색합니다.

```
#if defined(WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_3)
    DWORD flags = WINHTTP_FLAG_SECURE_PROTOCOL_TLS1 |
    WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_1 |
        WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_2 |
    WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_3;
#else
    DWORD flags = WINHTTP_FLAG_SECURE_PROTOCOL_TLS1 |
    WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_1 | WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_2;
#endif

if (!WinHttpSetOption(GetOpenHandle(), WINHTTP_OPTION_SECURE_PROTOCOLS, &flags,
    sizeof(flags)))
{
    AWS_LOGSTREAM_FATAL(GetLogTag(), "Failed setting secure crypto protocols with
    error code: " << GetLastError());
}
```

현재 빌드 시스템에 TLS 1.3이 있는 경우 `WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_3` 옵션 플래그가 정의됩니다. 자세한 내용은 Microsoft 웹 사이트에서 [WINHTTP\\_OPTION\\_SECURE\\_PROTOCOLS](#) 및 [TLS 프로토콜 버전 지원](#)을 참조하세요.

3. 다음 중 하나를 선택합니다.

- TLS 1.2를 적용하려면:

`#else` 지시문 아래에서 `flags` 변수의 값을 다음과 같이 변경합니다.

```
DWORD flags = WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_2;
```

- TLS 1.3을 적용하려면:

`#if defined(WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_3)` 지시문 아래에서 `flags` 변수의 값을 다음과 같이 변경합니다.

```
DWORD flags = WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_3;
```

4. 앞서 설명한 코드 변경을 수행한 경우 <https://github.com/aws/aws-sdk-cpp#building-the-sdk>에 있는 지침에 따라 AWS SDK for C++를 빌드하고 설치합니다.

5. 애플리케이션의 서비스 클라이언트에 대해, 클라이언트 구성에서 `verifySSL` 옵션을 활성화합니다(해당 옵션이 아직 활성화되지 않은 경우).

## WinINet 및 IXMLHTTPRequest2를 사용하여 TLS 1.2를 적용하려면

WinINet 및 IXMLHTTPRequest2 라이브러리에 대한 보안 프로토콜을 지정하는 API는 없습니다. 따라서 AWS SDK for C++에서는 운영 체제의 기본값을 사용합니다. 다음 절차에 따라 TLS 1.2 사용을 강제 적용하도록 Windows 레지스트리를 업데이트할 수 있습니다. 다만, 그 결과는 Schannel에 의존하는 모든 애플리케이션에 영향을 미치는 전역적 변경 사항임을 유의하세요.

1. 레지스트리 편집기를 열고 `Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols`로 이동합니다.
2. 해당 하위 키가 아직 없는 경우 TLS 1.0, TLS 1.1, TLS 1.2 하위 키를 생성합니다.
3. 각 하위 키 아래에 Client 하위 키와 Server 하위 키를 생성합니다.
4. 다음 키와 값을 생성합니다.

| Key name                         | Key type | Value |
|----------------------------------|----------|-------|
| -----                            | -----    | ----- |
| TLS 1.0\Client\DisabledByDefault | DWORD    | 0     |
| TLS 1.1\Client\DisabledByDefault | DWORD    | 0     |
| TLS 1.2\Client\DisabledByDefault | DWORD    | 0     |
| TLS 1.0\Client\Enabled           | DWORD    | 0     |
| TLS 1.1\Client\Enabled           | DWORD    | 0     |
| TLS 1.2\Client\Enabled           | DWORD    | 1     |

TLS 1.2\Client\Enabled 키만 1로 설정되어 있습니다. 이 키를 1로 설정하면 TLS 1.2만 유일하게 허용되는 보안 프로토콜로 적용됩니다.

## WinINet 및 IXMLHTTPRequest2를 사용하여 TLS 1.3을 적용하려면

WinINet 및 IXMLHTTPRequest2 라이브러리에 대한 보안 프로토콜을 지정하는 API는 없습니다. 따라서 AWS SDK for C++에서는 운영 체제의 기본값을 사용합니다. 다음 절차에 따라 TLS 1.3 사용을 강제 적용하도록 Windows 레지스트리를 업데이트할 수 있습니다. 다만, 그 결과는 Schannel에 의존하는 모든 애플리케이션에 영향을 미치는 전역적 변경 사항임을 유의하세요.

1. 레지스트리 편집기를 열고 `Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols`로 이동합니다.

2. 해당 하위 키가 아직 없는 경우 TLS 1.0, TLS 1.1, TLS 1.2, TLS 1.3 하위 키를 생성합니다.
3. 각 하위 키 아래에 Client 하위 키와 Server 하위 키를 생성합니다.
4. 다음 키와 값을 생성합니다.

| Key name                         | Key type | Value |
|----------------------------------|----------|-------|
| -----                            | -----    | ----- |
| TLS 1.0\Client\DisabledByDefault | DWORD    | 0     |
| TLS 1.1\Client\DisabledByDefault | DWORD    | 0     |
| TLS 1.2\Client\DisabledByDefault | DWORD    | 0     |
| TLS 1.3\Client\DisabledByDefault | DWORD    | 0     |
| TLS 1.0\Client\Enabled           | DWORD    | 0     |
| TLS 1.1\Client\Enabled           | DWORD    | 0     |
| TLS 1.2\Client\Enabled           | DWORD    | 0     |
| TLS 1.3\Client\Enabled           | DWORD    | 1     |

TLS 1.3\Client\Enabled 키만 1로 설정되어 있습니다. 이 키를 1로 설정하면 TLS 1.3만 유일하게 허용되는 보안 프로토콜로 적용됩니다.

## Amazon S3 암호화 클라이언트 마이그레이션(V1에서 V2로)

### Note

Amazon S3 암호화 클라이언트의 V2를 사용 중이고 V3로 마이그레이션하려는 경우 섹션을 참조하세요 [Amazon S3 암호화 클라이언트 마이그레이션\(V2에서 V3로\)](#).

이 주제는 애플리케이션을 Amazon Simple Storage Service(Amazon S3) 암호화 클라이언트 버전 1(V1)에서 버전 2(V2)로 마이그레이션하고 마이그레이션 프로세스 전반에 걸쳐 애플리케이션 가용성을 보장하는 방법을 설명합니다.

## 마이그레이션 개요

이 마이그레이션은 다음 두 단계로 진행됩니다.

1. 새 형식을 읽도록 기존 클라이언트를 업데이트하세요. 먼저, AWS SDK for C++ 의 업데이트된 버전을 애플리케이션에 배포합니다. 이렇게 하면 기존 V1 암호화 클라이언트가 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

먼저 기존 클라이언트를 최신 SDK 릴리스로 업데이트해야 합니다. 이 단계를 완료한 후 애플리케이션의 V1 클라이언트는 애플리케이션의 코드 기반을 업데이트하지 않고도 V2 암호화 클라이언트로 암호화된 객체를 복호화할 수 있습니다.

### 최신 버전의 빌드 및 설치 AWS SDK for C++

소스에서 SDK를 사용하는 애플리케이션

소스 AWS SDK for C++ 에서를 빌드하고 설치하는 경우 GitHub의 [aws/aws-sdk-cpp](https://github.com/aws/aws-sdk-cpp)에서 SDK 소스를 다운로드하거나 복제합니다. 그런 다음 일반 빌드 및 설치 단계를 반복합니다.

1.8.x 이전 버전 AWS SDK for C++ 에서 업그레이드하는 경우 각 메이저 버전에 도입된 주요 변경 사항은 [변경 로그](#)를 참조하세요. 를 빌드하고 설치하는 방법에 대한 자세한 내용은 단원을 AWS SDK for C++참조하십시오 [소스 코드에서 AWS SDK for C++ 가져오기](#).

Vcpkg에서 SDK를 사용하는 애플리케이션

애플리케이션에서 [Vcpkg](#)를 사용하여 SDK 업데이트를 추적하는 경우 기존 Vcpkg 업그레이드 방법을 사용하여 SDK를 최신 버전으로 업그레이드하면 됩니다. 단, 버전이 릴리스된 시점과 패키지 관리자를 통해 사용할 수 있게 되는 시점 사이에 시차가 있을 수 있습니다. [소스에서 설치](#)하면 항상 최신 버전을 사용할 수 있습니다.

다음 명령을 실행하여 aws-sdk-cpp 패키지를 업그레이드할 수 있습니다.

```
vcpkg upgrade aws-sdk-cpp
```

aws-sdk-cpp 패키지의 버전을 확인합니다.

```
vcpkg list aws-sdk-cpp
```

버전은 1.8.24 이상이어야 합니다.

에서 Vcpkg을 사용하는 방법에 대한 자세한 내용은 섹션을 AWS SDK for C++참조하십시오 [패키지 관리](#) [자에서 AWS SDK for C++ 가져오기](#).

## 애플리케이션 빌드, 설치 및 배포

애플리케이션이 정적으로 연결되어 있는 경우 애플리케이션에서 AWS SDK for C++ 코드 변경이 필요하지 않지만 최신 SDK 변경 사항을 사용하려면 애플리케이션을 다시 빌드해야 합니다. 동적 연결의 경우에는 이 단계가 필요하지 않습니다.

애플리케이션의 종속성 버전을 업그레이드하고 애플리케이션 기능을 확인한 후 플릿에 애플리케이션 배포를 진행합니다. 애플리케이션 배포가 완료되면 V2 암호화 및 복호화 클라이언트를 사용하도록 애플리케이션을 마이그레이션하는 다음 단계로 진행할 수 있습니다.

## 암호화 및 복호화 클라이언트를 V2로 마이그레이션

다음 단계는 Amazon S3 암호화 클라이언트를 V1에서 V2로 코드를 성공적으로 마이그레이션하는 방법을 보여줍니다. 코드 변경이 필요하므로 애플리케이션에 정적으로 연결하든 동적으로 연결하든 관계없이 애플리케이션을 다시 빌드해야 합니다 AWS SDK for C++.

### 새 암호화 자료 사용

V2 Amazon S3 암호화 클라이언트와 V2 암호화 구성에서는 다음 암호화 자료가 더 이상 사용되지 않습니다.

- SimpleEncryptionMaterials
- KMSEncryptionMaterials

다음 보안 암호화 자료로 대체되었습니다.

- SimpleEncryptionMaterialsWithGCMAAD
- KMSWithContextEncryptionMaterials

V2 S3 암호화 클라이언트를 구성하려면 다음과 같은 코드 변경이 필요합니다.

- S3 암호화 클라이언트를 생성할 때 **KMSEncryptionMaterials**를 사용하는 경우:
  - V2 S3 암호화 클라이언트를 생성할 때 KMSEncryptionMaterials를 KMSWithContextEncryptionMaterials로 바꾸고 V2 암호화 구성에 이를 지정합니다.
  - V2 Amazon S3 암호화 클라이언트를 사용하여 객체를 배치할 때는 CEK 암호화를 위한 KMS 컨텍스트로 문자열-문자열 컨텍스트 맵을 명시적으로 제공해야 합니다. 이 맵은 비어 있을 수도 있습니다.

- S3 암호화 클라이언트를 생성할 때 **SimpleEncryptionMaterials**를 사용하는 경우:
  - V2 Amazon S3 암호화 클라이언트를 생성할 때 SimpleEncryptionMaterials를 SimpleEncryptionMaterialsWithGCMAAD로 바꾸고 V2 암호화 구성에 이를 지정합니다.
  - V2 Amazon S3 암호화 클라이언트를 사용하여 객체를 배치할 때는 빈 문자열-문자열 컨텍스트 맵을 명시적으로 제공해야 합니다. 그렇지 않으면 SDK에서 오류를 반환합니다.

예: KMS/KMSWithContext 키 래핑 알고리즘 사용

마이그레이션 전(KMS 키 래핑)

```
auto materials = Aws::MakeShared<KMSEncryptionMaterials>("s3Encryption",
  CUSTOMER_MASTER_KEY_ID);
CryptoConfiguration cryptoConfig;
S3EncryptionClient encryptionClient(materials, cryptoConfig);
// Code snippet here to setup the putObjectRequest object.
encryptionClient.PutObject(putObjectRequest);
```

마이그레이션 후(KMSWithContext 키 래핑)

```
auto materials = Aws::MakeShared<KMSWithContextEncryptionMaterials>("s3EncryptionV2",
  CUSTOMER_MASTER_KEY_ID);
CryptoConfigurationV2 cryptoConfig(materials);
S3EncryptionClientV2 encryptionClient(cryptoConfig);
// Code snippet here to setup the putObjectRequest object.
Aws::Map<Aws::String, Aws::String> kmsContextMap;
kmsContextMap.emplace("client", "aws-sdk-cpp");
kmsContextMap.emplace("version", "1.8.0");
encryptionClient.PutObject(putObjectRequest, kmsContextMap /* could be empty as well
*/);
```

예: AES/AES-GCM 키 래핑 알고리즘 사용

마이그레이션 전(AES 키 래핑)

```
auto materials = Aws::MakeShared<SimpleEncryptionMaterials>("s3Encryption",
  HashingUtils::Base64Decode(AES_MASTER_KEY_BASE64));
CryptoConfiguration cryptoConfig;
S3EncryptionClient encryptionClient(materials, cryptoConfig);
// Code snippet here to setup the putObjectRequest object.
encryptionClient.PutObject(putObjectRequest);
```

## 마이그레이션 후(AES-GCM 키 래핑)

```
auto materials = Aws::MakeShared<SimpleEncryptionMaterialsWithGCMAAD>("s3EncryptionV2",
    HashingUtils::Base64Decode(AES_MASTER_KEY_BASE64));
CryptoConfigurationV2 cryptoConfig(materials);
S3EncryptionClientV2 encryptionClient(cryptoConfig);
// Code snippet here to setup the putObjectRequest object.
encryptionClient.PutObject(putObjectRequest, {} /* must be an empty map */);
```

## 추가 예제

다음 예제는 V1에서 V2로의 마이그레이션과 관련된 특정 사용 사례를 해결하는 방법을 보여줍니다.

### 레거시 Amazon S3 암호화 클라이언트로 암호화된 객체 복호화

기본적으로 V2 Amazon S3 암호화 클라이언트를 사용하여 더 이상 사용되지 않는 키 래핑 알고리즘 또는 더 이상 사용되지 않는 콘텐츠 암호화 스키마로 암호화된 객체를 해독할 수 없습니다.

다음 키 래핑 알고리즘은 더 이상 사용되지 않습니다.

- KMS
- AES\_KEY\_WRAP

다음 콘텐츠 암호화 스키마는 더 이상 사용되지 않습니다.

- CBC
- CTR

에서 레거시 Amazon S3 암호화 클라이언트 AWS SDK for C++ 를 사용하여 객체를 암호화하는 경우 다음과 같은 경우 더 이상 사용되지 않는 메서드를 사용할 수 있습니다.

- SimpleEncryptionMaterials 또는 KMSEncryptionMaterials를 사용한 경우.
- 암호화 구성에서 ENCRYPTION\_ONLY로 Crypto Mode를 사용한 경우

더 이상 사용되지 않는 키 래핑 알고리즘이나 더 이상 사용되지 않는 콘텐츠 암호화 스키마로 암호화된 객체를 V2 Amazon S3 암호화 클라이언트를 사용하여 복호화하려면 V2 암호화 구성에서 SecurityProfile의 기본값을 V2에서 V2\_AND\_LEGACY로 재정의해야 합니다.

## 예제

### 사전 마이그레이션

```
auto materials = Aws::MakeShared<KMSEncryptionMaterials>("s3Encryption",
    CUSTOMER_MASTER_KEY_ID);
CryptoConfiguration cryptoConfig;
S3EncryptionClient encryptionClient(materials, cryptoConfig);
// Code snippet here to setup the getObjectRequest object.
encryptionClient.GetObject(getObjectRequest);
```

### 마이그레이션 후

```
auto materials = Aws::MakeShared<KMSWithContextEncryptionMaterials>("s3EncryptionV2",
    CUSTOMER_MASTER_KEY_ID);
CryptoConfigurationV2 cryptoConfig(materials);
cryptoConfig.SetSecurityProfile(SecurityProfile::V2_AND_LEGACY);
S3EncryptionClientV2 encryptionClient(cryptoConfig);
// Code snippet here to setup the getObjectRequest object.
encryptionClient.GetObject(getObjectRequest);
```

## 범위 지정으로 객체 복호화

레거시 Amazon S3 암호화 클라이언트를 사용하면 S3 객체를 복호화할 때 수신할 바이트 범위를 지정할 수 있습니다. V2 Amazon S3 암호화 클라이언트에서 이 기능은 기본적으로 DISABLED입니다. 따라서 V2 암호화 구성에서 RangeGetMode의 기본값을 DISABLED에서 ALL로 재정의해야 합니다.

## 예제

### 사전 마이그레이션

```
auto materials = Aws::MakeShared<KMSEncryptionMaterials>("s3Encryption",
    CUSTOMER_MASTER_KEY_ID);
CryptoConfiguration cryptoConfig;
S3EncryptionClient encryptionClient(materials, cryptoConfig);
// Code snippet here to setup the getObjectRequest object.
getObjectRequest.WithRange("bytes=38-75");
encryptionClient.GetObject(getObjectRequest);
```

### 마이그레이션 후

```
auto materials = Aws::MakeShared<KMSWithContextEncryptionMaterials>("s3EncryptionV2",
    CUSTOMER_MASTER_KEY_ID);
CryptoConfigurationV2 cryptoConfig(materials);
cryptoConfig.SetUnAuthenticatedRangeGet(RangeGetMode::ALL);
S3EncryptionClientV2 encryptionClient(cryptoConfig);
// Code snippet here to setup the getObjectRequest object.
getObjectRequest.WithRange("bytes=38-75");
encryptionClient.GetObject(getObjectRequest);
```

## CMK를 사용하여 객체 복호화

KMSWithContextEncryptionMaterials로 암호화된 객체를 복호화할 때 V2 Amazon S3 암호화 클라이언트는 빈 마스터 키를 제공하여 KMS에서 적절한 CMK를 찾으도록 할 수 있습니다. 이 기능은 기본적으로 DISABLED입니다. KMS 암호화 자료에 대해 SetKMSTDecryptWithAnyCMK(true)를 호출하여 명시적으로 구성해야 합니다.

### 예제

#### 사전 마이그레이션

```
auto materials = Aws::MakeShared<KMSEncryptionMaterials>("s3Encryption", ""/* provide
    an empty KMS Master Key*/);
CryptoConfiguration cryptoConfig;
S3EncryptionClient encryptionClient(materials, cryptoConfig);
// Code snippet here to setup the getObjectRequest object.
encryptionClient.GetObject(getObjectRequest);
```

#### 마이그레이션 후

```
auto materials = Aws::MakeShared<KMSWithContextEncryptionMaterials>("s3EncryptionV2",
    ""/* provide an empty KMS Master Key*/);
materials.SetKMSTDecryptWithAnyCMK(true);
CryptoConfigurationV2 cryptoConfig(materials);
S3EncryptionClientV2 encryptionClient(cryptoConfig);
// Code snippet here to setup the getObjectRequest object.
encryptionClient.GetObject(getObjectRequest);
```

이러한 모든 마이그레이션 시나리오에 대한 전체 코드는 Github의 [Amazon S3 암호화 예제](#)를 참조하세요.

# Amazon S3 암호화 클라이언트 마이그레이션(V2에서 V3로)

## Note

Amazon S3 암호화 클라이언트의 V1을 사용하는 경우 V3로 마이그레이션하기 전에 먼저 V2로 마이그레이션해야 합니다. V3 [Amazon S3 암호화 클라이언트 마이그레이션\(V1에서 V2로\)](#)을(를) 참조하세요.

이 주제에서는 애플리케이션을 Amazon Simple Storage Service(Amazon S3) 암호화 클라이언트 버전 2(V2)에서 버전 3(V3)으로 마이그레이션하고 마이그레이션 프로세스 전반에 걸쳐 애플리케이션 가용성을 보장하는 방법을 보여줍니다. V3는 명령 파일의 데이터 키 번조를 방지하여 보안을 강화하는 ALG\_AES\_256\_GCM\_HKDF\_SHA512\_COMMIT\_KEY 알고리즘과 커밋 정책을 도입합니다.

## 마이그레이션 개요

이 마이그레이션은 다음 두 단계로 진행됩니다.

1. 새 형식을 읽도록 기존 클라이언트를 업데이트하세요. 먼저, AWS SDK for C++ 의 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.

## V3 개념 이해

Amazon S3 암호화 클라이언트 버전 3에는 데이터 키 번조에 대한 보호를 강화하는 새로운 보안 기능이 도입되었습니다. 이러한 개념을 이해하는 것은 성공적인 마이그레이션에 필수적입니다.

## 약정 정책

커밋 정책은 암호화 및 복호화 작업 중에 암호화 클라이언트가 키 커밋을 처리하는 방법을 제어합니다. V3는 다양한 마이그레이션 시나리오 및 보안 요구 사항을 지원하기 위한 세 가지 정책 옵션을 제공합니다.

## FORBID\_ENCRYPT\_ALLOW\_DECRYPT

암호화 동작: V2와 동일한 알고리즘을 사용하여 키 커밋 없이 객체를 암호화합니다.

복호화 동작: 키 커밋을 사용하거나 사용하지 않고 암호화된 객체의 복호화를 허용합니다.

보안 영향: 이 정책은 키 커밋을 적용하지 않으며 지침 파일에서 암호화된 데이터 키의 변조를 허용할 수 있습니다. 이 정책은 V2 클라이언트가 새로 암호화된 객체를 읽어야 하는 초기 마이그레이션 단계에서만 사용됩니다.

버전 호환성: 이 정책으로 암호화된 객체는 모든 V2 및 V3 구현에서 읽을 수 있습니다.

## REQUIRE\_ENCRYPT\_ALLOW\_DECRYPT(기본값)

암호화 동작: ALG\_AES\_256\_GCM\_HKDF\_SHA512\_COMMIT\_KEY 알고리즘을 사용하여 키 커밋으로 객체를 암호화합니다.

복호화 동작: 키 커밋으로 암호화된 객체와 키 커밋 없이 암호화된 객체를 모두 복호화할 수 있습니다.

보안 영향: 이 정책은 이전 객체를 읽을 때 이전 버전과의 호환성을 유지하면서 새로 암호화된 객체에 대한 강력한 보안을 제공합니다. 이는 대부분의 마이그레이션 시나리오에 권장되는 정책입니다.

버전 호환성: 이 정책으로 암호화된 객체는 V3 및 최신 V2 구현에서만 읽을 수 있습니다. V2 클라이언트는 이러한 객체를 해독할 수 없습니다. 그러나 이 정책을 사용하는 V3 클라이언트는 V2 클라이언트로 암호화된 객체를 복호화할 수 있습니다.

## REQUIRE\_ENCRYPT\_REQUIRE\_DECRYPT

암호화 동작: ALG\_AES\_256\_GCM\_HKDF\_SHA512\_COMMIT\_KEY 알고리즘을 사용하여 키 커밋으로 객체를 암호화합니다.

복호화 동작: 키 커밋으로 암호화된 객체의 복호화만 허용합니다. 키 커밋 없이 암호화된 객체를 거부합니다.

보안 영향: 이 정책은 모든 작업에 키 커밋을 적용하여 최고 수준의 보안을 제공합니다. 모든 객체가 키 커밋으로 다시 암호화되어 더 이상 레거시 V1 또는 V2 암호화 객체를 읽을 필요가 없는 경우에만 이 정책을 사용합니다.

버전 호환성: 이 정책으로 암호화된 객체는 V3 및 최신 V2 구현에서만 읽을 수 있습니다. 또한 이 정책을 사용하는 클라이언트는 V1 또는 V2 클라이언트로 암호화된 객체를 해독할 수 없습니다.

마이그레이션 고려 사항: 마이그레이션 중에 V2 클라이언트가 새 객체를 읽어야 하는 FORBID\_ENCRYPT\_ALLOW\_DECRYPT 경우로 시작한 다음 모든 클라이언트가 V3로 업그레이드 REQUIRE\_ENCRYPT\_ALLOW\_DECRYPT되면 로 이동합니다. 마지막으로 모든 레거시 객체를 다시 암호화한 후에 REQUIRE\_ENCRYPT\_REQUIRE\_DECRYPT만 고려합니다.

## ALG\_AES\_256\_GCM\_HKDF\_SHA512\_COMMIT\_KEY 알고리즘

ALG\_AES\_256\_GCM\_HKDF\_SHA512\_COMMIT\_KEY 알고리즘은 지침 파일에 저장된 암호화된 데이터 키에 대한 향상된 보안을 제공하는 V3에 도입된 새로운 암호화 알고리즘입니다.

지침 파일 영향: ALG\_AES\_256\_GCM\_HKDF\_SHA512\_COMMIT\_KEY 알고리즘은 암호화된 데이터 키를 포함한 암호화 메타데이터를 저장하는 별도의 S3 객체인 지침 파일에만 영향을 미칩니다. 객체 메타데이터에 암호화 메타데이터를 저장하는 객체(기본 스토리지 방법)는 이 알고리즘 변경의 영향을 받지 않습니다.

변조 방지: ALG\_AES\_256\_GCM\_HKDF\_SHA512\_COMMIT\_KEY 알고리즘은 암호화된 데이터 키를 암호화 컨텍스트에 암호화 방식으로 바인딩하여 데이터 키 변조를 방지합니다. 이렇게 하면 공격자가 지침 파일에서 다른 암호화된 데이터 키를 대체하지 못하여 의도하지 않은 키로 복호화될 수 있습니다.

버전 호환성: ALG\_AES\_256\_GCM\_HKDF\_SHA512\_COMMIT\_KEY 알고리즘으로 암호화된 객체는 V3 복호화 지원이 포함된 SDK의 V3 구현 및 최신 V3 V2 전환 버전에서만 복호화할 수 있습니다.

### Warning

중요: ALG\_AES\_256\_GCM\_HKDF\_SHA512\_COMMIT\_KEY 알고리즘으로 암호화를 활성화하기 전에( REQUIRE\_ENCRYPT\_ALLOW\_DECRYPT 또는 REQUIRE\_ENCRYPT\_REQUIRE\_DECRYPT 커밋 정책을 사용하여) 이러한 객체를 읽을 모든 클라이언트가 V3 복호화를 지원하는 V3 또는 최신 V3 V2 전환 버전으로 업그레이드되었는지 확인해야 합니다. 먼저 모든 리더를 업그레이드하지 않으면 새로 암호화된 객체에 대한 복호화 실패가 발생합니다.

## 새 형식을 읽도록 기존 클라이언트를 업데이트하세요

먼저 기존 클라이언트를 최신 SDK 릴리스로 업데이트해야 합니다. 이 단계를 완료한 후 애플리케이션의 V2 클라이언트는 애플리케이션의 코드 기반을 업데이트하지 않고도 V3 암호화 클라이언트로 암호화된 객체를 해독할 수 있습니다.

### 최신 버전의 빌드 및 설치 AWS SDK for C++

소스에서 SDK를 사용하는 애플리케이션

소스 AWS SDK for C++ 에서를 빌드하고 설치하는 경우 GitHub의 [aws/aws-sdk-cpp](https://github.com/aws/aws-sdk-cpp)에서 SDK 소스를 다운로드하거나 복제합니다. 그런 다음 일반 빌드 및 설치 단계를 반복합니다.

1.11.x 이전 버전 AWS SDK for C++ 에서 업그레이드하는 경우 각 메이저 버전에 도입된 주요 변경 사항은 [변경 로그](#)를 참조하세요. 를 빌드하고 설치하는 방법에 대한 자세한 내용은 단원을 AWS SDK for C++참조하십시오 [소스 코드에서 AWS SDK for C++ 가져오기](#).

Vcpkg에서 SDK를 사용하는 애플리케이션

애플리케이션에서 [Vcpkg](#)를 사용하여 SDK 업데이트를 추적하는 경우 기존 Vcpkg 업그레이드 방법을 사용하여 SDK를 최신 버전으로 업그레이드하면 됩니다. 단, 버전이 릴리스된 시점과 패키지 관리자를 통해 사용할 수 있게 되는 시점 사이에 시차가 있을 수 있습니다. [소스에서 설치](#)하면 항상 최신 버전을 사용할 수 있습니다.

다음 명령을 실행하여 aws-sdk-cpp 패키지를 업그레이드할 수 있습니다.

```
vcpkg upgrade aws-sdk-cpp
```

aws-sdk-cpp 패키지의 버전을 확인합니다.

```
vcpkg list aws-sdk-cpp
```

V3-encrypted 객체의 복호화를 지원하려면 버전이 1.11.x 이상이어야 합니다.

에서 Vcpkg을 사용하는 방법에 대한 자세한 내용은 섹션을 AWS SDK for C++참조하세요 [패키지 관리](#) [자에서 AWS SDK for C++ 가져오기](#).

## 애플리케이션 빌드, 설치 및 배포

애플리케이션이 정적으로 연결되어 있는 경우 애플리케이션에서 AWS SDK for C++코드 변경이 필요하지 않지만 최신 SDK 변경 사항을 사용하려면 애플리케이션을 다시 빌드해야 합니다. 동적 연결의 경우에는 이 단계가 필요하지 않습니다.

애플리케이션의 종속성 버전을 업그레이드하고 애플리케이션 기능을 확인한 후 플릿에 애플리케이션 배포를 진행합니다. 애플리케이션 배포가 완료되면 V3 암호화 및 복호화 클라이언트를 사용하도록 애플리케이션을 마이그레이션하는 다음 단계를 진행할 수 있습니다.

## 암호화 및 복호화 클라이언트를 V3로 마이그레이션

다음 단계에서는 Amazon S3 암호화 클라이언트의 V2에서 V3로 코드를 성공적으로 마이그레이션하는 방법을 보여줍니다. V3 코드 변경이 필요하므로 애플리케이션에 정적으로 연결하든 동적으로 연결하든 관계없이 애플리케이션을 다시 빌드해야 합니다 AWS SDK for C++.

### V3 암호화 클라이언트 사용

V3는 S3EncryptionClientV3 클래스 및를 도입CryptoConfigurationV3하여 V2에 상응하는 클래스를 대체합니다. V3의 주요 차이점은 다음과 같습니다.

- V3는 KMSWithContextEncryptionMaterials (V2와 동일)를 사용하지만에서 명시적 구성이 필요합니다CryptoConfigurationV3.
- 모든 PutObject 작업에는 암호화 컨텍스트 맵이 필요합니다(비어 있을 수 있음).
- V3는 암호화 및 복호화 동작을 제어하는 커밋 정책을 도입합니다.
- 기본적으로 V3는 ALG\_AES\_256\_GCM\_HKDF\_SHA512\_COMMIT\_KEY 알고리즘을 사용하여 키 커밋으로 암호화합니다.
- 레거시 알고리즘 복호화 구성 API가에서 `config.SetSecurityProfile(SecurityProfile::V2_AND_LEGACY);`로 변경됩니다 `config.AllowLegacy();`.

예: KMS 암호화를 사용하여 V2에서 V3로 마이그레이션

#### 마이그레이션 전(V2)

```
// Create encryption materials
auto materials = Aws::MakeShared<KMSWithContextEncryptionMaterials>("s3EncryptionV2",
    CUSTOMER_MASTER_KEY_ID);

// Create V2 crypto configuration
CryptoConfigurationV2 cryptoConfig(materials);

// Create V2 encryption client
S3EncryptionClientV2 encryptionClient(cryptoConfig);

// Put object with encryption context
Aws::Map<Aws::String, Aws::String> encryptionContext;
encryptionContext.emplace("client", "aws-sdk-cpp");
encryptionContext.emplace("version", "1.11.0");
```

```

PutObjectRequest putObjectRequest;
putObjectRequest.SetBucket(BUCKET_NAME);
putObjectRequest.SetKey(OBJECT_KEY);
// Set object body...

auto putOutcome = encryptionClient.PutObject(putObjectRequest, encryptionContext);

// Get object with encryption context
GetObjectRequest getObjectRequest;
getObjectRequest.SetBucket(BUCKET_NAME);
getObjectRequest.SetKey(OBJECT_KEY);

auto getOutcome = encryptionClient.GetObject(getObjectRequest, encryptionContext);

```

### 마이그레이션 중(이전 버전과의 호환성을 갖춘 V3)

```

// Create encryption materials
auto materials = Aws::MakeShared<KMSWithContextEncryptionMaterials>("s3EncryptionV3",
    CUSTOMER_MASTER_KEY_ID);

// Create V3 crypto configuration with materials
CryptoConfigurationV3 cryptoConfig(materials);

// Set commitment policy to maintain compatibility with V2 encrypted objects
// This allows V3 clients to decrypt objects encrypted by the V2 client
cryptoConfig.SetCommitmentPolicy(CommitmentPolicy::REQUIRE_ENCRYPT_ALLOW_DECRYPT);

// Create V3 encryption client
S3EncryptionClientV3 encryptionClient(cryptoConfig);

// Put object with encryption context
Aws::Map<Aws::String, Aws::String> encryptionContext;
encryptionContext.emplace("client", "aws-sdk-cpp");
encryptionContext.emplace("version", "1.11.0");

PutObjectRequest putObjectRequest;
putObjectRequest.SetBucket(BUCKET_NAME);
putObjectRequest.SetKey(OBJECT_KEY);
// Set object body...

auto putOutcome = encryptionClient.PutObject(putObjectRequest, encryptionContext);

```

```
// Get object with encryption context
GetObjectRequest getObjectRequest;
getObjectRequest.SetBucket(BUCKET_NAME);
getObjectRequest.SetKey(OBJECT_KEY);

auto getOutcome = encryptionClient.GetObject(getObjectRequest, encryptionContext);
```

## 마이그레이션 후(키 커밋이 포함된 V3)

```
// Create encryption materials
auto materials = Aws::MakeShared<KMSWithContextEncryptionMaterials>("s3EncryptionV3",
    CUSTOMER_MASTER_KEY_ID);

// Create V3 crypto configuration with materials
CryptoConfigurationV3 cryptoConfig(materials);

// Use the default commitment policy (REQUIRE_ENCRYPT_REQUIRE_DECRYPT)
// This encrypts with key commitment and does not decrypt V2 objects
// cryptoConfig.SetCommitmentPolicy(CommitmentPolicy::REQUIRE_ENCRYPT_ALLOW_DECRYPT);

// Create V3 encryption client
S3EncryptionClientV3 encryptionClient(cryptoConfig);

// Put object with encryption context
Aws::Map<Aws::String, Aws::String> encryptionContext;
encryptionContext.emplace("client", "aws-sdk-cpp");
encryptionContext.emplace("version", "1.11.0");

PutObjectRequest putObjectRequest;
putObjectRequest.SetBucket(BUCKET_NAME);
putObjectRequest.SetKey(OBJECT_KEY);
// Set object body...

auto putOutcome = encryptionClient.PutObject(putObjectRequest, encryptionContext);

// Get object with encryption context
GetObjectRequest getObjectRequest;
getObjectRequest.SetBucket(BUCKET_NAME);
getObjectRequest.SetKey(OBJECT_KEY);

auto getOutcome = encryptionClient.GetObject(getObjectRequest, encryptionContext);
```

## 추가 예제

이 섹션에서는 다양한 마이그레이션 시나리오 및 요구 사항을 지원하도록 V3 암호화 클라이언트 옵션을 구성하는 추가 예제를 제공합니다.

### 레거시 지원 활성화

V3 클라이언트는 REQUIRE\_ENCRYPT\_ALLOW\_DECRYPT 또는 FORBID\_ENCRYPT\_ALLOW\_DECRYPT 커밋 정책을 사용하는 경우에만 V2 클라이언트로 암호화된 객체를 복호화할 수 있습니다. 그러나 V1 클라이언트로 암호화된 객체를 복호화해야 하는 경우 AllowLegacy() 메서드를 사용하여 레거시 지원을 명시적으로 활성화해야 합니다.

레거시 지원을 사용해야 하는 경우:

- S3 암호화 클라이언트의 V1을 사용하여 암호화된 객체가 S3에 있습니다.
- 마이그레이션 프로세스 중에 V3 클라이언트를 사용하여 이러한 V1-encrypted 객체를 읽어야 합니다.
- REQUIRE\_ENCRYPT\_ALLOW\_DECRYPT 또는 FORBID\_ENCRYPT\_ALLOW\_DECRYPT 커밋 정책을 사용하고 있습니다.

#### Warning

레거시 지원은 마이그레이션 중에 일시적으로만 활성화해야 합니다. 모든 V1 객체가 V2 또는 V3로 다시 암호화되면 보안을 극대화하기 위해 레거시 지원을 비활성화합니다.

### 예: 레거시 지원 활성화

```
// Create encryption materials
auto materials = Aws::MakeShared<KMSWithContextEncryptionMaterials>("s3EncryptionV3",
    CUSTOMER_MASTER_KEY_ID);

// Create V3 crypto configuration
CryptoConfigurationV3 cryptoConfig(materials);

// Enable legacy support to read V1 encrypted objects
cryptoConfig.AllowLegacy();

// Set commitment policy (default is REQUIRE_ENCRYPT_REQUIRE_DECRYPT but we need to
    allow decryption)
```

```
cryptoConfig.SetCommitmentPolicy(CommitmentPolicy::REQUIRE_ENCRYPT_ALLOW_DECRYPT);

// Create V3 encryption client with legacy support enabled
S3EncryptionClientV3 encryptionClient(cryptoConfig);

// Now you can decrypt objects encrypted by V1, V2, and V3 clients
GetObjectRequest getObjectRequest;
getObjectRequest.SetBucket(BUCKET_NAME);
getObjectRequest.SetKey(LEGACY_OBJECT_KEY);

Aws::Map<Aws::String, Aws::String> encryptionContext;
auto getOutcome = encryptionClient.GetObject(getObjectRequest, encryptionContext);
```

## 스토리지 방법 구성

S3 암호화 클라이언트는 객체 메타데이터(기본값) 또는 별도의 명령 파일이라는 두 가지 방법으로 암호화 메타데이터를 저장할 수 있습니다. 에서 메서드를 사용하여 스토리지 `SetStorageMethod()` 메서드를 구성할 수 있습니다 `CryptoConfigurationV3`.

스토리지 방법 옵션:

### METADATA(기본값)

암호화 메타데이터는 객체의 메타데이터 헤더에 저장됩니다. 이는 모든 암호화 정보가 객체 자체와 함께 저장되므로 가장 일반적이고 편리한 방법입니다.

사용 시기: 대부분의 시나리오에서 이 방법을 사용합니다. 암호화 메타데이터가 객체와 함께 이동하므로 객체 관리를 간소화합니다.

### INSTRUCTION\_FILE

암호화 메타데이터는 접미사를 사용하여 별도의 S3 객체(지침 파일)에 저장됩니다. `instruction`.

사용 방법: 객체 메타데이터 크기가 문제가 되거나 암호화된 객체와 암호화 메타데이터를 분리해야 하는 경우 이 방법을 사용합니다. 지침 파일을 사용하려면 S3 객체(암호화된 객체와 해당 명령 파일)를 하나 대신 관리해야 합니다.

예: 스토리지 방법 구성

```
// Create encryption materials
```

```
auto materials = Aws::MakeShared<KMSWithContextEncryptionMaterials>("s3EncryptionV3",
    CUSTOMER_MASTER_KEY_ID);

// Create V3 crypto configuration
CryptoConfigurationV3 cryptoConfig(materials);

// Option 1: Use metadata storage (default, can be omitted)
cryptoConfig.SetStorageMethod(StorageMethod::METADATA);

// Option 2: Use instruction file storage
cryptoConfig.SetStorageMethod(StorageMethod::INSTRUCTION_FILE);

// Create V3 encryption client with the configured storage method
S3EncryptionClientV3 encryptionClient(cryptoConfig);

// Put object - encryption metadata will be stored according to the configured method
Aws::Map<Aws::String, Aws::String> encryptionContext;
encryptionContext.emplace("client", "aws-sdk-cpp");

PutObjectRequest putObjectRequest;
putObjectRequest.SetBucket(BUCKET_NAME);
putObjectRequest.SetKey(OBJECT_KEY);
// Set object body...

auto putOutcome = encryptionClient.PutObject(putObjectRequest, encryptionContext);

// If using INSTRUCTION_FILE, a separate object with key "OBJECT_KEY.instruction" will
// be created
```

#### Note

INSTRUCTION\_FILE 스토리지 방법을 사용할 때는 암호화된 객체를 삭제해도 명령 파일이 자동으로 삭제되지 않습니다. 두 객체를 별도로 관리해야 합니다.

# AWS SDK for C++ 개발자 안내서의 문서 기록

이 주제에서는 AWS SDK for C++ 개발자 안내서의 중요한 변경 사항을 나열합니다. 이 설명서에 대한 업데이트 알림을 받으려면 [RSS feed](#)를 구독하세요.

| 변경 사항                                         | 설명                                                                                       | 날짜           |
|-----------------------------------------------|------------------------------------------------------------------------------------------|--------------|
| <a href="#">클라이언트 구성 옵션</a>                   | 클라이언트 구성 변수 목록을 업데이트했습니다.                                                                | 2025년 9월 8일  |
| <a href="#">메모리 관리 및 목차 업데이트</a>              | 메모리 관리 파라미터를 최신 버전으로 업데이트했습니다.<br>AWS SDKs.                                              | 2025년 3월 14일 |
| <a href="#">Amazon S3 암호화 클라이언트 V3 마이그레이션</a> | Amazon S3 암호화 클라이언트의 V2에서 V3로 마이그레이션하는 방법에 대한 정보가 추가되었습니다. V3                            | 2024년 12월 2일 |
| <a href="#">사용자 지정 libcrypto</a>              | 사용자 지정 libcrypto 사용에 대한 내용을 추가했습니다.<br>CMake 최대 제한을 제거했습니다. 사용 가능한 CMake 파라미터를 업데이트했습니다. | 2024년 2월 20일 |
| <a href="#">목차</a>                            | 코드 예제에 더 쉽게 접근할 수 있도록 목차를 업데이트했습니다.                                                      | 2023년 6월 1일  |
| <a href="#">IAM 모범 사례 업데이트</a>                | IAM 모범 사례에 따라 가이드가 업데이트되었습니다. 자세한 내용은 <a href="#">IAM의 보안 모범 사례</a> 를 참조하세요.             | 2023년 3월 1일  |
| <a href="#">nuget 제거</a>                      | 사용 가능한 최신 버전이 너무 오래되어 nuget을 유효한 패키지 관리자 옵션으로 언급하는 내용을 제거했습니다.                           | 2022년 12월 2일 |

|                                          |                                                                                                  |               |
|------------------------------------------|--------------------------------------------------------------------------------------------------|---------------|
| <a href="#">시작하기에 대한 업데이트</a>            | 에서 지원하지 않으며 외부 옵션인을 명확하게 전달하도록 vcpkg 콘텐츠를 업데이트 AWS 했습니다. curl을 사용한 Windows용 SDK 빌드 지침을 업데이트했습니다. | 2022년 10월 18일 |
| <a href="#">ClientConfiguration 업데이트</a> | 최신 API를 정확히 반영하기 위해 ClientConfiguration 구조를 업데이트했습니다.                                            | 2022년 9월 22일  |
| <a href="#">시작하기 개선 사항</a>               | Windows 및 Linux에서 SDK 빌드 시작 방법에 대한 지침을 더욱 명확하게 개선했습니다.                                           | 2022년 6월 24일  |
| <a href="#">Windows curl 지원</a>          | curl을 사용하여 Windows용 SDK를 빌드하는 방법에 대한 참고 사항을 추가했습니다.                                              | 2022년 6월 15일  |
| <a href="#">EC2-Classic 사용 중지</a>        | EC2-Classic에 대한 참고 사항이 추가되었습니다.                                                                  | 2022년 4월 13일  |
| <a href="#">SDK 지표 활성화</a>               | 더 이상 사용되지 않는 SDK 지표 사용에 대한 정보를 제거했습니다.                                                           | 2022년 1월 20일  |
| <a href="#">AWS 서비스 작업</a>               | 코드 예제 리포지토리의 GitHub에서 사용할 수 있는 코드 예제 목록이 포함되었습니다.                                                | 2022년 1월 11일  |
| <a href="#">개선 사항</a>                    | 시작하기 섹션, 코드 예제 섹션 및 일반적인 표준화를 개선했습니다.                                                            | 2021년 6월 9일   |
| <a href="#">버전 업데이트</a>                  | SDK 일반 릴리스 버전을 1.9로 업데이트했습니다.                                                                    | 2021년 4월 20일  |

|                                            |                                                                   |              |
|--------------------------------------------|-------------------------------------------------------------------|--------------|
| <a href="#">사용 시작하기 AWS SDK for C++</a>    | 새로운 구성과 자세한 내용으로 섹션을 업데이트했습니다.                                    | 2021년 3월 17일 |
| <a href="#">Amazon S3 암호화 클라이언트 마이그레이션</a> | 애플리케이션을 Amazon S3 암호화 클라이언트 V1에서 V2로 마이그레이션하는 방법에 대한 정보가 추가되었습니다. | 2020년 8월 7일  |
| <a href="#">보안 내용</a>                      | 보안 콘텐츠를 추가했습니다.                                                   | 2020년 2월 6일  |
| <a href="#">버킷 생성, 나열, 삭제</a>              | Amazon S3 CreateBucket 예제를 지원하도록 업데이트했습니다 AWS 리전.                 | 2019년 6월 20일 |
| <a href="#">빌드 지침</a>                      | SDK 빌드 지침을 업데이트했습니다.                                              | 2019년 4월 16일 |
| <a href="#">비동기식 메서드</a>                   | 새로운 단원을 추가했습니다.                                                   | 2019년 4월 16일 |
| <a href="#">서비스 클라이언트 클래스</a>              | 다양한 업데이트.                                                         | 2019년 4월 5일  |
| <a href="#">Amazon S3 액세스 권한 관리</a>        | 다양한 업데이트.                                                         | 2019년 4월 3일  |
| <a href="#">빌드 및 구성 변수 업데이트</a>            | SDK 빌드 지침을 업데이트했습니다. 사용 가능한 AWS 클라이언트 구성 변수를 업데이트했습니다.            | 2019년 3월 1일  |
| <a href="#">vcpkg C++ 패키지 관리자</a>          | vcpkg C++ 패키지 관리자 설정 지침을 업데이트했습니다.                                | 2019년 1월 19일 |

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.