デベロッパーガイド

AWS SDK for C++

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

AWS SDK for C++: デベロッパーガイド

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

Amazon の商標およびトレードドレスは、Amazon のものではない製品またはサービスにも関連して、お客様に混乱を招いたり Amazon の信用を傷つけたり失わせたりするいかなる形においても使用することはできません。Amazon が所有していない他のすべての商標は、それぞれの所有者の所有物であり、Amazon と提携、接続、または後援されている場合とされていない場合があります。

Table of Contents

「AWS SDK for C++ デベロッパーガイド」とは 1
その他のドキュメントとリソース 1
SDK メジャーバージョンのメンテナンスとサポート 2
ご利用開始にあたって 3
による認証 AWS 3
コンソール認証情報の使用 3
IAM Identity Center の使用 4
詳細認証情報 6
ソースからの SDK の入手 6
Windows でのビルド 7
Linux/macOS でのビルド 12
シンプルなアプリケーションの作成 16
パッケージマネージャーからの SDK の取得 21
前提条件 7
vcpkg を使用して SDK を取得する 22
ビルドに関する問題のトラブルシューティング 24
CMake エラー: 「AWSSDK」によって提供されるパッケージ設定ファイルが見つかりませんでした 24
CMake エラー: ロードファイルが見つかりませんでした (現在の SDK バージョンは 1.8 です) 25
CMake エラー: ロードファイルが見つかりませんでした 26
ランタイムエラー: aws-*.dll が見つからなかったため処理を続行できません 26
サービスクライアントの設定 28
SDK 設定 29
外部でのクライアントの設定 30
コードでのクライアントの設定 32
宣言 32
説明 35
AWS リージョン 41
認証情報プロバイダー 41
認証情報プロバイダーチェーン 41
明示的な認証情報プロバイダー 45
アイデンティティのキャッシュ 45
CMake のパラメータ 45

CMake の一般的な変数とオプション 46
Android CMake の変数とオプション 61
ログ記録 64
HTTP 68
HttpClient と AWSClient で使用される iostream の制御 68
カスタム libcrypto の使用 69
SDK for C++ にカスタム libcrypto を組み込む方法 70
すべてのステップを Docker イメージにまとめる 72
SDK の使用 74
SDK の初期化と終了 74
AWS のサービス リクエストの実行 75
非同期プログラミング 76
非同期 SDK メソッド 76
SDK 非同期メソッドの呼び出し 76
非同期オペレーションの完了通知 78
ユーティリティモジュール 81
HTTP スタック 81
文字列ユーティリティ 81
ハッシュユーティリティ 81
JSON パーサー 81
XML パーサー 82
メモリ管理 82
メモリの割り当てと解放 82
STL および AWS 文字列とベクター 84
残された課題 85
ネイティブ SDK デベロッパー向けメモリ制御 85
エラー処理 86
AWS のサービス の呼び出し 88
コード例の開始方法 88
コード例の構造 88
Visual Studio でのコード例のビルドとデバッグ 89
ランタイムエラーのトラブルシューティングの開始方法 91
ガイド付きコード例 95
Amazon CloudWatch の例 96
Amazon DynamoDB の例 112
Amazon EC2 の例 128

Amazon S3の例 153
Amazon SQS の例 187
コードの例 204
ACM 205
アクション 205
API ゲートウェイ 235
シナリオ 236
Aurora 236
はじめに 237
基本 240
アクション 205
シナリオ 236
Auto Scaling 281
はじめに 237
基本 240
アクション 205
CloudTrail 312
アクション 205
CloudWatch 317
アクション 205
CloudWatch ログ 328
アクション 205
CodeBuild 332
アクション 205
Amazon Cognito ID プロバイダー 337
はじめに 237
アクション 205
シナリオ 236
DynamoDB 361
はじめに 237
基本 240
アクション 205
シナリオ 236
Amazon EC2 439
はじめに 237
アクション 205

EventBridge 475
アクション 205
AWS Glue 479
はじめに 237
基本 240
アクション 205
HealthImaging 518
はじめに 237
アクション 205
シナリオ 236
IAM 553
はじめに 237
基本 240
アクション 205
AWS IoT 594
はじめに 237
基本 240
アクション 205
AWS IoT data 637
アクション 205
Lambda 640
はじめに 237
基本 240
アクション 205
シナリオ 236
MediaConvert 664
アクション 205
Amazon RDS 673
はじめに 237
基本 240
アクション 205
シナリオ 236
Amazon RDS データサービス 710
シナリオ 236
Amazon Rekognition 711
はじめに 237

アクション 205
シナリオ 236
Amazon S3 717
開始方法 237
基本 240
アクション 205
シナリオ 236
Secrets Manager 801
アクション 205
Amazon SES 802
アクション 205
シナリオ 236
Amazon SNS 825
はじめに 237
アクション 205
シナリオ 236
Amazon SQS 867
はじめに 237
アクション 205
シナリオ 236
AWS STS 904
アクション 205
Amazon Transcribe ストリーミング 906
アクション 205
シナリオ 236
セキュリティ 915
データ保護 915
ID とアクセス管理 917
対象者 917
アイデンティティによる認証 918
ポリシーを使用したアクセス権の管理 919
AWS のサービスと IAM の連携の仕組み 921
AWS ID とアクセスのトラブルシューティング 921
コンプライアンス検証 923
レジリエンス 924
インフラストラクチャセキュリティ 924

最小 TLS バージョンの適用 925
libcurl を使用してすべてのプラットフォームで特定の TLS バージョンを適用する 925
Windows で特定の TLS バージョンを適用する 926
Amazon S3 暗号化クライアントの移行 (V1 から V2) 930
移行の概要 930
新しいフォーマットを読み取るために既存のクライアントを更新する 930
暗号化および復号クライアントを V2 に移行する 932
その他の例 934
Amazon S3 暗号化クライアントの移行 (V2 から V3) 936
移行の概要 937
V3 の概念について 937
新しいフォーマットを読み取るために既存のクライアントを更新する 940
暗号化クライアントと復号クライアントを V3 に移行する 941
その他の例 944
ドキュメント履歴 948
..... cmli

「AWS SDK for C++ デベロッパーガイド」とは

AWS SDK for C++ デベロッパーガイドへようこそ。

AWS SDK for C++ は、Amazon Web Services (AWS) 用の最新の C++ (C++11 以降) インターフェイスを提供します。AWS のほぼすべての機能に対して、高レベルと低レベルの両方の API を提供し、依存関係を最小限に抑えつつ、Windows、macOS、Linux、モバイル環境でのプラットフォームの移植性を実現します。

の開始方法AWS SDK for C++

> **Note**
>
> AWS IoT SDK と `aws-iot-device-sdk-cpp` はこの SDK とは別のものです。AWS IoT Device SDK for C++ v2 は GitHub の `aws-iot-device-sdk-cpp-v2` で入手できます。AWS IoT の詳細については、「AWS IoT デベロッパーガイド」の「AWS IoT とは」を参照してください。

その他のドキュメントとリソース

このガイドに加えて、以下の AWS SDK for C++ 開発者のための貴重なオンラインリソースもあります。

- 「AWS SDK とツールのリファレンスガイド」: AWS SDK に共通する設定、機能、その他の基本概念が記載されています。
- GitHub:
  - SDK ソース
  - SDK に関する問題
- AWS SDK for C++ API リファレンス
- AWS C++ Developer Blog
- AWS Code Sample Catalog
- SDK ライセンス
- ビデオ: AWS re:Invent 2015 での AWS SDK for C++ の紹介

SDK メジャーバージョンのメンテナンスとサポート

SDK メジャーバージョンのメンテナンスとサポート、およびその基礎的な依存関係については、AWS SDK とツール共有設定および認証情報リファレンスガイド で以下を参照してください。

- AWS SDK とツールのメンテナンスポリシー
- AWS SDK とツールのバージョンサポートマトリクス

の開始方法AWS SDK for C++

AWS SDK for C++ は、Amazon Web Services に接続するためのモジュール化されたクロスプラットフォームのオープンソースライブラリです。

AWS SDK for C++ は CMake を使用して、ビデオゲーム、システム、モバイル、組み込みデバイスなど、複数のドメインにわたる多様なプラットフォームをサポートしています。CMake は、アプリケーションの依存関係を管理し、ビルド対象のプラットフォームに適した Makefile を生成するビルドツールです。CMake は、プラットフォームやアプリケーションで使用されないビルド部分を除外します。

AWS のリソースにアクセスするコードを実行する前に、AWS でのコードの認証方法を設定する必要があります。

- AWS SDK for C++ AWS を使用した での認証

AWS SDK for C++ をコード内で使用するには、SDK のソースを直接ビルドするか、パッケージマネージャーで SDK の実行可能ファイルを取得します。

- ソースコードからの AWS SDK for C++ の入手
- パッケージマネージャーからの AWS SDK for C++ の取得

CMake に関するビルドの問題が発生した場合は、「AWS SDK for C++ のビルドに関する問題のトラブルシューティング」を参照してください。

AWS SDK for C++ AWS を使用した での認証

を使用して開発 AWS するときに、コードが で認証される方法を確立する必要があります AWS のサービス。 AWS リソースへのプログラムによるアクセスは、環境と利用可能な AWS アクセスに応じてさまざまな方法で設定できます。主な認証方式の選択肢と SDK 用の設定方法については、「AWS SDK とツールのリファレンスガイド」の「認証とアクセス」を参照してください。

コンソール認証情報の使用

ローカル開発では、新しいユーザーが既存の AWS マネジメントコンソールのサインイン認証情報を使用して AWS サービスにプログラムでアクセスすることをお勧めします。ブラウザベースの認証フローの後、 は CLI AWS Tools for PowerShell や AWS SDKs AWS などのローカル開発ツールで動作

する一時的な認証情報 AWSを生成します。この機能は、特に長期的なアクセスキーの管理よりもインタラクティブ認証を希望する場合、 AWS CLI 認証情報の設定と管理のプロセスを簡素化します。

この方法を選択した場合は、 CLI AWS を使用してコンソール認証情報でログインする手順に従います。詳細については、「コンソール認証情報を使用した AWS ローカル開発のログイン」を参照してください。

CLI AWS で設定すると、デフォルトの認証情報プロバイダーチェーンは CLI AWS によってキャッシュされたログイントークンを使用してリクエストを自動的に開始します。

IAM Identity Center の使用

この方法には、設定を容易に AWS CLI し、 AWS アクセスポータルに定期的にサインインするための のインストールが含まれます。

この方法を選択した場合は、「AWS SDK とツールのリファレンスガイド」の「IAM Identity Center 認証」の手順を完了します。その後、環境には次の要素が含まれている必要があります。

- アプリケーションを実行する前に AWS アクセスポータルセッションを開始 AWS CLIするために使用する 。
- SDK から参照できる設定値のセットを含む [default] プロファイルがある共有 AWSconfig ファイル。このファイルの場所を確認するには、AWS SDK とツールのリファレンスガイドの「共有ファイルの場所」を参照してください。
- 共有 config ファイルは region 設定を設定します。これにより、SDK AWS リージョン が AWS リクエストに使用するデフォルトが設定されます。このリージョンは、使用するリージョンが指定されていない SDK サービスリクエストに使用されます。
- SDK は、リクエストを AWSに送信する前に、プロファイルの SSO トークンプロバイダー設定を使用して認証情報を取得します。IAM Identity Center アクセス許可セットに接続された IAM ロールである sso_role_name値は、アプリケーションで AWS のサービス 使用されている へのアクセスを許可する必要があります。

  次のサンプル config ファイルは、SSO トークンプロバイダー設定で設定されたデフォルトプロファイルを示しています。プロファイルの sso_session 設定は、指定された sso-session セクションを参照します。sso-session セクションには、 AWS アクセスポータルセッションを開始するための設定が含まれています。

```
[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
```

IAM Identity Center 認証を使用するには、アプリケーションに追加のパッケージ (SSOや などSSOOIDC) AWS SDK for C++ は必要ありません。

AWS アクセスポータルセッションを開始する

アクセスするアプリケーションを実行する前に AWS のサービス、SDK が IAM Identity Center 認証を使用して認証情報を解決するためのアクティブな AWS アクセスポータルセッションが必要です。設定したセッションの長さによっては、アクセスが最終的に期限切れになり、SDK で認証エラーが発生します。 AWS アクセスポータルにサインインするには、 で次のコマンドを実行します AWS CLI。

```
aws sso login
```

デフォルトのプロファイルを設定している場合は、`--profile` オプションを指定してコマンドを呼び出す必要はありません。SSO トークンプロバイダー設定で名前付きプロファイルを使用している場合、コマンドは `aws sso login --profile named-profile` です。

アクティブなセッションが既にあるかどうかをテストするには、次の AWS CLI コマンドを実行します。

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

このコマンドへの応答により、共有 `config` ファイルに設定されている IAM Identity Center アカウントとアクセス許可のセットが報告されます。

> **Note**
>
> 既にアクティブな AWS アクセスポータルセッションがあり、 を実行している場合はaws sso login、認証情報を指定する必要はありません。

サインインプロセスにより、データ AWS CLI へのアクセスを許可するように求められる場合があります。 AWS CLI は SDK for Python 上に構築されているため、アクセス許可メッセージにはbotocore名前のバリエーションが含まれている可能性があります。

詳細認証情報

人間のユーザーとは、別名人的 ID と呼ばれ、人、管理者、デベロッパー、オペレーター、およびアプリケーションのコンシューマーを指します。 AWS 環境とアプリケーションにアクセスするには、ID が必要です。組織のメンバーである人間のユーザーは、ワークフォースアイデンティティとも呼ばれ、これはデベロッパーであるあなたを意味します。アクセス時に一時的な認証情報を使用します AWS。人間のユーザーの ID プロバイダーを使用して、一時的な認証情報を提供するロールを引き受けることで、 AWS アカウントへのフェデレーションアクセスを提供できます。一元的なアクセス管理を行うには、 AWS IAM アイデンティティセンター (IAM Identity Center) を使用して、アカウントへのアクセスとそれらのアカウント内のアクセス許可を管理することをお勧めします。その他の代替案については、以下を参照してください。

- ベストプラクティスの詳細については、IAM ユーザーガイドの「IAM でのセキュリティのベストプラクティス」を参照してください。
- 短期 AWS 認証情報を作成するには、IAM ユーザーガイドの「一時的なセキュリティ認証情報」を参照してください。
- 他の AWS SDK for C++ 認証情報プロバイダーの詳細については、「 SDK およびツールリファレンスガイド」の「標準化された認証情報プロバイダー」を参照してください。 AWS SDKs

ソースコードからの AWS SDK for C++ の入手

SDK をソースからビルドしてローカルにインストールすることで、独自のコードから AWS SDK for C++ を使用できます。

プロセスの概要

一般的なプロセス	詳細なプロセス
SDK ソースをビルドしてインストールする 1. CMake を使用して SDK のビルドファイルを生成します。	まずソースから SDK をビルドしてインストールします。 • Windows でのビルド

一般的なプロセス	詳細なプロセス
2. SDK をビルドします。 3. SDK をインストールします。	• Linux/macOS でのビルド
SDK を使用してアプリケーションをビルドする 1. SDK を使用した独自のコードを記述するか、サンプルアプリケーションを利用して cmake ファイルに AWSSDK パッケージを追加します。 2. CMake を使用して、アプリケーションのビルドファイルを生成します。 3. アプリケーションをビルドします。 4. アプリケーションを実行します。	その後、SDK を使用して独自のアプリケーションを開発します。 • シンプルなアプリケーションの作成

Windows での AWS SDK for C++ のビルド

AWS SDK for C++ を設定するには、ソースから直接 SDK をビルドするか、パッケージマネージャーを使用してライブラリからダウンロードできます。

SDK のソースはサービスごとに個別のパッケージに分割されています。SDK 全体のインストールには最大 1 時間かかる場合があります。プログラムで使用する特定のサービスのみをインストールすれば、インストール時間を短縮でき、ディスクのサイズも抑えられます。インストールするサービスを選択するには、プログラムが使用する各サービスのパッケージ名を把握しておく必要があります。パッケージディレクトリのリストは、GitHub の aws/aws-sdk-cpp で確認できます。パッケージ名は、そのサービスのディレクトリ名のサフィックス部分です。

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

前提条件

大規模な AWS クライアントをビルドするには、最低でも 4 GB の RAM が必要です。SDK は、t2.micro、t2.small など Amazon EC2 の小規模インスタンスタイプで、メモリ不足のためビルドに失敗する可能性があります。

AWS SDK for C++ を使用するには、次のいずれかが必要です。

- Microsoft Visual Studio 2015 以降、
- GNU Compiler Collection (GCC) 4.9 以降、または
- Clang 3.3 以降。

curl を使用した SDK for Windows のビルド

Windows では、SDK はデフォルトの HTTP クライアントとして WinHTTP を使用してビルドされます。ただし、WinHTTP 1.0 は HTTP/2 双方向ストリーミングをサポートしていません。これは、Amazon Transcribe や Amazon Lex などの一部の AWS のサービス で必要となります。そのため、SDK に curl サポートを組み込んでビルドすることが必要になる場合があります。curl の使用可能なすべてのダウンロードオプションを確認するには、「curl Releases and Downloads」を参照してください。curl サポートを組み込んで SDK をビルドする 1 つの方法は次のとおりです。

curl ライブラリサポートを組み込んで SDK をビルドするには

1. 「curl for Windows」にアクセスし、Microsoft Windows 用の curl バイナリパッケージをダウンロードします。
2. パッケージをコンピュータ上のフォルダ (例: `C:\curl`) に解凍します。
3. 「CA certificates extracted from Mozilla」にアクセスし、`cacert.pem` ファイルをダウンロードします。この Privacy Enhanced Mail (PEM) ファイルには、安全な Web サイトの信頼性を検証するために使用される有効なデジタル証明書のバンドルが含まれています。証明書は GlobalSign や Verisign などの認証局 (CA) 企業によって発行されています。
4. その `cacert.pem` ファイルを、先ほど展開した `bin` サブフォルダ (例: `C:\curl\bin`) に移動します。そのファイルの名前を `curl-ca-bundle.crt` に変更します。

また、以降の手順で Microsoft Build Engine (MSBuild) が curl の `dll` を見つけられる必要があります。そのため、curl の `bin` フォルダパスを Windows の `PATH` 環境変数 (例: `set PATH=%PATH%;C:\curl\bin`) に追加します。SDK をビルドする新しいコマンドプロンプトを開くたびに、この設定を追加する必要があります。あるいは、この設定が記憶されるように、Windows のシステム設定で環境変数をグローバルに設定することもできます。

以降の手順で SDK をソースからビルドする際には、ステップ 5 (ビルドファイルの生成) を参照して、curl を組み込むために必要なコマンド構文を確認してください。

コードを書く際には、caFile の中で コード内での AWS SDK for C++ サービスクライアントの設定 を証明書ファイルのパスに設定する必要があります。Amazon Transcribe の使用例については、GitHub の transcribe-streaming コードサンプルリポジトリで「AWS」を参照してください。

ソースからの SDK のビルド

コマンドラインツールを使用して、ソースから SDK をビルドできます。この方法では、SDK ビルドをカスタマイズできます。使用可能なオプションについては、「CMake のパラメータ」を参照してください。主なステップは 3 つです。まず、CMake を使用してファイルをビルドします。次に、MSBuild を使用して、オペレーティングシステムとビルドツールチェーンに対応した SDK バイナリをビルドします。最後に、そのバイナリを開発マシン上の適切な場所にインストールまたはコピーします。

SDK をソースからビルドするには

1. CMake (バージョン 3.13 以上) と、プラットフォームに関連するビルドツールをインストールします。cmake を PATH に追加することをお勧めします。CMake のバージョンを確認するには、コマンドプロンプトを開き、コマンド **cmake --version** を実行します。
2. コマンドプロンプトで、SDK を保存するフォルダに移動します。
3. 最新のソースコードを取得します。

   バージョン 1.11 では、git サブモジュールを使用して外部依存関係をラップしています。これには、「AWS SDK とツールのリファレンスガイド」に記載されている CRT ライブラリが含まれています。

   GitHub の aws/aws-sdk-cpp から SDK ソースをダウンロードまたはクローンします。

   - Git でのクローン: HTTPS

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

   - Git でのクローン: SSH

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

4. 生成されたビルドファイルは、SDK ソースディレクトリの外部に保存することをお勧めします。ビルドファイルを保存するための新しいディレクトリを作成し、そのフォルダに移動します。

```
mkdir sdk_build
cd sdk_build
```

5. cmake を実行してビルドファイルを生成します。cmake コマンドラインで、デバッグバージョンとリリースバージョンのどちらをビルドするかを指定します。この手順全体を通して Debug を選択し、アプリケーションコードをデバッグ設定で実行します。この手順全体を通して Release を選択し、アプリケーションコードをリリース設定で実行します。Windows では、SDK のインストール場所は通常 \Program Files (x86)\aws-cpp-sdk-all\ です。コマンド構文

   {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}

   ビルド出力を変更するその他の方法については、「CMake のパラメータ」を参照してください。

   ビルドファイルを生成するには、次のいずれかを実行します。

   - ビルドファイルを生成する (AWS のサービス の全体): SDK 全体をビルドするには、Debug または Release バージョンを指定して cmake を実行します。例:

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

   - ビルドファイルを生成する (AWS のサービス の一部): SDK の特定のサービスまたはサービスパッケージのみをビルドするには、BUILD_ONLY パラメータでサービス名をセミコロンで区切って指定して CMake を実行します。次の例では、Amazon S3 サービスパッケージのみをビルドします。

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

   - ビルドファイルを生成 (curl 使用): curl の前提条件を満たした後、curl サポートを SDK に組み込むには、追加で FORCE_CURL、CURL_INCLUDE_DIR、CURL_LIBRARY の 3 つの cmake コマンドオプションが必要です。例:

```
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」というエラーが表示された場合は、**cmake --version** を実行して CMake のバージョンを確認してください。CMake のバージョン 3.13 以上を使用する必要があります。

6. SDK のバイナリをビルドします。SDK の全体をビルドする場合、この処理には 1 時間以上かかることがあります。コマンド構文

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

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

   > **Note**
   >
   > 「The code execution cannot proceed ... dll not found. Reinstalling the program may fix this problem.」というエラーが発生した場合は、cmake コマンドを再度実行します。

7. 管理者権限でコマンドプロンプトを開き、先ほど CMAKE_PREFIX_PATH パラメータを使用して指定した場所に SDK をインストールします。コマンド構文

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

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

Windows での Android 向けのビルド

Android 向けにビルドするには、-DTARGET_ARCH=ANDROID コマンドラインに cmake を追加します。AWS SDK for C++ には、適切な環境変数 (ANDROID_NDK) を参照して必要なものを追加する CMake ツールチェーンファイルが含まれています。

Windows で SDK for Android をビルドするには、Visual Studio (2015 以降) のデベロッパーコマンドプロンプトから cmake を実行する必要があります。また、NMAKE がインストールされており、パスにコマンド **git** と **patch** が含まれている必要があります。Windows システムに Git がインストールされている場合、通常 **patch** は隣接するディレクトリ (`.../Git/usr/bin/`) にあります。これらの要件を確認したら、cmake のコマンドラインを少し変更して NMAKE を使用するようにします。

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

NMAKE は直列にビルドを行います。より高速にビルドを行うには、NMAKE の代わりに JOM をインストールし、cmake の呼び出しを次のように変更することをお勧めします。

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

アプリケーションの例については、「AWS SDK for C++ を使用した Android アプリケーションの設定」を参照してください。

Linux/macOS での AWS SDK for C++ のビルド

AWS SDK for C++ を設定するには、ソースから直接 SDK をビルドするか、パッケージマネージャーを使用してライブラリからダウンロードできます。

SDK のソースはサービスごとに個別のパッケージに分割されています。SDK 全体のインストールには最大 1 時間かかる場合があります。プログラムで使用する特定のサービスのみをインストールすれば、インストール時間を短縮でき、ディスクのサイズも抑えられます。インストールするサービスを選択するには、プログラムが使用する各サービスのパッケージ名を把握しておく必要があります。パッケージディレクトリのリストは、GitHub の aws/aws-sdk-cpp で確認できます。パッケージ名は、そのサービスのディレクトリ名のサフィックス部分です。

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

前提条件

大規模な AWS クライアントをビルドするには、最低でも 4 GB の RAM が必要です。SDK は、t2.micro、t2.small など Amazon EC2 の小規模インスタンスタイプで、メモリ不足のためビルドに失敗する可能性があります。

AWS SDK for C++ を使用するには、次のいずれかが必要です。

- GNU Compiler Collection (GCC) 4.9 以降、または
- Clang 3.3 以降。

Linux システムの追加要件

`libcurl`、`libopenssl`、`libuuid`、`zlib`、および (オプションで) Amazon Polly サポート用の `libpulse` のヘッダーファイル (`-dev` パッケージ) が必要です。パッケージはシステムのパッケージマネージャーで入手できます。

Debian/Ubuntu ベースのシステムにパッケージをインストールするには

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

Amazon Linux/Redhat/Fedora/CentOS ベースのシステムにパッケージをインストールするには

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

ソースからの SDK のビルド

vcpkg を使用せずにコマンドラインツールを使用して、SDK をソースからビルドできます。この方法では、SDK ビルドをカスタマイズできます。使用可能なオプションについては、「CMake のパラメータ」を参照してください。

SDK をソースからビルドするには

1. CMake (バージョン 3.13 以上) と、プラットフォームに関連するビルドツールをインストールします。`cmake` を `PATH` に追加することをお勧めします。CMake のバージョンを確認するには、コマンドプロンプトを開き、コマンド **`cmake --version`** を実行します。
2. コマンドプロンプトで、SDK を保存するフォルダに移動します。
3. 最新のソースコードを取得します。

   バージョン 1.11 では、git サブモジュールを使用して外部依存関係をラップしています。これには、「AWS SDK とツールのリファレンスガイド」に記載されている CRT ライブラリが含まれています。

GitHub の aws/aws-sdk-cpp から SDK ソースをダウンロードまたはクローンします。

- Git でのクローン: HTTPS

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

- Git でのクローン: SSH

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

4. 生成されたビルドファイルは、SDK ソースディレクトリの外部に保存することをお勧めします。ビルドファイルを保存するための新しいディレクトリを作成し、そのフォルダに移動します。

```
mkdir sdk_build
cd sdk_build
```

5. cmake を実行してビルドファイルを生成します。cmake コマンドラインで、デバッグバージョンとリリースバージョンのどちらをビルドするかを指定します。この手順全体を通して Debug を選択し、アプリケーションコードをデバッグ設定で実行します。この手順全体を通して Release を選択し、アプリケーションコードをリリース設定で実行します。コマンド構文

```
{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}
```

ビルド出力を変更するその他の方法については、「CMake のパラメータ」を参照してください。

> **Note**
>
> Mac の大文字と小文字を区別しないファイルシステムでビルドする場合、ビルドを実行するディレクトリで pwd コマンドの出力を確認します。pwd 出力で、/Users や Documents などのディレクトリ名に大文字と小文字が混在していることを確認します。

ビルドファイルを生成するには、次のいずれかを実行します。

- ビルドファイルを生成する (AWS のサービス の全体): SDK 全体をビルドするには、Debug または Release バージョンを指定して cmake を実行します。例:

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

- ビルドファイルを生成する (AWS のサービス の一部): SDK の特定のサービスまたはサービスパッケージのみをビルドするには、BUILD_ONLY パラメータでサービス名をセミコロンで区切って指定して CMake を実行します。次の例では、Amazon S3 サービスパッケージのみをビルドします。

```
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」というエラーが表示された場合は、**`cmake --version`** を実行して CMake のバージョンを確認してください。CMake のバージョン 3.13 以上を使用する必要があります。

6. SDK のバイナリをビルドします。SDK の全体をビルドする場合、処理に 1 時間以上かかることがあります。

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

7. SDK をインストールします。インストール先に選択した場所によっては、権限の昇格が必要になる場合があります。

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

Linux での Android 向けのビルド

Android 向けにビルドするには、`-DTARGET_ARCH=ANDROID` コマンドラインに cmake を追加します。AWS SDK for C++ には、適切な環境変数 (ANDROID_NDK) を参照して必要なものを追加する CMake ツールチェーンファイルが含まれています。アプリケーションの例については、「AWS SDK for C++ を使用した Android アプリケーションの設定」を参照してください。

AWS SDK for C++ を使用したシンプルなアプリケーションの作成

CMake は、アプリケーションの依存関係を管理し、ビルド対象のプラットフォームに適した Makefile を生成するためのビルドツールです。CMake を使用して、AWS SDK for C++ を利用するプロジェクトを作成・ビルドできます。

この例では、所有している Amazon S3 バケットをレポートします。この例を実行するために AWS アカウントに S3 バケットを所有している必要はありませんが、少なくとも 1 つある方が、処理内容をはるかに確認しやすくなります。バケットがまだない場合は、「Amazon Simple Storage Service ユーザーガイド」の「バケットの作成」を参照してください。

ステップ 1: コードを記述する

この例は、1 つのソースファイル (`hello_s3.cpp`) と 1 つの `CMakeLists.txt` ファイルを含む 1 つのフォルダで構成されています。プログラムは Amazon S3 を使用してストレージバケット情報をレポートします。このコードは GitHub の AWS コードサンプルリポジトリでも入手できます。

`CMakeLists.txt` のビルド設定ファイルでは、多くのオプションを設定できます。詳細については、CMake のウェブサイトにある「CMake チュートリアル」を参照してください。

> **Note**
>
> 詳細解説: `CMAKE_PREFIX_PATH` の設定
> デフォルトでは、AWS SDK for C++ は macOS、Linux、Android、およびその他の非 Windows プラットフォームでは `/usr/local` に、Windows では `\Program Files (x86)\aws-cpp-sdk-all` にインストールされます。
> これらの標準の場所に AWS SDK をインストールすると、CMake は必要なリソースを自動的に検出します。ただし、AWS SDK をカスタムの場所にインストールする場合は、SDK ビルドによって生成される次のリソースの場所を CMake に明示的に指定する必要があります。
>
> - `AWSSDKConfig.cmake`: プロジェクト内で AWS SDK ライブラリをどのように見つけて使用するかを CMake に指示する設定ファイル。このファイルがないと、CMake は AWS SDK のヘッダーファイルを見つけたり、AWS SDK ライブラリにリンクしたり、適切なコンパイラフラグを設定したりできません。
> - (バージョン 1.8 以前の場合) 依存関係の場所: `aws-c-event-stream`、`aws-c-common`、`aws-checksums`
>
> カスタムインストールパスを設定するには:

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

カスタムインストール用に CMAKE_PREFIX_PATH を設定しないと、CMake が CMakeLists.txt 内の find_package(AWSSDK) を処理しようとした際に、「Could not find AWSSDK」 (AWSSDK が見つかりませんでした) などのエラーでビルドが失敗します。

> **Note**
>
> 詳細解説: Windows ランタイムライブラリ
> プログラムを実行するには、実行可能ファイルのある場所に複数の DLL (aws-c-common.dll、aws-c-event-stream.dll、aws-checksums.dll、aws-cpp-sdk-core.dll) と、プログラムの構成要素に応じた特定の DLL が (この例では Amazon S3 を使用するため aws-cpp-sdk-s3 も) 必要です。CMakeLists.txt ファイル内の 2 番目の if ステートメントは、これらのライブラリをインストール場所から実行可能ファイルの場所にコピーして、この要件を満たします。AWSSDK_CPY_DYN_LIBS は、AWS SDK for C++ によって定義されたマクロであり、SDK の DLL をインストール場所からプログラムの実行可能ファイルのある場所にコピーします。これらの DLL が実行可能ファイルのある場所に存在しない場合、「file not found」 (ファイルが見つかりません) というランタイム例外が発生します。このようなエラーが発生した場合は、CMakeLists.txt ファイルのこの部分を確認して、使用している環境に応じて調整してください。

フォルダとソースファイルを作成するには

1. ソースファイルを保持するための hello_s3 ディレクトリ/プロジェクトを作成します。

   > **Note**
   >
   > Visual Studio でこの例を完了するには、[新しいプロジェクトの作成] を選択してから、[CMake プロジェクト] を選択します。プロジェクトに hello_s3 という名前を付けます。このプロジェクト名は CMakeLists.txt ファイルで使用されます。

2. そのフォルダ内に、所有している Amazon S3 バケットをレポートする次のコードを含む hello_s3.cpp ファイルを追加します。

```
#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. プロジェクト名、実行可能ファイル、ソースファイル、リンクされたライブラリを指定する CMakeLists.txt ファイルを追加します。

```
# 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})
```

ステップ 2: CMake を使用してビルドする

CMake は `CMakeLists.txt` の情報を使用して実行可能プログラムをビルドします。

使用している IDE の標準的な手順に従って、アプリケーションをビルドすることをお勧めします。

コマンドラインからアプリケーションをビルドするには

1. **cmake** がアプリケーションをビルドするためのディレクトリを作成します。

   ```
   mkdir my_project_build
   ```

2. ビルドディレクトリに移動し、プロジェクトのソースディレクトリへのパスを指定して **cmake** を実行します。

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

3. **cmake** がビルドディレクトリを生成した後、**make** (または Windows では **nmake**)、もしくは MSBUILD (`msbuild ALL_BUILD.vcxproj` または `cmake --build . --config=Debug`) を使用してアプリケーションをビルドできます。

ステップ 3: を実行する

このアプリケーションを実行すると、Amazon S3 バケットの合計数と各バケットの名前がコンソール出力で表示されます。

使用している IDE の標準的な手順に従って、アプリケーションを実行することをお勧めします。

> **Note**
>
> 必ずサインインしてください。IAM Identity Center を使用して認証する場合は、AWS CLI `aws sso login` コマンドを使用してサインインすることを忘れないでください。

コマンドラインからプログラムを実行するには

1. ビルド結果が生成された Debug ディレクトリに移動します。
2. 実行可能ファイルの名前を使用してプログラムを実行します。

```
hello_s3
```

AWS SDK for C++ を使用したその他の例については、「AWS SDK for C++ を使用した AWS のサービス の呼び出しのガイド付きコード例」を参照してください。

パッケージマネージャーからの AWS SDK for C++ の取得

> **Important**
>
> homebrew や vcpkg などのパッケージマネージャーを使用している場合:
> SDK for C++ を新しいバージョンに更新した後は、SDK に依存するライブラリや実行可能ファイルを再コンパイルする必要があります。

AWS SDK for C++ を設定するには、ソースから直接 SDK をビルドするか、パッケージマネージャーを使用してライブラリからダウンロードできます。

SDK のソースはサービスごとに個別のパッケージに分割されています。SDK 全体のインストールには最大 1 時間かかる場合があります。プログラムで使用する特定のサービスのみをインストールす

れば、インストール時間を短縮でき、ディスクのサイズも抑えられます。インストールするサービスを選択するには、プログラムが使用する各サービスのパッケージ名を把握しておく必要があります。パッケージディレクトリのリストは、GitHub の aws/aws-sdk-cpp で確認できます。パッケージ名は、そのサービスのディレクトリ名のサフィックス部分です。

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

前提条件

大規模な AWS クライアントをビルドするには、最低でも 4 GB の RAM が必要です。SDK は、t2.micro、t2.small など Amazon EC2 の小規模インスタンスタイプで、メモリ不足のためビルドに失敗する可能性があります。

Linux/macOS

AWS SDK for C++ を Linux/macOS で使用するには、次のいずれかが必要です。

- GNU Compiler Collection (GCC) 4.9 以降、または
- Clang 3.3 以降。

Windows

AWS SDK for C++ を Windows で使用するには、次のいずれかが必要です。

- Microsoft Visual Studio 2015 以降、
- GNU Compiler Collection (GCC) 4.9 以降、または
- Clang 3.3 以降。

vcpkg を使用して SDK を取得する

> **Important**
>
> 使用可能な vcpkg ディストリビューションは外部のコントリビューターによって提供されており、AWS から直接提供されているものではありません。最新バージョンはソースからインストールすることで常に使用できます。

vcpkg は外部のコントリビューターによって更新・保守されているパッケージマネージャーです。このパッケージマネージャーは AWS を通じて提供されているわけではなく、AWS SDK for C++ の最新版が反映されていない可能性があることに注意してください。AWS によってバージョンがリリースされてから、外部パッケージマネージャーで使用可能になるまでに遅延があります。最新バージョンはソースからインストールすることで常に使用できます。

システムに vcpkg をインストールする必要があります。

- vcpkg の GitHub Readme の手順に従って vcpkg をダウンロードしてブートストラップし、プロンプトが表示されたら次のオプションを使用します。
  - これらの手順の一環として、次の入力が求められます。

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

    SDK 全体をインストールするには、`.\vcpkg\vcpkg install "aws-sdk-cpp[*]" --recurse` を入力します。または、インストールする特定のサービスのみを、そのパッケージ名を角かっこ囲んで指定します (例: `.\vcpkg\vcpkg install "aws-sdk-cpp[s3, ec2]" --recurse`)。

  出力には、次のようなメッセージが表示されます。

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

- 後で CMake で使用するために、`-DCMAKE_TOOLCHAIN_FILE` コマンド全体をコピーしておきます。vcpkg の GitHub Readme には、このコマンドをどこでツールセットに使用するかも記載されています。
- また、vcpkg 経由でインストールしたビルド設定タイプも、後で必要になる場合があるため、メモしておきます。コンソール出力には、ビルド設定と SDK のバージョンが表示されます。次の出力例は、ビルド設定が「x86-windows」で、インストールされた AWS SDK for C++ のバージョンが 1.8 であることを示しています。

  ```
  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++ をインストールしたら、SDK を使用して独自のアプリケーションを開発できます。「シンプルなアプリケーションの作成」に示している例では、所有している Amazon S3 バケットが出力されます。

AWS SDK for C++ のビルドに関する問題のトラブルシューティング

ソースから AWS SDK for C++ をビルドする際、次のようにビルドに関する一般的な問題が発生する可能性があります。

トピック

- CMake エラー: 「AWSSDK」によって提供されるパッケージ設定ファイルが見つかりませんでした
- CMake エラー: ロードファイルが見つかりませんでした (現在の SDK バージョンは 1.8 です)
- CMake エラー: ロードファイルが見つかりませんでした
- ランタイムエラー: aws-*.dll が見つからなかったため処理を続行できません

CMake エラー: 「AWSSDK」によって提供されるパッケージ設定ファイルが見つかりませんでした

インストールされた SDK が見つからない場合、CMake は次のエラーを表示します。

```
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
 "AWSSDK"
1> [CMake]   provides a separate development package or SDK, be sure it has been
1> [CMake]   installed.
```

このエラーを解決するには、CMake に SDK のインストール先 (例: Windows、Linux/macOS) を指定します。CMakeLists.txt ファイルで最初に find_package() を呼び出す前に、次のコマンドを挿入します。例については、「???」を参照してください。

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

CMake エラー: ロードファイルが見つかりませんでした (現在の SDK バージョンは 1.8 です)

インストールされたライブラリが見つからない場合、CMake は次のエラーを表示します。

```
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/
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
```

このエラーを解決するには、CMake に SDK のインストール先 (例: Windows、Linux/macOS) を指定します。CMakeLists.txt ファイルで最初に find_package() を呼び出す前に、次のコマンドを挿入します。例については、「???」を参照してください。

```
#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()
```

この解決策は SDK バージョン 1.8 専用です。これより前のバージョンでは依存関係の扱いが異なります。バージョン 1.9 では、aws-sdk-cpp ライブラリと aws-c-* ライブラリの間に中間レイヤーを導入することで、これらの問題に対処しています。この新しいレイヤーは aws-crt-cpp という

名前で、SDK for C++ の git サブモジュールです。aws-crt-cpp は aws-c-* ライブラリ (aws-c-common、aws-checksums、 aws-c-event-streamなど) も独自の git サブモジュールとして含んでいます。これにより、SDK for C++ はすべての CRT ライブラリを再帰的に取得でき、ビルドプロセスが改善されます。

CMake エラー: ロードファイルが見つかりませんでした

インストールされたライブラリが見つからない場合、CMake は次のエラーを表示します。

```
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
```

このエラーを解決するには、CMake に共有ライブラリをビルドするように指定します。CMakeLists.txt ファイルで最初に find_package( ) を呼び出す前に、次のコマンドを挿入します。例については、「???」を参照してください。

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

ランタイムエラー: **aws-*.dll** が見つからなかったため処理を続行できません

必要な DLL が見つからない場合、CMake は次のようなエラーを表示します。

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

このエラーは、SDK for C++ に必要なライブラリまたは実行可能ファイルがアプリケーションの実行可能ファイルと同じフォルダにないため発生します。このエラーを解決するには、SDK のビルド出力をアプリケーションの実行可能ファイルのある場所にコピーします。エラーとなる DLL の具体的なファイル名は、使用している AWS サービスによって異なります。次のいずれかを行います。

- AWS SDK for C++ のインストール先にある /bin フォルダの内容をアプリケーションのビルドフォルダにコピーします。
- CMakeLists.txt ファイル内でマクロ AWSSDK_CPY_DYN_LIBS を使用すると、これらのファイルを自動でコピーできます。

このマクロを使用するには、AWSSDK_CPY_DYN_LIBS(SERVICE_LIST "" ${CMAKE_CURRENT_BINARY_DIR}) または AWSSDK_CPY_DYN_LIBS(SERVICE_LIST "" ${CMAKE_CURRENT_BINARY_DIR}/${CMAKE_BUILD_TYPE}) の呼び出しを CMakeLists.txt ファイルに追加します。例については、「???」を参照してください。

ビルド環境に合った正しいコピー先パスを選択してください。コマンドラインからのビルドでは出力がサブフォルダ (/Debug) に配置されることが多いですが、Visual Studio やその他の IDE ではそうとは限りません。出力された実行可能ファイルの場所を確認し、マクロが正しい場所にコピーしていることを確認してください。この種の変更を行う際は、ビルド出力ディレクトリの内容を削除して、次回のビルドをクリーンな状態から開始することをお勧めします。

AWS SDK for C++ でのサービスクライアントの設定

AWS のサービス にプログラムでアクセスするために、AWS SDK for C++ では各 AWS のサービス に対してクライアントクラスを使用します。たとえば、アプリケーションが Amazon EC2 にアクセスする必要がある場合、アプリケーションはそのサービスとインターフェイスをとる Amazon EC2 クライアントオブジェクトを作成します。次に、サービスクライアントを使用して、その AWS のサービス に対してリクエストを実行します。

AWS のサービス にリクエストを行うには、まずサービスクライアントを作成する必要があります。コードが使用する各 AWS のサービス には、それぞれ独自のライブラリと、サービスとやり取りするための専用の型があります。クライアントは、サービスが提供する各 API オペレーションに対応するメソッドをそれぞれ公開しています。

SDK の動作を設定する方法は多岐にわたりますが、最終的にはすべてサービスクライアントの動作に関係しています。どのような設定も、それに基づいて作成されたサービスクライアントが使用されるまでは効果がありません。

AWS のサービス を使用して開発する際には、AWS によりコードがどのように認証するかを確立する必要があります。使用する AWS リージョン を設定する必要があります。

AWS SDK とツールのリファレンスガイドには、AWS SDK の多くに共通する設定、機能、その他の基本概念も含まれています。

トピック

- AWS SDK for C++ での Aws::SDKOptions を使用した一般的な設定
- 外部での AWS SDK for C++ サービスクライアントの設定
- コード内での AWS SDK for C++ サービスクライアントの設定
- AWS SDK for C++ 用の AWS リージョン の設定
- AWS SDK for C++ 認証情報プロバイダーの使用
- AWS SDK for C++ のビルドに使用する CMake のパラメータ
- AWS SDK for C++ での口グ記録の設定と使用
- AWS SDK for C++ での HTTP クライアントのオーバーライド
- AWS SDK for C++ 内で HttpClient と AWSClient で使用される iostream の制御
- AWS SDK for C++ でのカスタム libcrypto ライブラリの使用

AWS SDK for C++ での **Aws::SDKOptions** を使用した一般的な設定

Aws::SDKOptions 構造体には、SDK 設定オプションが含まれています。Aws::SDKOptions は一般的な SDK 設定に、ClientConfiguration 構造体は AWS のサービス との通信の設定に焦点を当てています。

Aws::SDKOptions のインスタンスは Aws::InitAPI と Aws::ShutdownAPI のメソッドに渡されます。両方のメソッドには同じインスタンスを渡す必要があります。

次のサンプルは、使用可能なオプションの一部を示しています。

- デフォルトのロガーを使用してログ記録をオンにする

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

- デフォルトの HTTP クライアントファクトリをオーバーライドする

```
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 は std::shared_ptr ではなくクロージャ (匿名関数または Lambda 式とも呼ばれる) を受け取ります。SDK の各ファクトリ関数はこの方法で動作します。ファクトリのメモリ割り当てが行われる時点では、メモリマネージャーがまだインストールさ

れていないためです。メソッドにクロージャを渡すことで、安全なタイミングでメモリマネージャーが呼び出され、メモリ割り当てが行われます。この処理を実現する簡単な方法は、Lambda 式を使用することです。

- グローバル SIGPIPE ハンドラーを使用する

  curl と OpenSSL を使用して SDK for C++ をビルドする場合は、シグナルハンドラーを指定する必要があります。カスタムのシグナルハンドラーを使用しない場合は、`installSigPipeHandler` を `true` に設定します。

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

  `installSigPipeHandler` が `true` の場合、SDK for C++ は SIGPIPE シグナルを無視するハンドラーを使用します。SIGPIPE の詳細については、GNU オペレーティングシステムのウェブサイトで「Operation Error Signals」を参照してください。curl ハンドラーの詳細については、curl ウェブサイトで CURLOPT_NOSIGNAL の説明を参照してください。

  curl や OpenSSL の基盤ライブラリは、リモート側が接続を閉じたことを通知するために SIGPIPE シグナルを送信することがあります。これらのシグナルはアプリケーション側で処理する必要があります。curl のこの機能の詳細については、curl のウェブサイトで「libcurl thread safety」を参照してください。この動作は SDK に自動で組み込まれていません。シグナルハンドラーは各アプリケーションに対してグローバルであり、ライブラリが SDK の依存関係であるためです。

外部での AWS SDK for C++ サービスクライアントの設定

多くの設定はコードの外部で管理できます。設定が外部で管理されている場合、その設定はすべてのアプリケーションに適用されます。ほとんどの設定は、環境変数または個別の共有 AWS `config` ファイルを使用して行うことができます。共有 `config` ファイルでは「プロファイル」と呼ばれる異なる設定セットを保持して、環境やテストごとに異なる設定を提供できます。

環境変数と共有 config ファイルの設定は標準化されており、AWS SDK やツール全体で共有されて、異なる言語やアプリ間でも一貫した機能をサポートします。

SDK 設定の詳細については、「AWS SDK とツールのリファレンスガイド」を参照してください。SDK が環境変数または設定ファイルから解決できるすべての設定を確認するには、「AWS SDK とツールのリファレンスガイド」の「設定リファレンス」を参照してください。

AWS のサービス にリクエストを行うには、まずそのサービスのクライアントをインスタンス化します。タイムアウトや HTTP クライアント、再試行設定など、サービスクライアントの共通設定を行うことができます。

各サービスクライアントには AWS リージョン と認証情報プロバイダーが必要です。SDK はこれらの値を使用して、リソースの正しいリージョンにリクエストを送信し、正しい認証情報でリクエストに署名します。これらの値はコード内でプログラムで指定することも、環境から自動的にロードされるようにすることもできます。

SDK には、設定の値を見つけるために順番に確認する一連の場所 (またはソース) があります。

1. コードまたはサービスクライアント自体に設定されている明示的な設定は、他の設定よりも優先されます。
2. 環境変数
   - 環境変数の設定の詳細については、「AWS SDK とツールのリファレンスガイド」の「環境変数」を参照してください。
   - シェルの環境変数は、システム全体、ユーザー全体、特定のターミナルセッションなど、さまざまなスコープレベルで設定できることに注意してください。
3. 共有 config および credentials ファイル
   - これらのファイルの設定については、「AWS SDK とツールのリファレンスガイド」の「共有 config ファイルと credentials ファイル」を参照してください。
4. SDK ソースコード自体によって提供されるデフォルト値は、最後に使用されます。
   - Region などの一部のプロパティにはデフォルトがありません。これらのプロパティは、コード、環境設定、または共有 config ファイルのいずれかで明示的に指定する必要があります。SDK が必要な設定を解決できない場合、API リクエストは実行時に失敗する可能性があります。

> **Note**
>
> SDK が環境変数または設定ファイルから解決できるすべての設定を確認するには、「AWS SDK とツールのリファレンスガイド」の「設定リファレンス」を参照してください。

コード内での AWS SDK for C++ サービスクライアントの設定

設定がコード内で直接処理される場合、設定のスコープは、そのコードを使用するアプリケーションに限定されます。そのアプリケーション内には、すべてのサービスクライアントに対するグローバル設定、特定の AWS のサービス タイプのすべてのクライアントに対する設定、特定のサービスクライアントインスタンスに対する設定のオプションがあります。

AWS SDK for C++ には、アプリケーションで使用する AWS のサービス との対話機能を提供する AWS のサービス クライアントクラスが含まれています。SDK for C++ では、クライアントのデフォルト設定を変更できます。これは次のような処理を行う際に役立ちます。

- プロキシを使用したインターネットへの接続
- HTTP トランスポートの設定 (接続タイムアウトやリクエスト再試行など) の変更
- TCP ソケットバッファのサイズに関するヒントの指定

`ClientConfiguration` は SDK for C++ における構造体で、コード内でインスタンス化して使用できます。次のスニペットは、このクラスを使用してプロキシ経由で Amazon S3 にアクセスする方法を示しています。

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

設定変数の宣言

`ClientConfiguration` 構造体は次のメンバー変数を宣言しています。

```
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);
```

設定変数の説明

次のリストでは、クライアントの動作をカスタマイズするために使用できる `ClientConfiguration` メンバ変数について説明しています。

accountId

アカウントベースのエンドポイントルーティング用の AWS アカウント ID を指定します。形式 111122223333 を使用します。アカウントベースのエンドポイントルーティングにより、一部のサービスのリクエストパフォーマンスが向上します。

accountIdEndpointMode

アカウントベースのエンドポイントルーティングの動作を制御します。有効な値は「required」、「disabled」、または「preferred」です。デフォルト値は「preferred」です。必要に応じてアカウントベースのエンドポイントルーティングをオフにするには、「disabled」に設定します。

allowSystemProxy

HTTP クライアントがシステムプロキシ設定を検出するかどうかを制御します。デフォルト設定は false です。自動プロキシ検出を有効にするには、true に設定します。

appId

オプションのアプリケーション固有の識別子を指定します。設定すると、この値は `User-Agent` ヘッダーに `App/{appId}` という形式で追加されます。この値は、`AWS_SDK_UA_APP_ID` 環境変数または `sdk_ua_app_id` 共有設定プロファイル属性を使用して設定できます。

caPath、caFile

SSL 証明書の信頼ストアの場所を HTTP クライアントに指示します。信頼ストアの例は、OpenSSL `c_rehash` ユーティリティで準備されたディレクトリです。シンボリックリンクを使用している環境でない限り、これらの変数を設定する必要はありません。これらの変数は Windows および macOS システムでは効果がありません。

checksumConfig

チェックサムの計算と検証の設定が含まれます。`requestChecksumCalculation` および `responseChecksumValidation` が含まれ、デフォルト値は WHEN_SUPPORTED です。

configFactories

`Executor` や `RetryStrategy` などのクライアントユーティリティクラスを初期化するためのファクトリメソッドを指定します。オーバーライドされない限り、デフォルトのファクトリが使用されます。

requestTimeoutMs および connectTimeoutMs

HTTP リクエストがタイムアウトするまでの待機時間をミリ秒単位で指定します。例えば、大きなファイルを転送する際には、これらの値を増やすことを検討してください。

credentialProviderConfig

認証情報プロバイダーの設定が含まれます。この構造を使用して、SDK が AWS 認証情報を取得する方法をカスタマイズします。

disableExpectHeader

CURL HTTP クライアントにのみ適用されます。デフォルトでは、CURL は HTTP リクエストに「Expect: 100-Continue」ヘッダーを追加します。これは、ヘッダー受信直後にサーバーがエラーを返すような状況で、HTTP ペイロードが送信されるのを防ぐためです。この動作により、ラウンドトリップが節約され、ペイロードが小さくネットワークレイテンシーが影響する場合に有効です。この変数のデフォルト設定は false です。true に設定すると、CURL は HTTP リクエストヘッダーと本文ペイロードを一括して送信するようになります。

disableIMDS

インスタンスメタデータサービス (IMDS) 呼び出しを無効にするかどうかを制御します。デフォルト設定は false です。EC2 インスタンスの外部で実行する際に IMDS 呼び出しを無効にするには、true に設定します。

disableImdsV1

IMDSv2 を許可しつつ、IMDSv1 呼び出しを無効にするかどうかを制御します。デフォルト設定は false です。セキュリティを強化するために IMDSv1 呼び出しのみを無効にするには、true に設定します。

enableClockSkewAdjustment

HTTP 試行後に毎回クロックスキューを調整するかどうかを制御します。デフォルト設定は false です。

enableEndpointDiscovery

エンドポイント検出を使用するかどうかを制御します。デフォルトでは、リージョンエンドポイントまたはオーバーライドされたエンドポイントが使用されます。エンドポイント検出を有効にするには、変数を true に設定します。

enableHostPrefixInjection

HTTP ホストが DiscoverInstances リクエストに「data-」プレフィックスを追加するかどうかを制御します。デフォルトでは、この動作は有効です。無効にするには、変数を false に設定します。

enableHttpClientTrace

HTTP クライアントトレースをデバッグ目的で有効にするかどうかを制御します。デフォルト設定は false です。詳細な HTTP リクエストおよびレスポンスのログ記録を有効にするには、true に設定します。

enableTcpKeepAlive

TCP キープアライブパケットを送信するかどうかを制御します。デフォルト設定は true です。`tcpKeepAliveIntervalMs` 変数と組み合わせて使用します。この変数は、WinINet および IXMLHTTPRequest2 クライアントには適用されません。

endpointOverride

サービスとの通信時に使用するオーバーライド用の HTTP エンドポイントを指定します。

executor

非同期エグゼキューターハンドラーの実装を参照します。デフォルトの動作では、非同期呼び出しごとにスレッドを作成してデタッチします。この動作を変更するには、`Executor` クラスのサブクラスを実装し、そのインスタンスをこの変数に割り当てます。

followRedirects

HTTP 300 番台のリダイレクトコードを処理する際の動作を制御します。

httpLibOverride

デフォルトの HTTP ファクトリから返される HTTP 実装を指定します。Windows のデフォルトの HTTP クライアントは WinHTTP です。他のすべてのプラットフォームのデフォルトの HTTP クライアントは CURL です。

httpLibPerfMode

HTTP ライブラリのパフォーマンスモードを指定します。デフォルト設定は LOW_LATENCY です。さまざまなパフォーマンス特性に合わせた最適化のために、この設定を調整できます。

httpRequestTimeoutMs

HTTP リクエストのタイムアウトをミリ秒単位で指定します。デフォルト値は 0 (タイムアウトなし) です。大きなファイルを転送する際は、この値を増やすことを検討してください。

lowSpeedLimit

許可される最小転送速度をバイト/秒単位で指定します。転送速度が指定された速度を下回ると、転送オペレーションは中止されます。デフォルト設定は 1 バイト/秒です。この変数は CURL クライアントにのみ適用されます。

maxConnections

1 つのサーバーへの HTTP 接続の最大数を指定します。デフォルト値は 25 です。許容される最大値は明示されていませんが、利用している帯域幅で無理なく対応できる範囲にとどめる必要があります。

nonProxyHosts

プロキシ設定をバイパスするホスト名の配列を指定します。この設定を使用して、プロキシ設定から特定のホストを除外します。

profileName

設定に使用する AWS プロファイル名を指定します。SDK は、AWS 設定ファイルの指定されたプロファイルから設定をロードします。

proxyCaFile

プロキシ接続に使用する認証局ファイルへのパスが、デフォルトと異なる場合に、そのパスを指定します。

proxyCaPath

プロキシ接続時に使用する認証局トラストストアへのパスが、デフォルトと異なる場合に、そのパスを指定します。

proxyScheme、proxyHost、proxyPort、proxyUserName、proxyPassword

AWS とのすべての通信用のプロキシを設定するために使用されます。Burp スイートと組み合わせてデバッグする場合や、インターネット接続にプロキシを使用する場合などに便利です。

proxySSLCertPath

クライアント証明書が必要なプロキシ接続に使用する SSL 証明書ファイルへのパスを指定します。

proxySSLCertType

プロキシ接続に使用する SSL 証明書のタイプを指定します。一般的なタイプには PEM と DER があります。

proxySSLKeyPassword

プロキシ接続で使用する SSL プライベートキーがパスワードで保護されている場合に、そのパスワードを指定します。

proxySSLKeyPath

プロキシ接続でクライアント証明書が必要とされる場合に、SSL プライベートキーファイルへのパスを指定します。

proxySSLKeyType

プロキシ接続で使用する SSL プライベートキーのタイプを指定します。一般的なタイプには PEM と DER があります。

writeRateLimiter および readRateLimiter

トランスポートレイヤーで使用される帯域幅をスロットリングするための読み取りおよび書き込みレートリミッターの実装を参照します。デフォルトでは、読み取りおよび書き込みレートはスロットリングされません。スロットリングを導入するには、`RateLimiterInterface` のサブクラスを実装し、そのインスタンスをこれらの変数に割り当てます。

region

使用する AWS リージョン (us-east-1 など) を指定します。デフォルトでは、該当する AWS 認証情報で設定された既定リージョンが使用されます。

requestCompressionConfig

リクエスト圧縮の設定が含まれます。この構造を使用して、送信前にリクエストをいつどのように圧縮するかを制御します。

retryStrategy

再試行戦略の実装を参照します。デフォルトの戦略では、エクスポネンシャルバックオフポリシーを実装します。別の戦略を適用するには、`RetryStrategy` クラスのサブクラスを実装し、そのインスタンスをこの変数に割り当てます。

scheme

URI アドレス指定スキーム (HTTP または HTTPS) を指定します。デフォルトのスキームは HTTPS です。

tcpKeepAliveIntervalMs

TCP 接続でキープアライブパケットを送信する時間間隔をミリ秒単位で指定します。デフォルトの間隔は 30 秒です。最小設定は 15 秒です。この変数は、WinINet および IXMLHTTPRequest2 クライアントには適用されません。

telemetryProvider

メトリクスやトレースデータを収集するためのテレメトリプロバイダーの実装を参照します。オブザーバビリティ機能を有効にするには、この設定を行います。

userAgent

内部使用のみ。この変数の設定は変更しないでください。

useDualStack

IPv4 および IPv6 のデュアルスタックエンドポイントを使用するかどうかを制御します。すべての AWS サービスがすべてのリージョンで IPv6 をサポートしているわけではないことに注意してください。

useFIPS

連邦情報処理標準 (FIPS) 140-2 検証済みの暗号化モジュールを使用するかどうかを制御します。デフォルト設定は false です。FIPS 準拠が必要な場合は true に設定します。

verifySSL

SSL 証明書を検証するかどうかを制御します。デフォルトでは、SSL 証明書は検証されます。検証を無効にするには、変数を false に設定します。

version

リクエストに使用する HTTP バージョンを指定します。デフォルト設定は HTTP_VERSION_2TLS (TLS 経由の HTTP/2) です。

winHTTPOptions

Windows 固有の HTTP 設定オプションが含まれます。useAnonymousAuth (デフォルト設定は false) などが含まれます。

AWS SDK for C++ 用の AWS リージョン の設定

AWS リージョン を使用して、特定の地理的エリアで動作する AWS のサービス にアクセスできます。これは、冗長性を確保するためや、ユーザーがアクセスする場所の近くでのデータとアプリケーションの実行を維持するために有効です。

> **Important**
>
> ほとんどのリソースは特定の AWS リージョン に属しており、SDK 使用時には正しいリージョンを指定する必要があります。

共有 AWS `config` ファイルまたは環境変数でデフォルトリージョンを設定する例については、「AWS SDK とツールのリファレンスガイド」の「AWS リージョン」を参照してください。

AWS リージョン リクエストに使用するために、AWS SDK for C++ のデフォルト AWS を設定する必要があります。このデフォルトは、リージョンが指定されていないすべての SDK サービスメソッドの呼び出しに使用されます。SDK for C++ では、コードでのクライアントの設定 を使用してデフォルトのリージョンを設定することもできます。

AWS SDK for C++ 認証情報プロバイダーの使用

へのすべてのリクエストは、 によって発行された認証情報を使用して暗号で署名 AWS する必要があります AWS。実行時、SDK は複数の場所を確認して認証情報の設定値を取得します。

を使用した認証は AWS 、コードベース外で処理できます。多くの認証方法は、認証情報プロバイダーチェーンを使用して、SDK によって自動的に検出、使用、更新されます。

プロジェクトの AWS 認証を開始するためのガイド付きオプションについては、 SDK およびツールリファレンスガイドの「認証とアクセス」を参照してください。 AWS SDKs

認証情報プロバイダーチェーン

クライアントのコンストラクト時に認証情報プロバイダーを明示的に指定しない場合、SDK for C++ は認証情報プロバイダーチェーンを使用して、認証情報を提供できる一連の場所を確認します。SDK がこれらの場所のいずれかで認証情報を見つけると、検索は停止します。

認証情報の取得順序

すべての SDK には、 AWS のサービスに対するリクエストに使用する有効な認証情報を取得するためにチェックする一連の場所 (またはソース) があります。有効な認証情報が見つかると、検索は停止されます。この体系的な検索は、認証情報プロバイダーチェーンと呼ばれます。

チェーンのステップごとに、値を設定するさまざまな方法があります。コード内で直接値を設定することが常に優先され、次に環境変数として を設定し、共有ファイルで を設定します AWS config。詳細については、『AWS SDK とツールのリファレンスガイド』の「設定の優先順位」を参照してください。

SDK は、共有 AWS config ファイルと credentials ファイル内の[default]プロファイルから認証情報をロードしようとします。AWS_PROFILE 環境変数を使用することで、SDK がロードする名前付きプロファイルを選択できます。これにより、[default] の代わりに任意のプロファイルを使用できます。config および credentialsファイルは、 AWS SDKsとツールによって共有されます。AWS SDKsおよびツールリファレンスガイドには、すべての SDK および で使用される AWS SDKs設定に関する情報が記載されています AWS CLI。共有 AWS configファイルを使用して SDK を設定する方法の詳細については、「共有設定ファイルと認証情報ファイル」を参照してください。環境変数を設定して SDK を設定する方法の詳細については、「環境変数のサポート」を参照してください。

で認証するために AWS、 SDK for C++ は認証情報プロバイダーを次の順序でチェックします。

1. AWS アクセスキー (一時的および長期的な認証情報)

   SDK は、 AWS_ACCESS_KEY_IDと AWS_SECRET_ACCESS_KEY、AWS_SESSION_TOKEN環境変数、または共有 AWS credentialsファイルから認証情報をロードしようとします。

   - このプロバイダーの設定に関するガイダンスについては、「AWS SDK とツールのリファレンスガイド」の「 AWS アクセスキー」を参照してください。
   - このプロバイダーの SDK 設定プロパティの詳細については、「AWS SDK とツールのリファレンスガイド」の「 AWS アクセスキー」を参照してください。

2. AWS STS ウェブ ID

   アクセスを必要とするモバイルアプリケーションまたはクライアントベースのウェブアプリケーションを作成すると AWS、 AWS Security Token Service (AWS STS) は、パブリック ID プロバイダー (IdP) を介して認証されたフェデレーティッドユーザーの一時的なセキュリティ認証情報のセットを返します。

- プロファイルでこれを指定すると、SDK またはツールは `AssumeRoleWithWebIdentity` API メソッドを使用して AWS STS 一時的な認証情報の取得を試みます。この方法の詳細については、「AWS Security Token Service API リファレンス」の「AssumeRoleWithWebIdentity」を参照してください。
- このプロバイダーの設定に関するガイダンスについては、「AWS SDK とツールのリファレンスガイド」の「ウェブアイデンティティまたは OpenID Connect とのフェデレーション」を参照してください。
- このプロバイダーの SDK 設定プロパティの詳細については、「AWS SDK とツールのリファレンスガイド」の「ロール認証情報プロバイダーを引き受ける」を参照してください。

3. AWS IAM アイデンティティセンター

   IAM Identity Center を使用して認証する場合、SDK for C++ は AWS CLI コマンド を実行してセットアップされたシングルサインオントークンを使用します`aws sso login`。SDK は、IAM Identity Center が有効なトークンと交換した一時的な認証情報を使用します。次に、SDK は AWS のサービスを呼び出すときにこの一時的な認証情報を使用します。このプロセスの詳細については、「AWS SDK とツールのリファレンスガイド」の「AWS のサービスの SDK 認証情報解決について理解する」を参照してください。

   - このプロバイダーの設定に関するガイダンスについては、「AWS SDK とツールのリファレンスガイド」の「IAM Identity Center 認証」を参照してください。
   - このプロバイダーの SDK 設定プロパティの詳細については、「AWS SDK とツールのリファレンスガイド」の「IAM Identity Center 認証情報プロバイダー」を参照してください。

4. Signin を使用したログイン認証情報 ID AWS リゾルバー

   AWS サインインとコンソール認証情報を使用して認証する場合、SDK for C++ は CLI `aws login --profile`で `aws login`または を実行してセットアップされたコンソール認証情報を使用します。SDK は、 AWS サービスを呼び出すときにこれらの認証情報を使用します。

   - このプロセスの詳細については、 SDK およびツールリファレンスガイドの「コンソール認証情報を使用した AWS ローカル開発用ログイン」を参照してください。 AWS SDKs

5. 外部プロセスプロバイダー

   このプロバイダーは、オンプレミスの認証情報ストアからの認証情報の取得やオンプレミスの ID プロバイダーとの統合など、カスタム実装を提供するために使用できます。

   - このプロバイダーの設定方法の一例については、「AWS SDK とツールのリファレンスガイド」の「IAM Roles Anywhere」を参照してください。

- このプロバイダーの SDK 設定プロパティの詳細については、「AWS SDK とツールのリファレンスガイド」の「プロセス認証情報プロバイダー」を参照してください。

6. Amazon ECS および Amazon EKS コンテナ認証情報

   Amazon Elastic Container Service のタスクと Kubernetes サービスアカウントには IAM ロールを関連付けることができます。IAM ロールで付与されたアクセス許可は、タスクで実行されているコンテナまたはポッドのコンテナによって引き受けられます。このロールにより、(コンテナ上の) SDK for C++ アプリケーションコードが他の AWS のサービスを使用できるようになります。

   SDK は、Amazon ECS と Amazon EKS によって自動的に設定される `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` または `AWS_CONTAINER_CREDENTIALS_FULL_URI` の環境変数から認証情報を取得しようとします。

   - Amazon ECS 用にこのロールを設定する方法の詳細については、「Amazon Elastic Container Service デベロッパーガイド」の「Amazon ECS タスクの IAM ロール」を参照してください。
   - Amazon EKS の設定情報については、「Amazon EKS ユーザーガイド」の「Amazon EKS Pod Identity Agent の設定」を参照してください。
   - このプロバイダーの SDK 設定プロパティの詳細については、「AWS SDK とツールのリファレンスガイド」の「コンテナ認証情報プロバイダー」を参照してください。

7. Amazon EC2 インスタンスメタデータサービス

   IAM ロールを作成し、インスタンスにアタッチします。インスタンス上の SDK for C++ アプリケーションは、ロールによって提供された認証情報をインスタンスメタデータから取得しようとします。

   - このロールの設定とメタデータの使用の詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 の IAM ロール」および「インスタンスメタデータの操作」を参照してください。
   - このプロバイダーの SDK 設定プロパティの詳細については、「AWS SDK とツールのリファレンスガイド」の「IMDS 認証情報プロバイダー」を参照してください。

認証情報プロバイダーチェーンは、GitHub `AWSCredentialsProviderChain`の AWS SDK for C++ ソースコードの で確認できます。

新しいユーザーが開始するための推奨アプローチに従った場合は、「開始方法」トピックAWS SDK for C++ AWS を使用した での認証の 中に AWS ログイン認証情報認証を設定します。その他の認証方法もさまざまな状況で役に立ちます。セキュリティリスクを避けるため、常に短期の認証情報を使

用することをお勧めします。その他の認証方法については、『AWS SDK とツールのリファレンスガイド』の「認証とアクセス」を参照してください。

明示的な認証情報プロバイダー

SDK による認証方法の検出のために認証情報プロバイダーチェーンに依存するのではなく、特定の認証情報プロバイダーを指定できます。そのためには、サービスクライアントのコンストラクタで認証情報を指定します。

次の例では、認証情報プロバイダーチェーンを使用する代わりに一時的なアクセス認証情報を直接指定して、Amazon Simple Storage Service クライアントを作成しています。

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

アイデンティティのキャッシュ

SDK は、認証情報や SSO トークンなど他のアイデンティティタイプをキャッシュします。デフォルトでは、SDK は遅延キャッシュ方式を使用します。この方式では、認証情報は最初のリクエスト時にロードされ、キャッシュされて、有効期限が近づいたら、次のリクエスト時に自動的に更新が試みられます。同じ `Aws::Client::ClientConfiguration` から作成されたクライアントはキャッシュを共有します。

AWS SDK for C++ のビルドに使用する CMake のパラメータ

このセクションに記載された CMake のパラメータを使用して、SDK のビルド方法をカスタマイズします。

これらのオプションは、CMake GUI ツールまたはコマンドラインで -D を使用して設定できます。例:

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

CMake の一般的な変数とオプション

SDK ソースコードのビルドプロセスに影響する **cmake** の一般的な変数とオプションを次に示します。

> **Note**
>
> これらのパラメータは、SDK for C++ 自体の SDK ソースコードをビルドする際に使用します。

トピック

- 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 定義に基づいて任意のクライアントをビルドします。定義を `code-generation/api-definitions` フォルダに配置し、この引数を **cmake** に渡します。**cmake** の設定ステップでクライアントが生成され、ビルドのサブディレクトリとして組み込まれます。これは、API Gateway サービスのいずれかを使用するための C++ クライアントを生成する場合に特に便利です。例:

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

> **Note**
>
> ADD_CUSTOM_CLIENTS パラメータを使用するには、Python 2.7、Java (JDK 1.8 以上)、Maven がインストールされており、PATH に含まれている必要があります。

AUTORUN_UNIT_TESTS

ON の場合は、ビルド後にユニットテストを自動的に実行します。

値

ON | OFF

デフォルト

ON

AWS_AUTORUN_LD_LIBRARY_PATH

CMake により自動的に実行されるユニットテスト用に LD_LIBRARY_PATH に追加するパス。オーバーライドされた依存関係にカスタムランタイムライブラリが必要な場合は、このパスを設定します。

値

String[]

デフォルト

(N/A)

AWS_SDK_WARNINGS_ARE_ERRORS

ON の場合、コンパイラの警告をエラーとして扱います。新しいコンパイラや一般的でないコンパイラでエラーが発生する場合は、これを OFF にしてみてください。

値

ON | OFF

デフォルト

ON

AWS_USE_CRYPTO_SHARED_LIBS

共有 crypto ライブラリが見つかった場合、FindCrypto がそのライブラリを使用するように設定します。代わりに BUILD_SHARED_LIBS を使用するように設定するには、これを OFF にします。

値

ON | OFF

デフォルト

VOFF

AWS_TEST_REGION

統合テストに使用する AWS リージョン。

値

String[]

デフォルト

(N/A)

BUILD_BENCHMARKS

ON の場合、ベンチマーク実行可能ファイルをビルドします。

値

ON | OFF

デフォルト

VOFF

BUILD_DEPS

ON の場合、サードパーティの依存関係をビルドします。

値

ON | OFF

デフォルト

ON

BUILD_ONLY

使用するクライアントのみをビルドします。aws-cpp-sdk-transfer などの高レベル SDK に設定すると、BUILD_ONLY では低レベルのクライアント依存関係を解決します。選択したプロジェクトに関連する統合テストやユニットテストが存在する場合は、それらもビルドされます。これはリスト引数であり、値はセミコロン (;) 文字で区切られます。例:

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

> **Note**
>
> コア SDK モジュール aws-sdk-cpp-core は、BUILD_ONLY パラメータの値に関係なく常にビルドされます。

BUILD_OPTEL

ON の場合、OpenTelemetry に基づくトレース実装がビルドされます。

値

ON | OFF

デフォルト

VOFF

BUILD_SHARED_LIBS

CMake に組み込まれているオプションですが、可視性を高めるためにここで明示しています。ON の場合、共有ライブラリをビルドし、それ以外の場合は、静的ライブラリのみをビルドします。

> **Note**
>
> SDK を動的リンクするには、すべてのビルドターゲットに対して USE_IMPORT_EXPORT シンボルを定義する必要があります。

値

ON | OFF

デフォルト

ON

BYPASS_DEFAULT_PROXY

ON の場合、IXmlHttpRequest2 使用時にマシンのデフォルトプロキシ設定をバイパスします。

値

ON | OFF

デフォルト

ON

CPP_STANDARD

C++ 14 および 17 コードベースで使用するカスタム C++ 標準を指定します。

値

11 | 14 | 17

デフォルト

11

CURL_INCLUDE_DIR

`libcurl` ヘッダーを含む curl インクルードディレクトリへのパス。

値

選択した *include* ディレクトリへの文字列パス。例えば、*D:/path/to/dir/with/curl/include*。

デフォルト

(N/A)

CURL_LIBRARY

リンク対象の curl ライブラリファイルへのパス。このライブラリは、アプリケーションの要件に応じて静的ライブラリまたはインポートライブラリになります。

値

curl ライブラリファイルへの文字列パス。例えば、*D:/path/to/static/libcur/file/ie/libcurl.lib.a*。

デフォルト

(N/A)

CUSTOM_MEMORY_MANAGEMENT

カスタムメモリマネージャーを使用するには、値を 1 に設定します。すべての STL 型がカスタムの割り当てインターフェイスを使用するように、独自のアロケーターを導入できます。値を `0` に設定しても、Windows での DLL の安全性を高めるために STL テンプレート型の使用を検討するとよいでしょう。

静的リンクが `ON` の場合、カスタムメモリ管理はデフォルトで off (`0`) になります。動的リンクが `ON` の場合、カスタムメモリ管理はデフォルトで on (1) になり、DLL 間でのメモリ割り当てと解放を回避します。

> **Note**
>
> リンカーの不一致エラーを防ぐには、ビルドシステム全体で同じ値 (`0` または `1`) を使用する必要があります。

SDK によるメモリ割り当てを処理する独自のメモリマネージャーを導入するには、`-DCUSTOM_MEMORY_MANAGEMENT` を設定し、SDK に依存するすべてのビルドターゲットに `USE_AWS_MEMORY_MANAGEMENT` を定義する必要があります。

DISABLE_INTERNAL_IMDSV1_CALLS

`ON` の場合、インスタンスメタデータサービス の V1 API への内部呼び出しは行われません。`OFF` の場合、IMDSv2 呼び出しが失敗すると IMDSv1 にフォールバックされます。IMDSv1 および IMDSv2 の詳細については、「Amazon EC2 ユーザーガイド」の「インスタンスメタデータを使用してインスタンスメタデータにアクセスする」を参照してください。

値

ON | OFF

デフォルト

VOFF

ENABLE_ADDRESS_SANITIZER

`ON` の場合、gcc または clang の Address Sanitizer をオンにします。

値

ON | OFF

デフォルト

VOFF

ENABLE_CURL_LOGGING

`ON` の場合、curl の内部ログを SDK のロガーにパイプ出力します。

値

ON | OFF

デフォルト

VOFF

ENABLE_HTTP_CLIENT_TESTING

ON の場合、対応する HTTP クライアントテストスイートをビルドして実行します。

値

ON | OFF

デフォルト

VOFF

enable_rtti

RTTI (実行時型情報) を有効にして SDK をビルドするかどうかを制御します。

値

ON | OFF

デフォルト

ON

ENABLE_TESTING

SDK ビルド時にユニットテストおよび統合テストプロジェクトをビルドするかどうかを制御します。

値

ON | OFF

デフォルト

ON

ENABLE_UNITY_BUILD

ON の場合、ほとんどの SDK ライブラリは 1 つの生成された .cpp ファイルとしてビルドされます。これにより、静的ライブラリのサイズを大幅に削減し、コンパイル時間を短縮できます。

値

ON | OFF

デフォルト

VOFF

ENABLE_VIRTUAL_OPERATIONS

このパラメータは通常、コード生成のために REGENERATE_CLIENTS と併用されます。

ENABLE_VIRTUAL_OPERATIONS が ON で、REGENERATE_CLIENTS が ON の場合、サービスクライアントのオペレーション関連の関数は virtual としてマークされます。

ENABLE_VIRTUAL_OPERATIONS が OFF で、REGENERATE_CLIENTS が ON の場合、オペレーション関連の関数には virtual が追加されず、サービスクライアントクラスは final としてマークされます。

ENABLE_VIRTUAL_OPERATIONS が OFF の場合、SDK はコンパイル時に gcc および clang の -ffunction-sections および -fdata-sections コンパイラフラグも追加します。

詳細については、GitHub の「CMake Parameters」を参照してください。

値

ON | OFF

デフォルト

ON

ENABLE_ZLIB_REQUEST_COMPRESSION

これをサポートするサービスの場合、リクエストコンテンツは圧縮されます。依存関係が使用可能な場合、デフォルトでオンになります。

値

ON | OFF

デフォルト

ON

FORCE_CURL

Windows のみ。ON の場合、デフォルトの WinHTTP データ転送プロバイダーの代わりに curl クライアントの使用を適用します。

値

ON | OFF

デフォルト

VOFF

FORCE_SHARED_CRT

ON の場合、SDK は C ランタイムに動的にリンクされます。それ以外の場合は、BUILD_SHARED_LIBS 設定を使用します (SDK の以前のバージョンとの下位互換性に必要な場合があります)。

値

ON | OFF

デフォルト

ON

G

Visual Studio ソリューションや Xcode プロジェクトなどのビルドアーティファクトを生成します。

Windows での例

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

詳細については、使用中のプラットフォームに対応した CMake ドキュメントを参照してください。

MINIMIZE_SIZE

ENABLE_UNITY_BUILD の上位互換設定。ON の場合、ENABLE_UNITY_BUILD および追加のバイナリサイズ削減設定が有効になります。

値

ON | OFF

デフォルト

VOFF

NO_ENCRYPTION

ON の場合、デフォルトのプラットフォーム固有の暗号化実装がライブラリに組み込まれないようにします。独自の暗号化実装を組み込む場合は、この設定を ON にします。

値

ON | OFF

デフォルト

VOFF

NO_HTTP_CLIENT

ON の場合、デフォルトのプラットフォーム固有の HTTP クライアントがライブラリに組み込まれないようにします。ON の場合、独自のプラットフォーム固有の HTTP クライアント実装を提供する必要があります。

値

ON | OFF

デフォルト

VOFF

REGENERATE_CLIENTS

ON の場合、このパラメータでは生成されたすべてのコードが削除され、code-generation/api-definitions フォルダからクライアントディレクトリが再生成されます。例:

```
-DREGENERATE_CLIENTS=1
```

> **Note**
>
> REGENERATE_CLIENTS パラメータを使用するには、Python 2.7、Java (JDK 1.8 以上)、Maven がインストールされており、PATH に含まれている必要があります。

REGENERATE_DEFAULTS

ON の場合、このパラメータでは生成されたすべてのデフォルトコードが削除され、code-generation/defaults フォルダから再生成されます。例:

```
-DREGENERATE_DEFAULTS=1
```

> **Note**
>
> REGENERATE_DEFAULTS パラメータを使用するには、Python 2.7、Java (JDK 1.8 以上)、Maven がインストールされており、PATH に含まれている必要があります。

SIMPLE_INSTALL

ON の場合、インストールプロセスでは bin/ と lib/ 以下にプラットフォーム固有の中間ディレクトリは挿入されません。1 つのインストールディレクトリでマルチプラットフォーム向けリリースを作成する場合は、OFF に設定します。

値

ON | OFF

デフォルト

ON

TARGET_ARCH

クロスコンパイルやモバイルプラットフォーム向けにビルドを行うには、ターゲットプラットフォームを指定する必要があります。デフォルトでは、ビルドはホストオペレーティングシステムを検出し、検出されたオペレーティングシステム向けにビルドします。

> **Note**
>
> TARGET_ARCH が ANDROID の場合、追加のオプションを使用できます。「Android CMake の変数とオプション」を参照してください。

値

WINDOWS | LINUX | APPLE | ANDROID

USE_CRT_HTTP_CLIENT

ON の場合、共通ランタイム HTTP クライアントが使用され、WinHttp や libcurl などのレガシーシステムはビルドも組み込みもされません。

値

ON | OFF

デフォルト

VOFF

USE_IXML_HTTP_REQUEST_2

Windows のみ。ON の場合、HTTP スタックに com オブジェクト IXmlHttpRequest2 を使用します。

値

ON | OFF

デフォルト

VOFF

USE_OPENSSL

ON の場合、SDK は OpenSSL を使用してビルドされます。それ以外の場合は、awslabs/aws-lc を使用します。AWS-LC は AWS およびその顧客向けに AWS 暗号化チームが管理している汎用暗号化ライブラリです。このパラメータを OFF にすると、OpenSSL の代わりに AWS-LC がシステムのデフォルトディレクトリにインストールされます。既に OpenSSL がシステムにインストールされている場合、このオプションは使用しないでください。

値

ON | OFF

デフォルト

ON

USE_TLS_V1_2

ON の場合、HTTP クライアントは TLS 1.2 を適用します。

値

ON | OFF

デフォルト

ON

USE_TLS_V1_3

ON の場合、HTTP クライアントは TLS 1.3 を適用します。

値

ON | OFF

デフォルト

VOFF

Android CMake の変数とオプション

SDK の Android ビルドを作成する場合 (TARGET_ARCH を ANDROID に設定している場合) は、次の変数を使用します。

トピック

- 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

Android のみ。どのアプリケーションバイナリインターフェイス (ABI) 向けにコードを出力するかを制御します。

> **Note**
>
> 有効なすべての Android ABI 値が現在サポートされているわけではありません。

値

arm64 | armeabi-v7a | x86_64 | x86 | mips64 | mips

デフォルト

armeabi-v7a

ANDROID_BUILD_CURL

Android のみ。ON の場合、curl もビルドします。

値

ON | OFF

デフォルト

ON

ANDROID_BUILD_OPENSSL

Android のみ。ON の場合、OpenSSL もビルドします。

値

ON | OFF

デフォルト

ON

ANDROID_BUILD_ZLIB

Android のみ。ON の場合、Zlib もビルドします。

値

ON | OFF

デフォルト

ON

ANDROID_NATIVE_API_LEVEL

Android のみ。SDK がビルド対象とする API レベルを制御します。ANDROID_STL を gnustl に設定した場合は、任意の API レベルを選択できます。libc++ を使用する場合は、21 以上の API レベルを使用する必要があります。

デフォルト

STL の選択によって異なります。

ANDROID_STL

Android のみ。SDK が使用する C++ 標準ライブラリの実装 (バリエーション) を指定します。

> **Important**
>
> `gnustl` オプションを使用すると、SDK 内でパフォーマンスの問題が発生する可能性があります。libc++_shared または libc++_static を使用することを強くお勧めします。

値

libc++_shared | libc++_static | gnustl_shared | gnustl_static

デフォルト

libc++_shared

ANDROID_TOOLCHAIN_NAME

Android のみ。SDK のビルドに使用されるコンパイラを制御します。

> **Note**
>
> Android NDK では GCC が廃止となっているため、デフォルト値を使用することをお勧めします。

デフォルト

standalone-clang

DISABLE_ANDROID_STANDALONE_BUILD

Android のみ。デフォルトでは、Android ビルドは NDK スクリプトによってコンストラクトされるスタンドアロンの clang ベースツールチェーンを使用します。独自のツールチェーンを使用するには、このオプションを ON にします。

値

ON | OFF

デフォルト

VOFF

NDK_DIR

Android のみ。ビルドシステムが Android NDK を検索する必要のあるオーバーライドパスを指定します。この変数を設定しない場合、ビルドシステムはデフォルトで環境変数 (ANDROID_NDK) を参照します。

AWS SDK for C++ でのログ記録の設定と使用

AWS SDK for C++ には、実行中に SDK によって実行されたアクションの記録を生成する、設定可能なログ記録が含まれています。ログ記録を有効にするには、SDKOptions の LogLevel をアプリケーションに適した詳細レベルに設定します。

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

7 つの詳細レベルから選択できます。デフォルトは Off で、ログは生成されません。Trace に設定すると、最も詳細なログが出力され、Fatal に設定すると、致命的なエラー状態のみをレポートする最小限のメッセージが生成されます。

アプリケーションでログ記録が有効になると、SDK によって aws_sdk_<date>.log というデフォルトの命名パターンに従って、実行可能ファイルのディレクトリにログファイルが生成されます。プレフィックス命名オプションによって生成されたログファイルは、ログファイルのアーカイブや削除ができるように、1 時間ごとにローテーションされます。

SDK の後期バージョンでは、基盤となる AWS 共通ランタイム (CRT) ライブラリへの依存度が高まっています。これらのライブラリは、各種 SDK 間で共通の機能や基本的なオペレーションを提供します。CRT ライブラリからのすべてのログメッセージは、デフォルトで SDK for C++ にリダイレクトされます。SDK for C++ に指定したログレベルとログ記録システムは、CRT にも適用されます。

前の例では、CRT は LogLevel::Info を継承し、Info レベルのメッセージも同じファイルに記録します。

CRT ライブラリのログ記録は、出力を別のログファイルにリダイレクトするか、CRT からのメッセージに別のログレベルを設定することで、個別に制御できます。多くの場合、CRT ライブラリの

詳細レベルを下げることで、ログが過剰にならないようにするのが有効です。例えば、CRT 出力のみのログレベルを次のように Warn に設定できます。

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

オプションでメソッド InitializeAWSLogging を使用することで、DefaultLogSystem の詳細レベルとログ出力先を制御できます。ログファイル名のプレフィックスを設定したり、出力先をファイルではなくストリームに変更したりすることもできます。

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

または、DefaultLogSystem を使用する代わりに、この方法を使用して独自のログ記録実装を提供することもできます。

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

メソッド InitializeAWSLogging を呼び出す場合は、プログラムの終了時に ShutdownAWSLogging を呼び出してリソースを解放します。

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

ログ記録を使用した統合テストの例

```
#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;
}
```

カスタムログ記録のための **Aws::Utils::Logging::DefaultLogSystem** のサブクラスの例

次のコードは、AWS SDK for C++ の一部である Aws::Utils::Logging::DefaultLogSystem クラスを継承する方法を示しています。この例では、ProcessFormattedStatement 仮想関数をオーバーライドしてログ記録をカスタマイズしています。

Aws::Utils::Logging::DefaultLogSystem は、AWS SDK for C++ 内でカスタムログ記録のために Aws::Utils::Logging::LogSystemInterface を継承しているクラスの 1 つです。

```
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;
}
```

GitHub で完全な例をご覧ください。

AWS SDK for C++ での HTTP クライアントのオーバーライド

Windows のデフォルトの HTTP クライアントは WinHTTP です。他のすべてのプラットフォームのデフォルトの HTTP クライアントは curl です。

オプションで、任意のサービスクライアントのコンストラクタに渡すカスタム `HttpClientFactory` を作成することで、HTTP クライアントのデフォルトをオーバーライドできます。HTTP クライアントをオーバーライドするには、SDK を curl サポートを有効にしてビルドする必要があります。Linux および macOS では curl サポートはデフォルトで有効ですが、Windows では追加のビルド手順が必要です。Windows で curl サポートを有効にして SDK をビルドする方法の詳細については、「Windows での AWS SDK for C++ のビルド」を参照してください。

AWS SDK for C++ 内で **`HttpClient`** と **`AWSClient`** で使用される iostream の制御

デフォルトでは、すべてのレスポンスは `stringbuf` をバックエンドとする入力ストリームを使用します。必要に応じて、このデフォルト動作をオーバーライドできます。例えば、Amazon S3 `GetObject` を使用していて、ファイル全体をメモリにロードしないようにする場合は、`AmazonWebServiceRequest` で `IOStreamFactory` を使用してラムダを渡し、ファイルストリームを作成できます。

ファイルストリームリクエストの例

```
 //! 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**
>
> GitHub には、その他のリソースもあります。完全なコード例は AWS コードサンプルリポジトリにあります。

AWS SDK for C++ でのカスタム libcrypto ライブラリの使用

デフォルトでは、AWS SDK for C++ は Transport Layer Security (TLS) にシステムの既定暗号化ライブラリを使用します。ただし SDK for C++ では、ソースから SDK をビルドする際に、オプションで別の libcrypto ライブラリを使用するように設定できます。これは、すべての暗号化オペレーションが OpenSSL のカスタム実装に置き換えられることを意味します。例えば、AWS-LC ライブラリを FIPS モードで使用することで、アプリケーションで FIPS 準拠が可能になります。

SDK for C++ にカスタム libcrypto を組み込む方法

ステップ 1: libcrypto ライブラリをビルドまたは入手する

AWS-LC は代替 libcrypto ライブラリの一例ですが、OpenSSL の任意のディストリビューションまたは OpenSSL 相当のものであれば機能します。

SDK for C++ とそれに依存する CRT はどちらも libcrypto を暗号化処理に使用するため、依存関係も同様に扱う必要があります。SDK for C++ では、リクエストが SDK の CRT S3 機能を使用するかどうかに応じて、2 種類の HTTP クライアントのいずれかに依存します。CRT は特に s2n に依存しており、これは起動時に初期化される TLS 実装です。SDK と s2n の両方には、BUILD_SHARED_LIBS の値に関係なく共有 libcrypto を強制使用するための CMake パラメータがあります。一般的に、CRT HTTP クライアントと通常の HTTP クライアントには、同じ libcrypto を使用します。この場合、どちらも依存ツリー内で OpenSSL を参照することになります。そのために SDK では AWS_USE_CRYPTO_SHARED_LIBS を使用し、s2n (CRT ベースの呼び出し) では S2N_USE_CRYPTO_SHARED_LIBS を使用します。依存関係の解決はこれら 2 つのライブラリ間で同じであり、通常は一致するように設定されますが、明示的に異なるように設定することもできます。

例えば、libcrypto ライブラリとして AWS-LC を使用するには、次のようにビルドします。

```
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 .
```

ステップ 2: curl をソースからビルドするか、libcrypto ライブラリを組み込んだ curl ディストリビューションを使用する

SDK for C++ を使用するには、HTTP リクエストを行うための HTTP クライアントがシステムにインストールされている必要があります。HTTP クライアントは、使用予定の libcrypto を組み込んでビルドする必要があります。HTTP クライアントは TLS オペレーションを担当するため、libcrypto ライブラリを使用します。

次の例では、インストール済みの AWS-LC を使用して curl ライブラリを再ビルドします。

```
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
```

ステップ 3: libcrypto ライブラリと curl ライブラリを使用して SDK をビルドする

これで、先ほど作成した libcrypto アーティファクトと curl アーティファクトを使用して SDK for C++ をビルドできるようになりました。SDK のこのビルドでは、すべての暗号化機能にカスタム libcrypto ライブラリを使用します。

```
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 .
```

すべてのステップを Docker イメージにまとめる

次のサンプル Docker ファイルは、Amazon Linux 2023 環境でこれらのステップを実装する方法を示しています。

```
# 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++ の使用

このセクションでは、「AWS SDK for C++ の開始方法」で取り上げていない内容を含め、AWS SDK for C++ の一般的な使用方法について説明します。

サービス固有のプログラミング例については、「AWS SDK for C++ のコード例」を参照してください。

トピック

- AWS SDK for C++ の初期化と終了
- AWS SDK for C++ を使用した AWS のサービス リクエストの実行
- AWS SDK for C++ を使用した非同期プログラミング
- AWS SDK for C++ で使用可能なユーティリティモジュール
- AWS SDK for C++ でのメモリ管理
- AWS SDK for C++ でのエラーの処理

AWS SDK for C++ の初期化と終了

AWS SDK for C++ を使用するアプリケーションは初期化する必要があります。同様に、アプリケーションの終了前に SDK を終了する必要があります。どちらのオペレーションも、初期化およびシャットダウンプロセスと、その後の SDK への呼び出しに影響する設定オプションを受け入れます。

AWS SDK for C++ を使用するすべてのアプリケーションは `aws/core/Aws.h` ファイルをインクルードする必要があります。

AWS SDK for C++ は、`Aws::InitAPI` を呼び出して初期化する必要があります。アプリケーションの終了前に、`Aws::ShutdownAPI` を呼び出して SDK を終了する必要があります。各メソッドは `Aws::SDKOptions` の引数を受け入れます。SDK への他のすべての呼び出しは、これら 2 つのメソッド呼び出しの間に実行できます。

`Aws::InitAPI` と **`Aws::ShutdownAPI`** の間に行われるすべての AWS SDK for C++ 呼び出しは、中かっこで囲むか、2 つのメソッド間に呼び出される関数によって行われる必要があります。

基本的なスケルトンアプリケーションを次に示します。

```
#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++ とその依存関係は C++ の静的オブジェクトを使用しており、静的オブジェクトの破棄順序は C++ 標準で明確に定義されていません。静的変数の破棄順序が不定であることによるメモリ問題を避けるため、**`Aws::InitAPI`** と **`Aws::ShutdownAPI`** の呼び出しを他の静的オブジェクトにラップしないでください。

AWS SDK for C++ を使用した AWS のサービス リクエストの実行

AWS のサービス にプログラムでアクセスするために、SDK では各 AWS のサービス に対してクライアントクラスを使用します。たとえば、アプリケーションが Amazon EC2 にアクセスする必要がある場合、アプリケーションはそのサービスとインターフェイスをとる Amazon EC2 クライアントオブジェクトを作成します。次に、サービスクライアントを使用して、その AWS のサービス に対してリクエストを実行します。

AWS のサービス にリクエストを行うには、まずサービスクライアントを作成して設定する必要があります。コードが使用する各 AWS のサービス には、それぞれ独自のライブラリと、サービスとやり取りするための専用の型があります。クライアントは、サービスが提供する各 API オペレーションに対応するメソッドをそれぞれ公開しています。

クライアントクラスの名前空間は `Aws::`*`Service`*`::`*`Service`*`Client` という規則に従います。例えば、AWS Identity and Access Management (IAM) のクライアントクラスは `Aws::IAM::IAMClient` であり、Amazon S3 のクライアントクラスは `Aws::S3::S3Client` です。

AWS サービスのすべてのクライアントクラスはスレッドセーフです。

クライアントクラスをインスタンス化する際には、AWS の認証情報を指定する必要があります。コード内、環境変数、または共有 AWS `config` 認証情報ファイルと共有 `credentials` 設定ファ

イルから、認証情報を提供できます。認証情報の詳細については、「推奨される IAM Identity Center 認証の設定手順」または「使用可能な別の認証情報プロバイダー」を参照してください。

AWS SDK for C++ を使用した非同期プログラミング

非同期 SDK メソッド

多くのメソッドに対して、AWS SDK for C++ では同期版と非同期版の両方が用意されています。メソッド名に Async サフィックスが付いている場合、そのメソッドは非同期です。例えば、Amazon S3 メソッド PutObject は同期ですが、PutObjectAsync は非同期です。

すべての非同期オペレーションと同様に、非同期 SDK メソッドは main の処理が終了する前に制御を返します。例えば、PutObjectAsync メソッドは Amazon S3 バケットへのファイルのアップロードが完了する前に制御を返します。アップロードオペレーションが続行されている間、アプリケーションは他の非同期メソッドの呼び出しなど、他のオペレーションを実行できます。非同期オペレーションが終了すると、関連付けられたコールバック関数が呼び出されて、アプリケーションに通知されます。

以下のセクションでは、PutObjectAsync 非同期メソッドの呼び出しを示すコード例について説明します。各セクションでは、コード例のソースファイル全体の個々の部分に焦点を当てています。

SDK 非同期メソッドの呼び出し

一般に、SDK メソッドの非同期バージョンは次の引数を受け入れます。

- 同期版のメソッドで使用されるものと同じ型のリクエストオブジェクトへの参照。
- レスポンスハンドラーコールバック関数への参照。このコールバック関数は、非同期オペレーションが終了すると呼び出されます。引数の 1 つには、オペレーションの結果が含まれます。
- AsyncCallerContext オブジェクトへのオプションの shared_ptr。このオブジェクトはレスポンスハンドラーコールバック関数に渡されます。テキスト情報をコールバックに渡すために使用できる UUID プロパティが含まれています。

次に示す uploadFileAsync メソッドは、SDK の Amazon S3 PutObjectAsync メソッドを設定して呼び出し、ファイルを Amazon S3 バケットに非同期的にアップロードします。

この関数は S3Client オブジェクトと PutObjectRequest オブジェクトへの参照を受け取ります。これらのオブジェクトは、非同期呼び出しが完了するまで存在している必要があるため、main 関数から渡されます。

AsyncCallerContext オブジェクトへの shared_ptr が生成されます。その UUID プロパティに Amazon S3 オブジェクト名が設定されます。デモンストレーションの目的で、レスポンスハンドラーコールバック関数はこのプロパティにアクセスし、その値を出力します。

PutObjectAsync の呼び出しには、uploadFileAsyncFinished というレスポンスハンドラーコールバック関数への参照引数が含まれます。このコールバック関数については、次のセクションで詳しく説明します。

```
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;
}
```

非同期オペレーションのリソースは、オペレーションが完了するまで存在している必要があります。例えば、クライアントおよびリクエストオブジェクトは、アプリケーションが完了通知を受け取るま

で存在している必要があります。アプリケーション自体は、非同期オペレーションが完了するまで終了できません。

このため、uploadFileAsync メソッドは S3Client および PutObjectRequest オブジェクトを uploadFileAsync メソッド内部で作成してローカル変数に保存するのではなく、それらのオブジェクトへの参照を引数として受け取ります。

この例では、PutObjectAsync メソッドは非同期処理を開始した直後に呼び出し元へ制御を返すため、呼び出し元ではアップロード処理中に他のタスクを実行できます。

uploadFileAsync メソッド内でクライアントがローカル変数に保存されていた場合、メソッド終了時にスコープ外となります。しかし、クライアントオブジェクトは非同期処理が完了するまで存在し続ける必要があります。

非同期オペレーションの完了通知

非同期オペレーションが完了すると、アプリケーションのレスポンスハンドラーコールバック関数が呼び出されます。この通知には、オペレーションの結果が含まれます。結果は、同期版メソッドが返すものと同じ Outcome 型クラスに保存されます。このコード例では、結果は PutObjectOutcome オブジェクトにあります。

レスポンスハンドラーのコールバック関数の例 uploadFileAsyncFinished を次に示します。非同期オペレーションが成功したか失敗したかを確認します。std::condition_variable を使用して、非同期オペレーションが終了したことをアプリケーションスレッドに通知します。

```
// 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();
}
```

非同期オペレーションが終了すると、それに関連するリソースは解放可能になります。アプリケーション自体も必要に応じて終了できます。

次のコードでは、アプリケーションが uploadFileAsync および uploadFileAsyncFinished メソッドをどのように使用するかを示しています。

アプリケーションは S3Client および PutObjectRequest オブジェクトを生成し、非同期処理が終了するまで保持します。uploadFileAsync を呼び出した後、アプリケーションは任意のオペレーションを実行できます。わかりやすくするために、この例では std::mutex と std::condition_variable を使用して、アップロードオペレーションの完了をレスポンスハンドラーコールバック関数が通知するまで待機します。

```
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;
}
```

GitHub で完全な例をご覧ください。

AWS SDK for C++ で使用可能なユーティリティモジュール

AWS SDK for C++ には、C++ での AWS アプリケーション開発の複雑さを軽減するための ユーティリティモジュールが多数含まれています。

HTTP スタック

接続プーリングを提供し、スレッドセーフで、必要に応じて再利用できる HTTP スタック。詳細については、「AWS クライアント設定」を参照してください。

ヘッダー	/aws/core/http/
API ドキュメント	Aws::Http

文字列ユーティリティ

基本的な文字列処理関数 (trim、lowercase、数値変換など)。

[Header] (ヘッダー)	aws/core/utils/StringUtils.h
API ドキュメント	Aws::Utils::StringUtils

ハッシュユーティリティ

SHA256、MD5、Base64、SHA256_HMAC などのハッシュ関数。

[Header] (ヘッダー)	/aws/core/utils/HashingUtils.h
API ドキュメント	Aws::Utils::HashingUtils

JSON パーサー

軽量ながら完全な機能を備えた JSON パーサー (*cJSON* を最小限の構造でラップ)。

[Header] (ヘッダー)	/aws/core/utils/json/JsonSerializer.h
API ドキュメント	Aws::Utils::Json::JsonValue

XML パーサー

軽量な XML パーサー (*tinyxml2* を最小限の構造でラップ)。RAII パターンがインターフェイスに追加されています。

[Header] (ヘッダー)	/aws/core/utils/xml/XmlSerializer.h
API ドキュメント	Aws::Utils::Xml

AWS SDK for C++ でのメモリ管理

AWS SDK for C++ は、ライブラリ内でのメモリの割り当てと解放を制御する方法が提供します。

> **Note**
>
> カスタムメモリ管理を利用できるのは、コンパイル時定数 USE_AWS_MEMORY_MANAGEMENT を定義してビルドされたライブラリを使用する場合のみです。
> コンパイル時定数なしでビルドされたバージョンのライブラリを使用した場合、InitializeAWSMemorySystem などのグローバルなメモリシステム関数は機能せず、代わりにグローバルな new および delete 関数が使用されます。

このコンパイル時定数の詳細については、「STL および AWS 文字列とベクター」を参照してください。

メモリの割り当てと解放

メモリを割り当てまたは解放するには

1. サブクラス 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**
>
> 必要に応じて AllocateMemory の型シグネチャを変更できます。

2. Aws::SDKOptions 構造体を使用して、カスタムメモリマネージャーの使用を設定します。構造体のインスタンスを Aws::InitAPI に渡します。アプリケーションの終了前に、同じインスタンスで Aws::ShutdownAPI を呼び出し、SDK をシャットダウンする必要があります。

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

  // ... do stuff

  Aws::ShutdownAPI(options);

  return 0;
}
```

STL および AWS 文字列とベクター

メモリマネージャーで初期化されると、AWS SDK for C++ はすべての割り当てと解放をメモリマネージャーに委譲します。メモリマネージャーが存在しない場合、SDK はグローバルな new と delete を使用します。

カスタム STL アロケーターを使用する場合は、すべての STL オブジェクトの型シグネチャをアロケーションポリシーに合わせて変更する必要があります。SDK の実装とインターフェイスで STL が広く使用されているため、SDK で 1 つの方式を採用すると、デフォルトの STL オブジェクトの直接的な受け渡しや割り当て制御が難しくなります。あるいは、内部ではカスタムアロケーターを使用しつつ、インターフェイスでは標準およびカスタム STL オブジェクトの両方を許容するハイブリッド方式も考えられますが、この方式ではメモリ問題の調査が難しくなる可能性があります。

この問題への解決策は、メモリシステムのコンパイル時定数 USE_AWS_MEMORY_MANAGEMENT を用いて、SDK が使用する STL 型を制御することです。

このコンパイル時定数が有効 (on) になっている場合、各型は、AWS メモリシステムに接続されたカスタムアロケーターを使用する STL 型に解決されます。

コンパイル時定数が無効 (off) の場合、すべての `Aws::*` 型は対応するデフォルトの `std::*` 型に解決されます。

SDK の **AWSAllocator.h** ファイルからのコード例

```
#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
```

コード例では、AwsAllocator はコンパイル時定数に応じてカスタムアロケーターまたはデフォルトアロケーターになります。

SDK の **AWSVector.h** ファイルからのコード例

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

このコード例では、Aws::* 型を定義しています。

コンパイル時定数が有効 (on) の場合、この型はカスタムメモリアロケーターと AWS メモリシステムを使用するベクターに解決されます。

コンパイル時定数が無効 (off) の場合、この型はデフォルトの型パラメータを持つ通常の std::vector に解決されます。

型エイリアスは、コンテナ、文字列ストリーム、文字列バッファなど、SDK 内でメモリ割り当てを行うすべての std:: 型に使用されます。AWS SDK for C++ はこれらの型を使用します。

残された課題

SDK 内のメモリ割り当ては制御できますが、STL 型は依然としてモデルオブジェクトの initialize および set メソッドの文字列パラメータを通じて、パブリックインターフェイスに多く使用されています。STL を使用せずに、代わりに文字列やコンテナを使用している場合、サービス呼び出しのたびに多くの一時オブジェクトを作成する必要があります。

STL を使用せずにサービス呼び出しを行う際の一時オブジェクトやメモリ割り当てを削減するために、次の対策を実装しています。

- 文字列を受け取るすべての Init/Set 関数には、const char* を受け取るオーバーロードがあります。
- コンテナ (マップ/ベクター) を受け取る Init/Set 関数には、1 つのエントリを受け取る add バリアントがあります。
- バイナリデータを受け取る Init/Set 関数には、データへのポインタと length 値を受け取るオーバーロードがあります。
- (オプション) 文字列を受け取るすべての Init/Set 関数には、ゼロ終端されていない const char* と length 値を受け取るオーバーロードもあります。

ネイティブ SDK デベロッパー向けメモリ制御

SDK コード内では次のルールに従います。

- new と delete は使用せず、代わりに Aws::New<> と Aws::Delete<> を使用します。

- new[] と delete[] は使用せず、Aws::NewArray<> と Aws::DeleteArray<> を使用します。
- std::make_shared を使用せず、Aws::MakeShared を使用します。
- 1 つのオブジェクトへの一意のポインタには Aws::UniquePtr を使用します。一意のポインタを生成するには、Aws::MakeUnique 関数を使用します。
- オブジェクトの配列への一意のポインタには、Aws::UniqueArray を使用します。一意のポインタを生成するには、Aws::MakeUniqueArray 関数を使用します。
- STL コンテナを直接使用しないでください。Aws:: typedef のいずれかを利用するか、必要なコンテナに typedef を追加してください。例:

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

- SDK に渡して管理される外部ポインタには shared_ptr を使用します。共有ポインタは、オブジェクトの生成方法に合った破棄ポリシーで初期化する必要があります。SDK がポインタのクリーンアップを行わない場合は、生ポインタを使用できます。

AWS SDK for C++ でのエラーの処理

AWS SDK for C++ で例外は使用されませんが、コード内で例外を使用することはできます。すべてのサービスクライアントは、結果とエラーコードを含む結果オブジェクトを返します。

エラー条件の処理例

```
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++ からの AWS のサービス の呼び出し

以下のセクションには、AWS SDK for C++を使用してAWSサービスを操作する方法を示す例、チュートリアル、タスク、ガイドが含まれています。

AWS SDK for C++ を初めて使用する場合は、最初に ご利用開始にあたって トピックに目を通すとよいでしょう。

その他のコード例は、GitHub の C++ コード例フォルダにあります。

その他のコード例は、このガイドの コードの例 章、または GitHub の AWS コードサンプルリポジトリにあります。

トピック

- AWS SDK for C++ コード例の開始方法
- AWS SDK for C++ でのランタイムエラーのトラブルシューティングの開始方法
- AWS SDK for C++ を使用した AWS のサービス の呼び出しのガイド付きコード例

AWS SDK for C++ コード例の開始方法

コード例の構造

Github の C++ コード例フォルダには、各 AWS サービスのプロジェクトフォルダが含まれています。通常、フォルダ内の個々の .cpp ソースファイルは、そのサービスの特定の機能またはアクションを示しています。例えば、Amazon DynamoDB の場合、データベースからの項目の取得とデータベースへの項目のアップロードは 2 つの異なるタイプのアクションであるため、DynamoDB フォルダにはそれぞれに個別のファイル (`get_item.cpp` と `put_item.cpp`) があります。各 .cpp ファイルには、スタンドアロン実行可能ファイルのエントリポイントとなる `main()` 関数が含まれています。プロジェクトの実行可能ファイルはビルドシステムによって指定されたフォルダに生成され、サンプルソースファイルごとに 1 つの実行可能ファイルがあります。実行可能ファイルの名前は、プラットフォームの慣習 (`{name}.exe`、単に `{name}` など) や、`CMakeLists.txt` に適用されるカスタムプレフィックス (`run_` など) に従います。

コード例の機能を実行するには

1. GitHub の AWS コードサンプルリポジトリから必要なコード例をダウンロードします。
2. .cpp ファイルを開き、その `main()` 関数および呼び出されるメソッドを確認します。

3. 「AWS SDK for C++ の開始方法」にある開始例に従ってプロジェクトをビルドします。ビルドすると、プロジェクト内の各ソースファイルに対応する実行可能ファイルが生成されます。
4. 選択した機能の実行可能ファイルを実行します。
   - コマンドプロンプトで、*.cpp ファイルの名前に基づいた実行可能ファイルを使用して、そのプログラムを実行します。
   - IDE 内で作業している場合は、デモンストレーションする機能の .cpp ファイルを選択し、スタートアップオプション (またはスタートアップオブジェクト) に設定します。

ユニットテスト

コード例のテストは GoogleTest フレームワークを使用して記述されています。詳細については、GoogleTest ウェブサイトの「GoogleTest Primer」を参照してください。

各コード例のユニットテストは、tests サブフォルダ内に独自の CMakeLists.txt ファイルと共に含まれています。各コード例のソースファイルには、それに対応する名前の gtest_*<source file>* テストファイルがあります。そのサブフォルダにあるテスト実行可能ファイルの名前は *<AWS #####>*_gtests です。

CMakeLists.txt ファイル

各サービスのフォルダには、CMakeLists.txt という名前のファイルが含まれています。これらのファイルの多くには、次のようなコンストラクトが含まれています。

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

フォルダ内の .cpp ファイルごとに、CMakeLists.txt ファイルによって、そのソースコードファイル名 (拡張子なし) に基づいた名前で実行可能ファイル (cmake: add_executable) がビルドされます。

Visual Studio でのコード例のビルドとデバッグ

Amazon S3 コード例のビルドと実行

1. Amazon S3 のコード例のソースを入手します。この手順では、AWS SDK for C++ を使用した Amazon S3 のコード例 コード例を使用して、Visual Studio での立ち上げと実行を行います。

2. Windows Explorer で、s3 フォルダ (例: \aws-doc-sdk-examples\cpp\example_code\s3) に移動します。
3. s3 コード例フォルダを右クリックし、[Open with Visual Studio] (Visual Studio で開く) を選択します。 Visual Studio の CMake プロジェクトでは、「プロジェクト」ファイルがなく、フォルダ全体がプロジェクトとして扱われます。
4. Visual Studio のトップメニューにある [Configuration Selector] ドロップダウンで、選択した設定が、SDK をソースからビルドした際のビルドタイプと一致していることを確認します。 例えば、debug (SDK インストール手順の CMake コマンドラインでの -DCMAKE_BUILD_TYPE=Debug) を使用してソースからビルドした場合は、Debug 設定を選択する必要があります。
5. CMakeLists.txt ファイルを開きます。
6. [保存] をクリックします。CMakeLists.txt ファイルで [Save] (保存) をクリックするたびに、Visual Studio は CMake によって生成されたファイルを更新します。 [Output] (出力) タブを表示している場合、この生成による結果のログメッセージを確認できます。

   - [Output] タブには [Show output from:] (出力元の表示:) というドロップダウンボックスがあり、デフォルトで [CMake] オプションが選択されています。
   - 最後のメッセージ出力では、CMake の生成が終了したことが通知されます。
   - 最後のメッセージでこのように通知されない場合は、CMake ファイルに問題があります。この問題が解決するまで、次の手順には進まないでください。 「AWS SDK for C++ のビルドに関する問題のトラブルシューティング」を参照してください。
   - CMake キャッシュは CMake によって高速化のために使用されています。CMake の問題に対処する際は、表示されるエラーメッセージが最新の変更に基づいたものとなるように、ビルド環境を一度クリーンな状態にリセットすることが重要です。 Solution Explorer で CMakeLists.txt を右クリックし、[CMake Cache] (CMake キャッシュ) を選択してから、[Delete Cache] (キャッシュの削除) を実行します。CMake の問題を順次解決していく際は、この操作を頻繁に行います。
7. Visual Studio 内からコード例をビルドして実行するために、Visual Studio はコマンドラインとは異なるフォルダ構造に実行可能ファイルを配置します。コードを実行するには、SDK の実行可能ファイルを正しい場所にコピーする必要があります。 CMakeLists ファイル内の「TODO」行 (40 行目付近) を探し、Visual Studio 用にコメントアウトされた行を選択します。Visual Studio ではビルドタイプ専用のサブフォルダは使用しないため、含まれていません。 CMakeLists.txt ファイル内でコメントアウトされた行を Visual Studio 用に切り替えます。

8. CMake キャッシュを削除し (上記を参照)、CMakeLists.txt ファイルをクリックしてタブを選択/アクティブにしてから、再度 CMakeLists.txt ファイルで [Save] (保存) を選択して、CMake ビルドファイルの生成を開始します。
9. 実行する「プログラム」のソースファイルを開きます。
   - 例えば、list_buckets.cpp を開きます。
   - Amazon S3 のコード例フォルダは、紹介されている各機能が、それぞれ専用の実行可能ファイルでデモンストレーションされるように構成されています。 例えば、list_buckets.cpp は、バケットの一覧表示のみをデモンストレーションする実行可能ファイルになります。
10. トップメニューで、[Build] (ビルド) を選択し、[Build All] (すべてビルド) を選択します。
    - [Output] (出力) タブの [Show output from] (出力元の表示) には、[Build] (ビルド) の選択内容が反映され、すべてのビルドおよびリンクメッセージが表示されます。
    - 最後の出力は「Build All succeeded.」 になります。
    - これで、各ソースファイルに対応する実行可能ファイルが生成されました。 この結果を確認するには、ビルド出力ディレクトリ (例: \aws-doc-sdk-examples\cpp\example_code\s3\out\build\x64-Debug) を調べます。
    - 実行可能ファイルに「run_」というプレフィックスが付いているのは、CMakeLists.txt ファイルでの指示によります。
11. トップメニューには、緑の矢印と、デバッグターゲットのドロップダウンセレクターがあります。 run_list_buckets.exe を選択してください。
12. 緑色の矢印の実行ボタンをクリックして、スタートアップ項目を選択します。
13. Visual Studio のデバッグコンソールウィンドウが開き、コードの出力が表示されます。
14. キーを押してウィンドウを閉じるか手動でウィンドウを閉じて、プログラムを終了します。コード内にブレークポイントを設定し、再度実行すればそのポイントで停止します。

AWS SDK for C++ でのランタイムエラーのトラブルシューティングの開始方法

AWS SDK for C++ を使用してアプリケーションを開発していく中で、AWS マネジメントコンソール と AWS CLI の両方を使いこなせるようになっておくことも重要です。これらのツールは、ランタイムエラー発生時のさまざまなトラブルシューティングや診断に互換的に使用できます。

次のチュートリアルでは、こうしたトラブルシューティングおよび診断作業の一例を示します。ここでは、いくつかの異なる理由で発生する可能性がある Access denied エラーに焦点を当てます。このチュートリアルは、エラーの実際の原因をどのように特定するかの例を示します。ここでは、主に次の 2 つの原因に注目します。現在のユーザーに対してアクセス許可が正しくないこと、リソースが利用できないことです。

プロジェクトのソースと実行可能ファイルを取得するには

1. GitHub の AWS コードサンプルリポジトリから Amazon S3 のコード例フォルダをダウンロードします。
2. delete_bucket.cpp を開き、main() と DeleteBucket() の 2 つのメソッドがあることを確認します。DeleteBucket() は SDK を使用してバケットを削除します。
3. 「AWS SDK for C++ の開始方法」で説明しているのと同じビルド手順を使用して、Amazon S3 の例をビルドします。ビルドプロセスでは、ソースファイルごとに実行可能ファイルが生成されます。
4. ビルドされた実行可能ファイルが出力されたフォルダをコマンドプロンプトで開きます。実行可能ファイル run_create_bucket を実行します (実際のファイル名はオペレーティングシステムによって異なります)。これにより、アカウント内にバケット (後で削除するバケット) が作成されます。
5. コマンドプロンプトで実行可能ファイル run_delete_bucket を実行します。この例では、削除対象のバケットの名前をパラメータとして指定する必要があります。正しくないバケット名 (意図的にタイプミスを含めた名前) を入力し、トラブルシューティングを試してみましょう。
6. Access Denied エラーメッセージが表示されることを確認します。*Access Denied* エラーメッセージが表示されると、Amazon S3 に対して完全なアクセス許可を持つユーザーを作成したかどうかが疑問になります。次はこの点を確認します。

AWS CLI をインストールし、AWS への呼び出しを行っているユーザーの名前を確認します。

1. 最新の AWS CLI を開発マシンにインストールする方法については、「AWS Command Line Interface ユーザーガイド」の「AWS CLI のインストール」を参照してください。
2. AWS CLI が動作していることを確認するには、コマンドプロンプトを開き、コマンド aws -\-version を実行します。

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

3. AWS への呼び出しを実際に行っているユーザーの名前を取得するには、AWS CLI コマンド `aws sts get-caller-identity` を実行します。次の出力例では、ユーザー名は userX です。

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

認証情報の指定方法は多数ありますが、AWS SDK for C++ AWS を使用した での認証 の手順に従っている場合、このユーザー名は AWS の共有認証情報ファイルから取得されます。その手順の中で、ユーザーに AmazonS3FullAccess アクセス許可を付与しました。

> **Note**
>
> 一般的に、ほとんどの AWS CLI コマンドは次の構文構造に従います。
>
> ```
> $ aws <command> <subcommand> [options and parameters]
> ```
>
> ここで、*command* はサービス、*subcommand* はそのサービスで呼び出されるメソッドです。詳細については、「AWS Command Line Interface ユーザーガイド」の「AWS CLI のコマンド構造」を参照してください。

ユーザーにバケットを削除するためのアクセス許可があるかを確認するには

1. AWS マネジメントコンソール を開き、ログインします。詳細については、「AWS マネジメントコンソール の開始方法」を参照してください。
2. メインナビゲーションバーの [Search for services...] (サービスを検索...) に「**IAM**」と入力し、表示された結果から IAM サービスを選択します。
3. ダッシュボードのサイドバーまたは [IAM Resources] (IAM リソース) で、[Users] (ユーザー) を選択します。
4. アカウントで使用可能なユーザーの一覧から、前の手順で取得したユーザー名を選択します。

5. [Summary] (概要) ページの [Permissions] (アクセス許可) タブを選択し、ポリシー名の一覧から [AmazonS3FullAccess] を選択します。
6. [Policy summary] (ポリシーの概要) と JSON データを確認します。このユーザーが Amazon S3 サービスに対して完全な権限を持っていることを確認します。

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

このような除外による確認手順は、問題の原因を特定する上でよく使用されます。今回の場合、ユーザーに正しいアクセス許可があることが確認されたため、問題は他にあると考えられます。つまり、バケットにアクセスするための正しいアクセス許可があるにもかかわらず `Access Denied` エラーが発生する場合は、自分のものでないバケットにアクセスしようとしている可能性があります。トラブルシューティングの次の手順として、プログラムに指定されたバケット名を確認すると、その名前のバケットがアカウントに存在しないこと、つまりそのバケットに「アクセス」できないことがわかります。

コード例を正常に動作するように更新するには

1. `delete_bucket.cpp` の `main()` 関数に戻り、enum を使用してリージョンを自分のアカウントのリージョンに変更します。自分のアカウントのリージョンを見つけるには、AWS マネジメントコンソール にログインし、右上隅にあるリージョンを確認します。また `main()` 内で、バケット名を自分のアカウントに存在するバケットに変更します。現在のバケット名を見つける方法はいくつかあります。
   - このコード例のフォルダ内にある `run_list_buckets` 実行可能ファイルを使用して、バケットの名前をプログラムで取得できます。
   - あるいは、次の AWS CLI コマンドを使用して Amazon S3 バケットを一覧表示することもできます。

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

   - あるいは、AWS マネジメントコンソール エイリアスを使用することもできます。メインナビゲーションバーの [Search for services...] (サービスの検索) で、「**S3**」と入力します。バケットのページに、アカウントのバケットが一覧表示されます。
2. コードを再ビルドし、更新された実行可能ファイル `run_delete_bucket` を実行します。

3. AWS マネジメントコンソール または AWS CLI のいずれかを使用して、先ほど作成した Amazon S3 バケットが削除されたことを確認します。

AWS SDK for C++ を使用した AWS のサービス の呼び出しのガイド付きコード例

AWS または AWS のコード例を初めて使用する場合は、コード例の開始方法 から始めることをお勧めします。

AWS SDK for C++ による AWS サービスの操作方法を示すソースコードは、このガイドの コードの例 の章に掲載されています。または、GitHub の AWS コードサンプルリポジトリで直接入手できます。

このセクションでは、いくつかの AWS サービスを選択し、それらを使用する例を示します。以下のガイド付き例は、GitHub にあるコード例の一部です。

補足説明付きのサービス例 (完全なリストについては AWS コードサンプルリポジトリを参照)

サービス	サービスがプログラムに提供する内容の概要
Amazon CloudWatch ()	使用中の AWS リソースのメトリクスを収集・モニタリングする
– Amazon DynamoDB	NoSQL データベースサービス
Amazon Elastic Compute Cloud (Amazon EC2)	安全でサイズ変更可能なコンピューティングキャパシティ
Amazon Simple Storage Service (Amazon S3)	データの保存と取得 (バケットへのオブジェクトの保存)
Amazon Simple Queue Service (Amazon SQS)	ソフトウェアコンポーネント間でメッセージを送信、保存、受信するためのメッセージキューイングサービス

非同期メソッドの使用方法を示す例もあります。

AWS ドキュメントチームに新しいコード例を提案する場合は、GitHub のコントリビューションガイドラインを参照してリクエストを作成してください。チームとしては、個々の API コールではなく、幅広いシナリオを示すコード例を作成することを重視しています。

Windows でのコード例の使用

SDK バージョン 1.9 を使用して Windows でコード例をビルドする場合は、「AWS SDK for C++ のビルドに関する問題のトラブルシューティング」を参照してください。

AWS SDK for C++ を使用した Amazon CloudWatch のコード例

Amazon CloudWatch (CloudWatch) は、AWS クラウドリソースと、AWS で実行するアプリケーションのモニタリングサービスです。以下の例のように、AWS SDK for C++ を使用して CloudWatch をプログラミングできます。

Amazon CloudWatch は、AWS のリソースおよび AWS で実行しているアプリケーションをリアルタイムでモニタリングします。CloudWatch を使用してメトリクスを収集および追跡できます。メトリクスとは、リソースやアプリケーションについて測定できる変数です。CloudWatch アラームは、ユーザーが定義したルールに基づいて、通知を送信したり、モニタリングしているリソースに自動的に変更を加えたりします。

CloudWatch の詳細については、「Amazon CloudWatch ユーザーガイド」を参照してください。

> **Note**
>
> このガイドには、特定の手法を示すために必要最低限のコードのみを掲載していますが、完全なコード例は GitHub で入手できます。GitHub では、ソースファイル単体をダウンロードすることも、リポジトリをローカルにクローンして、すべての例を取得、ビルド、実行することもできます。

トピック

- CloudWatch からのメトリクスの取得
- カスタムメトリクスデータを発行する
- CloudWatch アラームの使用
- CloudWatch でのアラームアクションの使用
- CloudWatch へのイベントの送信

CloudWatch からのメトリクスの取得

前提条件

作業を始める前に「AWS SDK for C++ の開始方法」を読むことをお勧めします。

コード例をダウンロードし、「コード例の開始方法」の説明に従ってソリューションをビルドします。

例を実行するには、リクエストに使用するユーザープロファイルに、AWS のサービスとアクションに対する適切なアクセス許可が付与されている必要があります。詳細については、「AWS 認証情報の提供」を参照してください。

メトリクスの一覧表示

CloudWatch メトリクスを一覧表示するには、ListMetricsRequest を作成し、CloudWatchClient の `ListMetrics` 関数を呼び出します。`ListMetricsRequest` を使用して、名前空間、メトリクス名、またはディメンションで返されたメトリクスをフィルタリングできます。

> **Note**
>
> AWS サービスが提供するメトリクスとディメンションの一覧は、「Amazon CloudWatch ユーザーガイド」の「Amazon CloudWatch メトリクスとディメンションのリファレンス」で参照できます。

含まれるもの:

```
#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>
```

コード

```
        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();
    }
```

メトリクスは、GetMetrics 関数を呼び出すことで ListMetricsResult に返されます。結果はページ分割される場合があります。結果の次のバッチを取得するには、ListMetricsResult オブジェクトの GetNextToken 関数の戻り値を使用して元のリクエストオブジェクトで SetNextToken を呼び出し、変更したリクエストオブジェクトを ListMetrics の再呼び出しに渡します。

完全な例をご覧ください。

詳細情報

- 「Amazon CloudWatch API リファレンス」の「ListMetrics」。

カスタムメトリクスデータを発行する

複数の AWS のサービスが AWS/ で始まる名前空間で独自のメトリクスをパブリッシュします。(AWS/ で始まらない限り) 独自の名前空間を使用してカスタムメトリクスデータをパブリッシュすることもできます。

前提条件

作業を始める前に「AWS SDK for C++ の開始方法」を読むことをお勧めします。

コード例をダウンロードし、「コード例の開始方法」の説明に従ってソリューションをビルドします。

例を実行するには、リクエストに使用するユーザープロファイルに、AWS のサービスとアクションに対する適切なアクセス許可が付与されている必要があります。詳細については、「AWS 認証情報の提供」を参照してください。

カスタムメトリクスデータを発行する

独自のメトリクスデータをパブリッシュするには、CloudWatchClient の PutMetricData 関数を PutMetricDataRequest で呼び出します。PutMetricDataRequest には、データ用に使用するカス

タム名前空間と、MetricDatum オブジェクト内のデータポイント自体に関する情報が含まれている必要があります。

> **Note**
>
> AWS/ で始まる名前空間を指定することはできません。AWS/ で始まる名前空間は、Amazon Web Services 製品による利用のために予約されています。

含まれるもの:

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

コード

```
        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;
        }
```

完全な例をご覧ください。

詳細情報

- 「Amazon CloudWatch ユーザーガイド」の「Amazon CloudWatch メトリクスの使用」。
- 「Amazon CloudWatch ユーザーガイド」の「AWS 名前空間」。
- 「Amazon CloudWatch API リファレンス」の「PutMetricData」。

CloudWatch アラームの使用

前提条件

作業を始める前に「AWS SDK for C++ の開始方法」を読むことをお勧めします。

コード例をダウンロードし、「コード例の開始方法」の説明に従ってソリューションをビルドします。

例を実行するには、リクエストに使用するユーザープロファイルに、AWS のサービスとアクションに対する適切なアクセス許可が付与されている必要があります。詳細については、「AWS 認証情報の提供」を参照してください。

アラームの作成

CloudWatch メトリクスに基づいてアラームを作成するには、アラーム条件を指定した PutMetricAlarmRequest を引数として、CloudWatchClient の `PutMetricAlarm` 関数を呼び出します。

含まれるもの:

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

コード

```
        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;
        }
```

完全な例をご覧ください。

アラームの一覧表示

作成した CloudWatch アラームを一覧表示するには、結果のオプションを設定可能な DescribeAlarmsRequest を指定して、CloudWatchClient の `DescribeAlarms` 関数を呼び出します。

含まれるもの:

```
#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>
```

コード

```
        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();
    }
```

アラームのリストは getMetricAlarms を [により返される](#) DescribeAlarmsResultDescribeAlarms で呼び出すことで取得できます。

結果はページ分割される場合があります。結果の次のバッチを取得するには、DescribeAlarmsResult オブジェクトの GetNextToken 関数の戻り値を使用して元のリクエストオブジェクトで SetNextToken を呼び出し、変更したリクエストオブジェクトを DescribeAlarms の再呼び出しに渡します。

> **Note**
>
> また、特定のメトリクスのアラームを取得するには、CloudWatchClient の DescribeAlarmsForMetric 関数を使用します。使用方法は DescribeAlarms と同様です。

[完全な例](#)をご覧ください。

アラームの削除

CloudWatch アラームを削除するには、削除するアラームの名前を含む [DeleteAlarmsRequest](#) を指定して、CloudWatchClient の DeleteAlarms 関数を呼び出します。

含まれるもの:

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

コード

```
    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;
        }
```

完全な例をご覧ください。

詳細情報

- 「Amazon CloudWatch ユーザーガイド」の「Amazon CloudWatch アラームの作成」
- 「Amazon CloudWatch API リファレンス」の「PutMetricAlarm」
- 「Amazon CloudWatch API リファレンス」の「DescribeAlarms」
- 「Amazon CloudWatch API リファレンス」の「DeleteAlarms」

CloudWatch でのアラームアクションの使用

CloudWatch アラームアクションを使用して、Amazon EC2 インスタンスを自動的に停止、終了、再起動、または復旧するといったアクションを実行するアラームを作成できます。

アラームの作成時に PutMetricAlarmRequest の `SetAlarmActions` 関数を使用することで、アラームアクションをアラームに追加できます。

前提条件

作業を始める前に「AWS SDK for C++ の開始方法」を読むことをお勧めします。

コード例をダウンロードし、「コード例の開始方法」の説明に従ってソリューションをビルドします。

例を実行するには、リクエストに使用するユーザープロファイルに、AWS のサービスとアクションに対する適切なアクセス許可が付与されている必要があります。詳細については、「AWS 認証情報の提供」を参照してください。

アラームアクションの有効化

CloudWatch アラームのアクションを有効にするには、アクションを有効にする 1 つ以上のアラームの名前を含む EnableAlarmActionsRequest を指定して、CloudWatchClient の EnableAlarmActions を呼び出します。

含まれるもの:

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

コード

```
    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;
```

完全な例をご覧ください。

アラームアクションの無効化

CloudWatch アラームのアクションを無効にするには、アクションを無効にする 1 つ以上のアラームの名前を含む DisableAlarmActionsRequest を指定して、CloudWatchClient の `DisableAlarmActions` を呼び出します。

含まれるもの:

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

コード

```
        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;
        }
```

完全な例をご覧ください。

詳細情報

- 「Amazon CloudWatch ユーザーガイド」の「インスタンスを停止、終了、再起動、または復旧するアラームを作成する」
- 「Amazon CloudWatch API リファレンス」の「PutMetricAlarm」
- 「Amazon CloudWatch API リファレンス」の「EnableAlarmActions」
- 「Amazon CloudWatch API リファレンス」の「DisableAlarmActions」

CloudWatch へのイベントの送信

CloudWatch Events は、AWS リソースの変更を示すシステムイベントをほぼリアルタイムで Amazon EC2 インスタンス、Lambda 関数、Kinesis ストリーム、Amazon ECS タスク、Step Functions ステートマシン、Amazon SNS トピック、Amazon SQS キュー、または組み込みターゲットに振り分けます。簡単なルールを使用して、一致したイベントを 1 つ以上のターゲット関数またはストリームに振り分けることができます。

> **Note**
>
> これらのコードスニペットでは、「AWS SDK for C++ の開始方法」の内容を理解し、「AWS 認証情報の提供」の情報を使用してデフォルトの AWS 認証情報を設定していることを前提としています。

イベントの追加

カスタム CloudWatch イベントを追加するには、各イベントの詳細を提供する PutEventsRequestEntry オブジェクトを 1 つ以上含む PutEventsRequest オブジェクトを指定して、CloudWatchEventsClient の PutEvents 関数を呼び出します。イベントのソースとタイプ、イベントに関連付けられたリソースなど、エントリの複数のパラメータを指定できます。

> **Note**
>
> putEvents への呼び出しごとに最大 10 個のイベントを指定できます。

含まれるもの:

```
#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>
```

コード

```
        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;
        }
```

ルールの追加

ルールを作成または更新するには、ルールの名前、イベントパターン、関連付ける IAM ロール、実行頻度を記述するスケジュール式などのオプションパラメータを含む PutRuleRequest を指定して、CloudWatchEventsClient の `PutRule` 関数を呼び出します。

含まれるもの:

```
#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>
```

コード

```
        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;
        }
```

ターゲットの追加

ターゲットは、ルールがトリガーされたときに呼び出されるリソースです。ターゲットの例には、Amazon EC2 インスタンス、Lambda 関数、Kinesis ストリーム、Amazon ECS タスク、Step Functions ステートマシン、組み込みターゲットなどがあります。

ルールにターゲットを追加するには、更新するルールを含む PutTargetsRequest とルールに追加するターゲットのリストを指定して、CloudWatchEventsClient の `PutTargets` 関数を呼び出します。

含まれるもの:

```
#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>
```

コード

```
        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;
```

```
        }
```

完全な例をご覧ください。

詳細情報

- 「Amazon CloudWatch Events ユーザーガイド」の「PutEvents によるイベントの追加」
- 「Amazon CloudWatch Events ユーザーガイド」の「ルールのスケジュール式」
- 「Amazon CloudWatch Events ユーザーガイド」の「CloudWatch Events のイベントタイプ」
- 「Amazon CloudWatch Events ユーザーガイド」の「イベントおよびイベントパターン」
- 「Amazon CloudWatch Events API リファレンス」の「PutEvents」
- 「Amazon CloudWatch Events API リファレンス」の「PutTargets」
- 「Amazon CloudWatch Events API リファレンス」の「PutRule」

AWS SDK for C++ を使用した Amazon DynamoDB の例

Amazon DynamoDB は、フルマネージド NoSQL データベースサービスであり、シームレスなスケーラビリティを備えた高速で予測可能なパフォーマンスを提供します。以下の例では、AWS SDK for C++ を使用して Amazon DynamoDB をプログラミングする方法を示しています。

> **Note**
>
> このガイドには、特定の手法を示すために必要最低限のコードのみを掲載していますが、完全なコード例は GitHub で入手できます。GitHub では、ソースファイル単体をダウンロードすることも、リポジトリをローカルにクローンして、すべての例を取得、ビルド、実行することもできます。

トピック

- DynamoDB でのテーブルの使用
- DynamoDB での項目の使用

DynamoDB でのテーブルの使用

テーブルは、DynamoDB データベースのすべての項目のコンテナです。DynamoDB のデータの追加または削除を行う前に、テーブルを作成する必要があります。

テーブルごとに、以下を定義する必要があります。

- AWS アカウント および AWS リージョン で一意のテーブル名。
- すべての値が一意である必要のあるプライマリキー。同じテーブル内でプライマリキーが重複する項目は存在できません。

  プライマリキーは、単一のパーティション (ハッシュ) キーで構成されるシンプルなプライマリキーにするか、パーティションとソート (範囲) キーで構成される複合プライマリキーにすることができます。

  各キーバリューには、ScalarAttributeType クラスによって列挙されるデータ型が関連付けられています。キー値は、バイナリ (B)、数値 (N)、または文字列 (S) になります。詳細については、「Amazon DynamoDB デベロッパーガイド」の「命名規則とデータ型」を参照してください。
- テーブル用に予約された読み込み/書き込みキャパシティーユニットの数を定義するプロビジョニングされたスループットの値。

> **Note**
>
> Amazon DynamoDB の料金表は、テーブルに設定したプロビジョニングされたスループット値に基づくため、テーブルに必要と予想される分だけのキャパシティーを予約します。テーブルのプロビジョニングされたスループットはいつでも変更できるため、必要に応じてキャパシティーを調整できます。

テーブルを作成する

新しい DynamoDB テーブルを作成するには、DynamoDB クライアントの `CreateTable` メソッドを使用します。テーブルのプライマリキーを識別するために使用する、テーブル属性とテーブルスキーマを構築する必要があります。また、初期プロビジョンドスループット値とテーブル名を指定する必要があります。`CreateTable` は非同期オペレーションです。`GetTableStatus` はテーブルの作成中は CREATING を返し、テーブルが使用可能になると ACTIVE を返します。

シンプルプライマリキーを使用してテーブルを作成する

このコードでは、シンプルプライマリキー (「Name」) を持つテーブルを作成します。

含まれるもの:

```
#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>
```

コード

```
//! 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);
}
```

完全な例をご覧ください。

複合プライマリキーを使用してテーブルを作成する

別の AttributeDefinition および KeySchemaElement を CreateTableRequest に追加します。

含まれるもの:

```
#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>
```

Code

```
//! 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);
}
```

GitHub で完全な例をご覧ください。

テーブルの一覧表示

特定のリージョン内のテーブルを一覧表示するには、DynamoDB クライアントの `ListTables` メソッドを呼び出します。

含まれるもの:

```
#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>
```

コード

```
//! 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;
}
```

デフォルトでは、1 回の呼び出しごとに最大 100 個のテーブルが返されます。返された ListTablesOutcome オブジェクトに対して GetExclusiveStartTableName を使用して、最後に評価されたテーブルを取得します。この値を使用して、前回の一覧表示で返された最後の値以降から、一覧表示を開始できます。

完全な例をご覧ください。

テーブルに関する情報を取得する

DynamoDB クライアントの DescribeTable メソッドを呼び出すことで、テーブルの詳細を確認できます。

含まれるもの:

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

Code

```
//! 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();
}
```

GitHub で完全な例をご覧ください。

テーブルを変更する

DynamoDB クライアントの UpdateTable メソッドを呼び出すことで、テーブルのプロビジョンドスループット値を随時変更できます。

含まれるもの:

```
#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>
```

コード

```
//! 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);
}
```

完全な例をご覧ください。

テーブルの削除

DynamoDB クライアントの DeleteTable メソッドを呼び出し、それにテーブルの名前を渡します。

含まれるもの:

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

Code

```
//! 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();
}
```

GitHub で完全な例をご覧ください。

詳細

- 「Amazon DynamoDB デベロッパーガイド」の「テーブルの使用に関するガイドライン」
- 「Amazon DynamoDB デベロッパーガイド」の「DynamoDB でのテーブルの使用」

DynamoDB での項目の使用

DynamoDB では、項目とは 属性の集合体です。各属性には名前と値があります。属性値はスカラー型、セット型、ドキュメント型のいずれかです。詳細については、「Amazon DynamoDB デベロッパーガイド」の「命名規則とデータ型」を参照してください。

テーブルから項目を取得する

DynamoDB クライアントの GetItem メソッドを呼び出します。取得する項目のテーブル名とプライマリキー値を含む GetItemRequest オブジェクトをそのメソッドに渡します。GetItemResult オブジェクトが返されます。

返された GetItemResult オブジェクトの GetItem( ) メソッドを使用して、項目に関連付けられているキー Aws::String と値 AttributeValue のペアの Aws::Map を取得できます。

含まれるもの:

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

Code

```
//! 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();
}
```

GitHub で完全な例をご覧ください。

テーブルに項目を追加する

各項目を表すキー Aws::String と値 AttributeValue のペアを作成します。これらには、テーブルのプライマリキーフィールドの値を含める必要があります。プライマリキーで特定される項目がすでにある場合、フィールドはリクエストによって更新されます。AddItem メソッドを使用して、それらの項目を PutItemRequest に追加します。

含まれるもの:

```
#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>
```

Code

```
//! 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);
}
```

GitHub で完全な例をご覧ください。

テーブルの既存の項目の更新

DynamoDBClient の UpdateItem メソッドを使用し、テーブル名、プライマリキー値、更新するフィールドとその対応する値を指定することで、テーブルに既に存在する項目の属性を更新できます。

インポート

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

コード

```
//! 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);
}
```

完全な例をご覧ください。

詳細

- 「Amazon DynamoDB デベロッパーガイド」の「項目の使用に関するガイドライン」
- 「Amazon DynamoDB デベロッパーガイド」の「DynamoDB での項目の使用」

AWS SDK for C++ を使用する Amazon EC2 の例

Amazon Elastic Compute Cloud (Amazon EC2) は、ユーザーがソフトウェアシステムを構築してホストするための、コンピューティング能力を自在に拡張および縮小できるウェブサービス (実際には Amazon のデータセンター内のサーバー) です。以下の例のように、AWS SDK for C++ を使用して Amazon EC2 をプログラミングできます。

> **Note**
>
> このガイドには、特定の手法を示すために必要最低限のコードのみを掲載していますが、完全なコード例は GitHub で入手できます。GitHub では、ソースファイル単体をダウンロードすることも、リポジトリをローカルにクローンして、すべての例を取得、ビルド、実行することもできます。

トピック

- Amazon EC2 インスタンスの管理
- Amazon EC2 での Elastic IP アドレスの使用
- Amazon EC2 のリージョンとアベイラビリティーゾーンの使用
- Amazon EC2 のキーペアでの作業

- Amazon EC2 でのセキュリティグループの使用

Amazon EC2 インスタンスの管理

前提条件

作業を始める前に「AWS SDK for C++ の開始方法」を読むことをお勧めします。

コード例をダウンロードし、「コード例の開始方法」の説明に従ってソリューションをビルドします。

例を実行するには、リクエストに使用するユーザープロファイルに、AWS のサービスとアクションに対する適切なアクセス許可が付与されている必要があります。詳細については、「AWS 認証情報の提供」を参照してください。

インスタンスの作成

新しい EC2 インスタンスを作成するには、EC2Client の RunInstances 関数の呼び出しで、使用する Amazon マシンイメージ (AMI) とインスタンスタイプを RunInstancesRequest で指定して渡します。

含まれるもの:

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

コード

```
    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;
    }
```

完全な例をご覧ください。

インスタンスを開始する

Amazon EC2 インスタンスを開始するには、EC2Client の StartInstances 関数の呼び出しで、開始するインスタンスの ID を StartInstancesRequest で指定して渡します。

含まれるもの:

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

コード

```
    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;
    }
```

完全な例をご覧ください。

インスタンスを停止する

Amazon EC2 インスタンスを停止するには、EC2Client の StopInstances 関数の呼び出しで、停止するインスタンスの ID を StopInstancesRequest で指定して渡します。

含まれるもの:

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

コード

```
    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;
    }
```

完全な例をご覧ください。

インスタンスを再起動する

Amazon EC2 インスタンスを再起動するには、EC2Client の RebootInstances 関数の呼び出しで、再起動するインスタンスの ID を RebootInstancesRequest で指定して渡します。

含まれるもの:

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

コード

```
    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;
   }
```

完全な例をご覧ください。

インスタンスを記述する

インスタンスを一覧表示するには、DescribeInstancesRequest を作成し、EC2Client の DescribeInstances 関数を呼び出します。これにより DescribeInstancesResponse オブジェクトが返されます。このオブジェクトを使用して、AWS アカウント および AWS リージョン の Amazon EC2 インスタンスを一覧表示できます。

インスタンスは予約ごとにグループ化されています。それぞれの予約は、インスタンスを起動した StartInstances の呼び出しに対応しています。インスタンスを一覧表示するには、まず DescribeInstancesResponse クラスの GetReservations 関数を呼び出し、返された各 Reservation オブジェクトに対して getInstances を呼び出します。

含まれるもの:

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

Code

```
   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;
        }
    }
```

結果はページ分割されます。さらに結果を取得するには、結果オブジェクトの GetNextToken 関数から返された値を元のリクエストオブジェクトの SetNextToken 関数に渡した後、次の DescribeInstances の呼び出しで同じリクエストオブジェクトを使用します。

完全な例をご覧ください。

インスタンスのモニタリングを有効にする

CPU やネットワークの使用率、使用可能なメモリ、ディスクの残り容量など、Amazon EC2 インスタンスのさまざまな側面をモニタリングできます。インスタンスのモニタリングの詳細については、「Amazon EC2 ユーザーガイド」の「Amazon EC2 のモニタリング」を参照してください。

インスタンスのモニタリングを開始するには、モニタリングするインスタンスの ID で MonitorInstancesRequest を作成し、EC2Client の `MonitorInstances` 関数に渡します。

含まれるもの:

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

コード

```
    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;
    }
```

完全な例をご覧ください。

インスタンスのモニタリングを無効にする

インスタンスのモニタリングを停止するには、モニタリングを停止するインスタンスの ID で UnmonitorInstancesRequest を作成し、EC2Client の UnmonitorInstances 関数に渡します。

含まれるもの:

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

コード

```
    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;
    }
```

完全な例をご覧ください。

詳細情報

- 「Amazon EC2 API リファレンス」の「RunInstances」
- 「Amazon EC2 API リファレンス」の「DescribeInstances」
- 「Amazon EC2 API リファレンス」の「StartInstances」
- 「Amazon EC2 API リファレンス」の「StopInstances」
- 「Amazon EC2 API リファレンス」の「RebootInstances」
- 「Amazon EC2 API リファレンス」の「DescribeInstances」
- 「Amazon EC2 API リファレンス」の「MonitorInstances」
- 「Amazon EC2 API リファレンス」の「UnmonitorInstances」

Amazon EC2 での Elastic IP アドレスの使用

前提条件

作業を始める前に「AWS SDK for C++ の開始方法」を読むことをお勧めします。

コード例をダウンロードし、「コード例の開始方法」の説明に従ってソリューションをビルドします。

例を実行するには、リクエストに使用するユーザープロファイルに、AWS のサービスとアクションに対する適切なアクセス許可が付与されている必要があります。詳細については、「AWS 認証情報の提供」を参照してください。

Elastic IP アドレスを割り当てる

Elastic IP アドレスを使用するにはまずアカウントに 1 つ割り当ててから、それをインスタンスまたはネットワークインターフェイスに関連付けます。

Elastic IP アドレスを割り当てるには、ネットワークタイプ (Classic EC2 または VPC) が含まれる AllocateAddressRequest オブジェクトを使用して EC2Client の `AllocateAddress` 関数を呼び出します。

> **Warning**
>
> 2022 年 8 月 15 日に、EC2-Classic の提供を終了し h ます。EC2-Classic は、VPC への移行をお勧めします。詳細については、「Linux インスタンス用 Amazon EC2 ユーザーガイド」または「Windows インスタンス用 Amazon EC2 ユーザーガイド」の「EC2-Classic から VPC への移行」を参照してください。ブログ記事「EC2-Classic Networking は販売終了になります — 準備方法はこちら」も参照してください。

レスポンスオブジェクトの AllocateAddressResponse クラスには割り当て ID が含まれており、この ID を使用することで、そのアドレスをインスタンスに関連付けることができます。その際は、EC2Client の `AssociateAddress` 関数の呼び出しで、割り当て ID とインスタンス ID を AssociateAddressRequest で指定して渡します。

含まれるもの:

```
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/AllocateAddressRequest.h>
#include <aws/ec2/model/AssociateAddressRequest.h>
#include <iostream>
```

コード

```
    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;
```

完全な例をご覧ください。

Elastic IP アドレスの記述

Elastic IP アドレスを一覧表示するには、EC2Client の `DescribeAddresses` 関数を呼び出します。返される結果オブジェクトには DescribeAddressesResponse が含まれています。このオブジェクトを使用して、アカウントの Elastic IP アドレスを表す Address オブジェクトのリストを取得できます。

含まれるもの:

```
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/DescribeAddressesRequest.h>
#include <aws/ec2/model/DescribeAddressesResponse.h>
```

```
#include <iomanip>
#include <iostream>
```

コード

```
    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;
    }
```

完全な例をご覧ください。

Elastic IP アドレスを解放する

Elastic IP アドレスを解放するには、EC2Client の `ReleaseAddress` 関数を呼び出して、解放する Elastic IP アドレスの割り当て ID を含む ReleaseAddressRequest を渡します。

含まれるもの:

```
#include <aws/ec2/EC2Client.h>
```

```
#include <aws/ec2/model/ReleaseAddressRequest.h>
#include <iostream>
```

コード

```
    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;
    }
```

Elastic IP アドレスを解放すると、その IP アドレスは AWS IP アドレスプールに解放され、後で使用できないことがあります。DNS レコード、およびそのアドレスと通信するすべてのサーバーまたはデバイスを更新してください。解放済みの Elastic IP アドレスを解放しようとした場合に、そのアドレスが既に別の AWS アカウントに割り当てられていると AuthFailure エラーが発生します。

デフォルト VPC を使用している場合、Elastic IP アドレスを解放すると、関連付けられているインスタンスから自動的に関連付けが解除されます。Elastic IP アドレスを解放せずに関連付けのみを解除するには、EC2Client の `DisassociateAddress` 関数を使用します。

デフォルト以外の VPC を使用している場合は、開放しようとする前に必ず `DisassociateAddress` を使用して Elastic IP アドレスの関連付けを解除する必要があります。そうしないと、Amazon EC2 からエラー (InvalidIPAddress.InUse) が返されます。

完全な例をご覧ください。

詳細情報

- 「Amazon EC2 ユーザーガイド」の「Elastic IP アドレス」。
- 「Amazon EC2 API リファレンス」の「AllocateAddress」
- 「Amazon EC2 API リファレンス」の「DescribeAddresses」
- 「Amazon EC2 API リファレンス」の「ReleaseAddress」

Amazon EC2 のリージョンとアベイラビリティーゾーンの使用

前提条件

作業を始める前に「AWS SDK for C++ の開始方法」を読むことをお勧めします。

コード例をダウンロードし、「コード例の開始方法」の説明に従ってソリューションをビルドします。

例を実行するには、リクエストに使用するユーザープロファイルに、AWS のサービスとアクションに対する適切なアクセス許可が付与されている必要があります。詳細については、「AWS 認証情報の提供」を参照してください。

リージョンの記述

AWS アカウント で使用可能な AWS リージョン を一覧表示するには、EC2Client の `DescribeRegions` 関数の呼び出しで、DescribeRegionsRequest を渡します。

結果オブジェクトに入れて DescribeRegionsResponse が返されます。その `GetRegions` 関数を呼び出して、各リージョンを表す Region オブジェクトのリストを取得します。

含まれるもの:

```
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/DescribeRegionsRequest.h>
```

コード

```
    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;
    }
```

完全な例をご覧ください。

アベイラビリティーゾーンの記述

アカウントで使用可能な各アベイラビリティーゾーンを一覧表示するには、EC2Client の DescribeAvailabilityZones 関数の呼び出しで、DescribeAvailabilityZonesRequest を渡します。

結果オブジェクトに入れて DescribeAvailabilityZonesResponse が返されます。その GetAvailabilityZones 関数を呼び出して、各アベイラビリティーゾーンを表す AvailabilityZone オブジェクトのリストを取得します。

含まれるもの:

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

コード

```
   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;

    }
```

完全な例をご覧ください。

詳細情報

- 「Amazon EC2 ユーザーガイド」の「リージョンとアベイラビリティーゾーン」
- 「Amazon EC2 API リファレンス」の「DescribeRegions」
- 「Amazon EC2 API リファレンス」の「DescribeAvailabilityZones」

Amazon EC2 のキーペアでの作業

前提条件

作業を始める前に「AWS SDK for C++ の開始方法」を読むことをお勧めします。

コード例をダウンロードし、「コード例の開始方法」の説明に従ってソリューションをビルドします。

例を実行するには、リクエストに使用するユーザープロファイルに、AWS のサービスとアクションに対する適切なアクセス許可が付与されている必要があります。詳細については、「AWS 認証情報の提供」を参照してください。

キーペアの作成

キーペアを作成するには、EC2Client の `CreateKeyPair` 関数の呼び出しで、キーの名前を CreateKeyPairRequest で指定して渡します。

含まれるもの:

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

```
#include <fstream>
```

コード

```
    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;
        }

    }
```

完全な例をご覧ください。

キーペアの記述

キーペアを一覧表示したり、それらに関する情報を取得したりするには、EC2Client の DescribeKeyPairs 関数の呼び出しで、DescribeKeyPairsRequest を渡します。

DescribeKeyPairsResponse が返されます。その GetKeyPairs 関数を呼び出して、キーペアの一覧にアクセスできます。この関数は、KeyPairInfo オブジェクトのリストを返します。

含まれるもの:

```
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/DescribeKeyPairsRequest.h>
#include <aws/ec2/model/DescribeKeyPairsResponse.h>
#include <iomanip>
#include <iostream>
```

コード

```
    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;
    }
```

完全な例をご覧ください。

キーペアの削除

キーペアを削除するには、EC2Client の DeleteKeyPair 関数の呼び出しで、削除するキーペアの名前を DeleteKeyPairRequest で指定して渡します。

含まれるもの:

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

コード

```
    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;
    }
```

完全な例をご覧ください。

詳細情報

- 「Amazon EC2 ユーザーガイド」の「Amazon EC2 キーペア」
- 「Amazon EC2 API リファレンス」の「CreateKeyPair」
- 「Amazon EC2 API リファレンス」の「DescribeKeyPairs」
- 「Amazon EC2 API リファレンス」の「DeleteKeyPair」

Amazon EC2 でのセキュリティグループの使用

前提条件

作業を始める前に「AWS SDK for C++ の開始方法」を読むことをお勧めします。

コード例をダウンロードし、「コード例の開始方法」の説明に従ってソリューションをビルドします。

例を実行するには、リクエストに使用するユーザープロファイルに、AWS のサービスとアクションに対する適切なアクセス許可が付与されている必要があります。詳細については、「AWS 認証情報の提供」を参照してください。

セキュリティグループの作成

セキュリティグループを作成するには、EC2Client の `CreateSecurityGroup` 関数の呼び出しで、キーの名前を CreateSecurityGroupRequest で指定して渡します。

含まれるもの:

```
#include <aws/ec2/EC2Client.h>
```

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

コード

```
    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;
```

完全な例をご覧ください。

セキュリティグループを設定する

セキュリティグループは、Amazon EC2 インスタンスへのインバウンド (ingress) とアウトバウンド (egress) トラフィックの両方を制御できます。

セキュリティグループに Ingress ルールを追加するには、EC2Client の AuthorizeSecurityGroupIngress 関数の呼び出しで、セキュリティグループの名前と割り当てるアクセスルール (IpPermission) を AuthorizeSecurityGroupIngressRequest オブジェクトで指定して渡します。以下の例では、セキュリティグループへの IP のアクセス許可の追加方法を説明します。

含まれるもの:

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

コード

```
   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;
   }
```

セキュリティグループに egress ルールを追加するには、EC2Client の `AuthorizeSecurityGroupEgress` 関数の呼び出しで、同様のデータを AuthorizeSecurityGroupEgressRequest で指定して渡します。

完全な例をご覧ください。

セキュリティグループを記述する

セキュリティグループのリストを取得したり情報を確認したりするには、EC2Client の DescribeSecurityGroups 関数の呼び出しで、DescribeSecurityGroupsRequest を渡します。

結果オブジェクトに入れて DescribeSecurityGroupsResponse が返されます。その GetSecurityGroups 関数を呼び出して、セキュリティグループのリストにアクセスできます。この関数は SecurityGroup オブジェクトのリストを返します。

含まれるもの:

```
#include <aws/ec2/EC2Client.h>
#include <aws/ec2/model/DescribeSecurityGroupsRequest.h>
#include <aws/ec2/model/DescribeSecurityGroupsResponse.h>
#include <iomanip>
#include <iostream>
```

コード

```
    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());
```

完全な例をご覧ください。

セキュリティグループの削除

セキュリティグループを削除するには、EC2Client の DeleteSecurityGroup 関数の呼び出しで、削除するセキュリティグループの ID を DeleteSecurityGroupRequest で指定して渡します。

含まれるもの:

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

コード

```
    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;
    }
```

完全な例をご覧ください。

詳細情報

- 「Amazon EC2 ユーザーガイド」の「Amazon EC2 セキュリティグループ」
- 「Amazon EC2 ユーザーガイド」の「Linux インスタンスへのインバウンドトラフィックの認可」
- 「Amazon EC2 API リファレンス」の「CreateSecurityGroup」
- 「Amazon EC2 API リファレンス」の「DescribeSecurityGroups」
- 「Amazon EC2 API リファレンス」の「DeleteSecurityGroup」
- 「Amazon EC2 API リファレンス」の「AuthorizeSecurityGroupIngress」

AWS SDK for C++ を使用した Amazon S3 のコード例

Amazon S3 は、どこからでも任意の量のデータを保存および取得できるオブジェクトストレージです。Amazon S3 とやり取りするために、AWS SDK for C++ には複数のクラスが用意されています。

> **Note**
>
> このガイドには、特定の手法を示すために必要最低限のコードのみを掲載していますが、完全なコード例は GitHub で入手できます。GitHub では、ソースファイル単体をダウンロードすることも、リポジトリをローカルにクローンして、すべての例を取得、ビルド、実行することもできます。

- `S3Client` クラス

  `S3Client` ライブラリは、Amazon S3 とのやり取りを完全サポートするインターフェイスです。

  このセットの `list_buckets_disabling_dns_cache.cpp` の例は、特に Linux/Mac 上の CURL で動作するように調整されています (ただし、Windows で動作す

るように変更することもできます)。Windows を使用している場合は、ビルド前に `list_buckets_disabling_dns_cache.cpp` ファイルを削除してください。このファイルは Linux の curl HttpClient に依存しているためです。

`S3Client` を利用したコードは GitHub の s3 フォルダ にあります。このセットのコード例で使用されている関数の完全なリストについては、GitHub の Readme を参照してください。

このセットの `s3` コード例の一部については、このガイドでさらに詳しく説明しています。

- バケットの作成、一覧表示、削除
- オブジェクトに対するオペレーション – データオブジェクトのアップロードとダウンロード
- Amazon S3 アクセス許可の管理
- バケットポリシーを使用した Amazon S3 バケットへのアクセスの管理
- ウェブサイトとしての Amazon S3 バケットの設定

- `S3CrtClient` クラス

  `S3CrtClient` は SDK バージョン 1.9 で追加されました。`S3CrtClient` により、Amazon S3 の GET (ダウンロード) および PUT (アップロード) オペレーションのスループットが向上します。`S3CrtClient` は AWS 共通ランタイム (CRT) ライブラリの上位に実装されています。

  `S3CrtClient` を利用したコードは GitHub の `s3-crt` フォルダ にあります。このセットのコード例で使用されている関数の完全なリストについては、GitHub の Readme を参照してください。

  - Amazon S3 オペレーションでの `S3CrtClient` の使用

- `TransferManager` クラス

  `TransferManager` は、File Transfer Protocol (FTP)、File Transfer Protocol over SSL (FTPS)、または Secure Shell (SSH) File Transfer Protocol (SFTP) を介して Amazon S3 との間でファイルを直接転送できるフルマネージドサービスです。

  `TransferManager` を利用したコードは GitHub の `transfer-manager` フォルダ にあります。このセットのコード例で使用されている関数の完全なリストについては、GitHub の Readme を参照してください。

  - Amazon S3 オペレーションでの TransferManager の使用

バケットの作成、一覧表示、削除

Amazon Simple Storage Service (Amazon S3) 内のすべてのオブジェクトまたはファイルは、オブジェクトのフォルダを表すバケットに含まれています。各バケットの名前は AWS 内でグローバルに

一意である必要があります。詳細については、「Amazon Simple Storage Service ユーザーガイド」の「Amazon S3 バケットの使用」を参照してください。

前提条件

作業を始める前に「AWS SDK for C++ の開始方法」を読むことをお勧めします。

コード例をダウンロードし、「コード例の開始方法」の説明に従ってソリューションをビルドします。

例を実行するには、リクエストに使用するユーザープロファイルに、AWS のサービスとアクションに対する適切なアクセス許可が付与されている必要があります。詳細については、「AWS 認証情報の提供」を参照してください。

バケットの一覧表示

list_buckets の例を実行するには、コマンドプロンプトで、ビルドシステムによる実行可能ファイルの作成先フォルダに移動します。実行可能ファイルを run_list_buckets のように実行します (完全な実行可能ファイル名はオペレーティングシステムによって異なります)。出力には、アカウントにバケットがある場合は一覧表示され、バケットがない場合は空のリストが表示されます。

list_buckets.cpp には 2 つの方法があります。

- main() が ListBuckets() を呼び出す。
- ListBuckets() が SDK を使用してバケットをクエリする。

S3Client オブジェクトは SDK の ListBuckets() メソッドを呼び出します。成功すると、このメソッドは ListBucketOutcome オブジェクトを返します。このオブジェクトには ListBucketResult オブジェクトが含まれています。ListBucketResult オブジェクトは GetBuckets() メソッドを呼び出して、アカウント内の各 Amazon S3 バケットに関する情報を含む Bucket オブジェクトのリストを取得します。

コード

```
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;
}
```

GitHub で完全な list_buckets の例を参照してください。

バケットを作成する

create_bucket の例を実行するには、コマンドプロンプトで、ビルドシステムによる実行可能ファイルの作成先フォルダに移動します。実行可能ファイルを run_create_bucket のように実行します (完全な実行可能ファイル名はオペレーティングシステムによって異なります)。このコードはアカウントに空のバケットを作成し、そのリクエストの成功または失敗を表示します。

create_bucket.cpp には 2 つの方法があります。

- main() が CreateBucket() を呼び出す。main() で、enum を使用して、アカウントのリージョンに応じて AWS リージョン を変更する必要があります。アカウントのリージョンを確認するには、AWS マネジメントコンソール にログインし、右上隅に表示されるリージョン名を見つけます。
- CreateBucket() が SDK を使用してバケットを作成する。

S3Client オブジェクトは SDK の CreateBucket() メソッドの呼び出しで、バケット名を CreateBucketRequest で指定して渡します。デフォルトでは、バケットは us-east-1 (バージニア北部) リージョンに作成されます。リージョンが us-east-1 でない場合、バケットが指定リージョンに作成されるように、このコードはバケットの制約を設定します。

コード

```
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();
}
```

GitHub で完全な create_buckets の例を参照してください。

バケットを削除する

delete_bucket の例を実行するには、コマンドプロンプトで、ビルドシステムによる実行可能ファイルの作成先フォルダに移動します。実行可能ファイルを run_delete_bucket のように実行します (完全な実行可能ファイル名はオペレーティングシステムによって異なります)。このコードは、アカウント内の指定されたバケットを削除し、リクエストの成功または失敗を表示します。

delete_bucket.cpp には 2 つの方法があります。

- main() が DeleteBucket() を呼び出す。main() で、enum を使用して、アカウントのリージョンに応じて AWS リージョン を変更する必要があります。また、bucket_name を削除対象のバケットの名前に変更する必要もあります。
- DeleteBucket() が SDK を使用してバケットを削除する。

S3Client オブジェクトは、SDK の DeleteBucket() メソッドの使用時に、削除対象のバケットの名前を DeleteBucketRequest オブジェクトで指定して渡します。成功するには、バケットが空である必要があります。

コード

```
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();
}
```

GitHub で完全な delete_bucket の例を参照してください。

オブジェクトに対するオペレーション

Amazon S3 オブジェクトは、データの集合であるファイルを表します。すべてのオブジェクトがバケット内にある必要があります。

前提条件

作業を始める前に「AWS SDK for C++ の開始方法」を読むことをお勧めします。

コード例をダウンロードし、「コード例の開始方法」の説明に従ってソリューションをビルドします。

例を実行するには、リクエストに使用するユーザープロファイルに、AWS のサービスとアクションに対する適切なアクセス許可が付与されている必要があります。詳細については、「AWS 認証情報の提供」を参照してください。

バケットにファイルをアップロードする

S3Client オブジェクトの PutObject 関数の使用時に、バケット名、キー名、アップロードするファイルを指定します。Aws::FStream は、ローカルファイルの内容をバケットにアップロードするために使用されます。バケットが存在している必要があり、存在しない場合はエラーが発生します。

オブジェクトを非同期でアップロードする例については、「AWS SDK for C++ を使用した非同期プログラミング」を参照してください。

Code

```
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();
}
```

GitHub で完全な例をご覧ください。

バケットに文字列をアップロードする

S3Client オブジェクトの PutObject 関数の使用時に、バケット名、キー名、アップロードするファイルを指定します。バケットが存在している必要があり、存在しない場合はエラーが発生します。この例では、前回と異なり、Aws::StringStream を使用してインメモリの文字列データオブジェクトをバケットに直接アップロードします。

オブジェクトを非同期でアップロードする例については、「AWS SDK for C++ を使用した非同期プログラミング」を参照してください。

Code

```
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();
}
```

GitHub で完全な例をご覧ください。

オブジェクトのリスト化

バケット内のオブジェクトのリストを取得するには、S3Client オブジェクトの ListObjects 関数を使用します。リストを取得するバケットの名前を ListObjectsRequest で指定して渡します。

ListObjects 関数は ListObjectsOutcome オブジェクトを返します。このオブジェクトを使用して、Object インスタンスの形式でオブジェクトのリストを取得できます。

Code

```
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;
}
```

GitHub で完全な例をご覧ください。

オブジェクトのダウンロード

S3Client オブジェクトの GetObject 関数の使用時に、バケット名とオブジェクトキーを GetObjectRequest で指定して渡します。GetObject は GetObjectOutcome オブジェクトを返します。このオブジェクトは GetObjectResult と S3Error 構成されています。GetObjectResult を使用して、S3 オブジェクトのデータにアクセスできます。

次の例は、Amazon S3 からオブジェクトをダウンロードします。オブジェクトの内容はローカル変数に保存され、最初の 1 行がコンソールに出力されます。

Code

```
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();
}
```

GitHub で完全な例をご覧ください。

オブジェクトの削除

S3Client オブジェクトの DeleteObject 関数の使用時に、ダウンロード対象のバケットとオブジェクトの名前を DeleteObjectRequest で指定して渡します。指定されたバケットとオブジェクトキーが存在している必要があり、存在しない場合エラーが発生します。

Code

```
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();
}
```

GitHub で完全な例をご覧ください。

Amazon S3 アクセス許可の管理

Amazon S3 バケットまたはオブジェクトのアクセス許可は、アクセスコントロールリスト (ACL) で定義されます。ACL では、バケット/オブジェクトの所有者、およびアクセス許可付与設定の一覧を指定します。アクセス許可付与設定では、ユーザー (被付与者) と、バケット/オブジェクトに対してそのユーザーに付与するアクセス許可 (READ や WRITE など) を指定します。

前提条件

作業を始める前に「AWS SDK for C++ の開始方法」を読むことをお勧めします。

コード例をダウンロードし、「コード例の開始方法」の説明に従ってソリューションをビルドします。

例を実行するには、リクエストに使用するユーザープロファイルに、AWS のサービスとアクションに対する適切なアクセス許可が付与されている必要があります。詳細については、「AWS 認証情報の提供」を参照してください。

オブジェクトのアクセスコントロールリストを管理する

オブジェクトのアクセスコントロールリストは S3Client メソッド GetObjectAcl を呼び出すことで取得できます。メソッドは、オブジェクトとそのバケットの名前を受け入れます。戻り値には ACL の Owner、および Grants の一覧が含まれます。

```
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";
    }
}
```

ACL は、新規に作成するか、既存の ACL のアクセス許可付与設定を編集することで変更できます。更新された ACL は、PutObjectAcl メソッドに渡すことで新しい現在の ACL になります。

次のコードでは、GetObjectAcl で取得した ACL に新しいアクセス許可付与設定を追加しています。ユーザー (被付与者) には、オブジェクトに対する READ アクセス許可が付与されます。変更された ACL は PutObjectAcl に渡され、新たな ACL として適用されます。

```
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;
}
```

GitHub で完全な例をご覧ください。

バケットのアクセスコントロールリストを管理する

多くの場合、バケットに対するアクセス許可を設定するには、バケットポリシーを定義することをお勧めします。ただし、希望すればバケットに対してアクセスコントロールリストを使用することも可能です。

バケットに対するアクセスコントロールリストの管理は、オブジェクトに使用されるものと同じです。GetBucketAcl メソッドでバケットの現在の ACL を取得し、PutBucketAcl メソッドで新しい ACL をバケットに適用します。

次のコードでは、バケットに対する ACL の取得および設定方法を示しています。

```
//! 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;
}
```

GitHub で完全な例をご覧ください。

バケットポリシーを使用した Amazon S3 バケットへのアクセスの管理

バケットポリシーを設定、取得、または削除して、Amazon S3 バケットへのアクセスを管理できます。

前提条件

作業を始める前に「AWS SDK for C++ の開始方法」を読むことをお勧めします。

コード例をダウンロードし、「コード例の開始方法」の説明に従ってソリューションをビルドします。

例を実行するには、リクエストに使用するユーザープロファイルに、AWS のサービスとアクションに対する適切なアクセス許可が付与されている必要があります。詳細については、「AWS 認証情報の提供」を参照してください。

バケットポリシーの設定

特定の S3 バケットに対するバケットポリシーを設定するには、S3Client の PutBucketPolicy 関数の呼び出しで、バケット名とポリシーの JSON 表現を PutBucketPolicyRequest で指定して渡します。

コード

```
//! 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"
            "}";
}
```

```
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 ユーティリティクラスを使用すると、有効な JSON オブジェクトをコンストラクトして PutBucketPolicy に渡すことができます。

GitHub で完全な例をご覧ください。

バケットポリシーの取得

Amazon S3 バケットのポリシーを取得するには、S3Client の GetBucketPolicy 関数の呼び出しで、バケット名を GetBucketPolicyRequest で指定して渡します。

Code

```
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();
}
```

GitHub で完全な例をご覧ください。

バケットポリシーの削除

バケットポリシーを削除するには、S3Client の DeleteBucketPolicy 関数の呼び出しで、バケット名を DeleteBucketPolicyRequest で指定して渡します。

コード

```
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();
}
```

バケットにポリシーが存在しない場合でも、この関数は成功します。指定した名前のバケットが存在していないか、バケットへのアクセス権がない場合は、AmazonServiceException がスローされます。

GitHub で完全な例をご覧ください。

詳細

- 「Amazon Simple Storage Service API リファレンス」の「PutBucketPolicy」
- 「Amazon Simple Storage Service API リファレンス」の「GetBucketPolicy」
- 「Amazon Simple Storage Service API リファレンス」の「DeleteBucketPolicy」
- 「Amazon Simple Storage Service ユーザーガイド」の「アクセスポリシー言語の概要」
- 「Amazon Simple Storage Service ユーザーガイド」の「バケットポリシーの例」

ウェブサイトとしての Amazon S3 バケットの設定

Amazon S3 バケットをウェブサイトとして動作するように設定できます。これを行うには、ウェブサイト設定をセットする必要があります。

前提条件

作業を始める前に「AWS SDK for C++ の開始方法」を読むことをお勧めします。

コード例をダウンロードし、「コード例の開始方法」の説明に従ってソリューションをビルドします。

例を実行するには、リクエストに使用するユーザープロファイルに、AWS のサービスとアクションに対する適切なアクセス許可が付与されている必要があります。詳細については、「AWS 認証情報の提供」を参照してください。

バケットのウェブサイト設定をセットする

Amazon S3 バケットのウェブサイト設定を行うには、S3Client の PutBucketWebsite 関数の呼び出しで、バケット名とそのウェブサイト設定を PutBucketWebsiteRequest オブジェクトで指定して渡します。ウェブサイト設定は WebsiteConfiguration オブジェクトに指定されている必要があります。

インデックスドキュメントの設定は必要ですが、他のすべてのパラメータはオプションです。

コード

```
bool AwsDoc::S3::putWebsiteConfig(const Aws::String &bucketName,
                                  const Aws::String &indexPage, const Aws::String
 &errorPage,
                                  const Aws::S3::S3ClientConfiguration &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::IndexDocument indexDocument;
    indexDocument.SetSuffix(indexPage);

    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**
>
> ウェブサイト設定をセットしても、バケットのアクセス権限は変更されません。ウェブ上でファイルが表示されるようにするには、バケットのファイルにパブリック読み取りアクセスを許可するバケットポリシーも設定する必要があります。詳細については、「バケットポリシーを使用した Amazon S3 バケットへのアクセス管理」を参照してください。

GitHub で完全な例をご覧ください。

バケットのウェブサイト設定を取得する

Amazon S3 バケットのウェブサイト設定を取得するには、S3Client の GetBucketWebsite 関数の呼び出しで、バケット名を GetBucketWebsiteRequest で指定して渡します。

この設定は、結果オブジェクトに入れて GetBucketWebsiteResult オブジェクトとして返されます。バケットのウェブサイト設定がない場合は、null が返されます。

Code

```
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();
}
```

GitHub で完全な例をご覧ください。

バケットのウェブサイト設定を削除する

Amazon S3 バケットのウェブサイト設定を削除するには、S3Client の DeleteBucketWebsite 関数の呼び出しで、設定を削除するバケットの名前を DeleteBucketWebsiteRequest で指定して渡します。

Code

```
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();
}
```

GitHub で完全な例をご覧ください。

詳細情報

- 「Amazon Simple Storage Service API リファレンス」の「PUT Bucket website」
- 「Amazon Simple Storage Service API リファレンス」の「GET Bucket website」
- 「Amazon Simple Storage Service API リファレンス」の「DELETE Bucket website」

Amazon S3 オペレーションでの TransferManager の使用

AWS SDK for C++ `TransferManager` クラスを使用して、ローカル環境から Amazon S3 へファイルを確実に転送し、S3 の複数の場所間でオブジェクトをコピーします。`TransferManager` は、転送の進行状況を取得し、アップロードとダウンロードの一時停止/再開を行うことができます。

> **Note**
>
> 不完全または部分的なアップロードに対して課金されないように、Amazon S3 バケットで AbortIncompleteMultipartUpload ライフサイクルルールを有効にすることをお勧めします。このルールは、開始後、指定された日数内に完了しないマルチパートアップロードを中止するよう Amazon S3 に指示します。設定した時間制限を超えると、Amazon S3 はアップロードを中止して、不完全なアップロードデータを削除します。
> 詳細については、Amazon S3 ユーザーガイドの「バケットのライフサイクル設定の指定」を参照してください。

前提条件

作業を始める前に「AWS SDK for C++ の開始方法」を読むことをお勧めします。

コード例をダウンロードし、「コード例の開始方法」の説明に従ってソリューションをビルドします。

例を実行するには、リクエストに使用するユーザープロファイルに、AWS のサービスとアクションに対する適切なアクセス許可が付与されている必要があります。詳細については、「AWS 認証情報の提供」を参照してください。

`TransferManager` を使用したオブジェクトのアップロードとダウンロード

この例では、`TransferManager` を使用して、大きなオブジェクトをメモリ内で転送する方法を示します。`UploadFile` および `DownloadFile` メソッドはどちらも非同期で呼び出され、リク

エストのステータス管理用に TransferHandle を返します。アップロードされたオブジェクトが bufferSize より大きい場合、マルチパートアップロードが実行されます。bufferSize はデフォルトで 5MB ですが、このサイズは TransferManagerConfiguration から設定できます。

```
        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;
```

GitHub で完全な例をご覧ください。

Amazon S3 オペレーションでの **S3CrtClient** の使用

`S3CrtClient` クラスは AWS SDK for C++ バージョン 1.9 で使用でき、Amazon S3 との間で大容量データファイルをアップロードおよびダウンロードする際のスループットが向上します。このリリースの改善点の詳細については、「AWS SDK for C++ v1.9 での Amazon S3 スループットの向上」を参照してください。

`S3CrtClient` は AWS 共通ランタイム (CRT) ライブラリの上位に実装されています。

> **Note**
>
> 不完全または部分的なアップロードに対して課金されないように、Amazon S3 バケットで AbortIncompleteMultipartUpload ライフサイクルルールを有効にすることをお勧めします。このルールは、開始後、指定された日数内に完了しないマルチパートアップロードを中止するよう Amazon S3 に指示します。設定した時間制限を超えると、Amazon S3 はアップロードを中止して、不完全なアップロードデータを削除します。
> 詳細については、Amazon S3 ユーザーガイドの「バケットのライフサイクル設定の指定」を参照してください。

前提条件

作業を始める前に「AWS SDK for C++ の開始方法」を読むことをお勧めします。

コード例をダウンロードし、「コード例の開始方法」の説明に従ってソリューションをビルドします。

例を実行するには、リクエストに使用するユーザープロファイルに、AWS のサービスとアクションに対する適切なアクセス許可が付与されている必要があります。詳細については、「AWS 認証情報の提供」を参照してください。

S3CrtClient を使用したオブジェクトのアップロードとダウンロード

この例では S3CrtClient の使用方法を示します。この例では、バケットを作成し、オブジェクトをアップロードし、そのオブジェクトをダウンロードした後、ファイルとバケットを削除します。PUT オペレーションはマルチパートアップロードになります。GET オペレーションは複数の「範囲指定」 GET リクエストになります。マルチパートアップロードの詳細については、「Amazon S3 ユーザーガイド」の「マルチパートアップロードを使用したオブジェクトのアップロードとコピー」を参照してください。

この例で使用されるデータファイル ny.json は、マルチパートアップロードとしてアップロードされます。この動作は、プログラムの正常な実行後にデバッグログを表示することで確認できます。

アップロードが失敗した場合、基礎となる CRT ライブラリで AbortMultipartUpload が発行されて、既にアップロードされたパートがクリーンアップされます。ただし、すべての失敗が内部で処理できるわけではありません (ネットワークケーブルが抜けている場合など)。アカウント内に部分的にアップロードされたデータが残らないように、Amazon S3 バケットにライフサイクルルールを設定することをお勧めします (部分的にアップロードされたデータにも料金が発生します)。ライフサイクルルールの設定方法については、「Amazon S3 のコスト削減のための不完全なマルチパートアップロードの検出と削除」を参照してください。

デバッグログを使用したマルチパートアップロードの詳細の調査

1. main() 内には、コードの更新手順が記載された「TODO」コメントがあることに注意してください。
   a. file_name: コードコメント内のリンクからサンプルデータファイル ny.json をダウンロードするか、自前の大きなデータファイルを使用します。
   b. region: 列挙値を使用して、region 変数をアカウントの AWS リージョン に更新します。自分のアカウントのリージョンを見つけるには、AWS マネジメントコンソール にログインし、右上隅にあるリージョンを確認します。
2. 例をビルドします。
3. 変数 file_name で指定されたファイルを実行可能ファイルのあるフォルダにコピーし、s3-crt-demo 実行可能ファイルを実行します。
4. 実行可能ファイルのあるフォルダで、最新の .log ファイルを見つけます。

5. ログファイルを開き、[search] (検索) を選択して、「**partNumber**」と入力します。
6. ログには次のようなエントリが含まれています。アップロードされたファイルのパートごとに partNumber と uploadId が指定されています。

```
PUT /my-object
partNumber=1&uploadId=gsk8vDbmnlA5EseDo._LDEgq22Qmt0SeuszYxMsZ9ABt503VqDIFOP8
content-length:8388608 host:my-bucketasdfasdf.s3.us-
east-2.amazonaws.com x-amz-content-sha256:UNSIGNED-PAYLOAD

and

PUT /my-object
partNumber=2&uploadId=gsk8vDbmnlA5EseDo._LDEgq22Qmt0SeuszYxMsZ9ABt503VqDIFOP8
content-length:8388608 host:my-bucketasdfasdf.s3.us-
east-2.amazonaws.com x-amz-content-sha256:UNSIGNED-PAYLOAD
```

GitHub で完全な例をご覧ください。

AWS SDK for C++ を使用した Amazon SQS のコード例

Amazon Simple Queue Service (Amazon SQS) は、フルマネージドのメッセージキューイングサービスであり、マイクロサービス、分散システム、およびサーバーレスアプリケーションの疎結合化とスケーリングを容易にします。以下の例のように、AWS SDK for C++ を使用して Amazon SQS をプログラミングできます。

> **Note**
>
> このガイドには、特定の手法を示すために必要最低限のコードのみを掲載していますが、完全なコード例は GitHub で入手できます。GitHub では、ソースファイル単体をダウンロードすることも、リポジトリをローカルにクローンして、すべての例を取得、ビルド、実行することもできます。

トピック

- Amazon SQS メッセージキューの使用
- Amazon SQS メッセージの送信、受信、削除
- Amazon SQS メッセージキューでロングポーリングを有効にする

- Amazon SQS での可視性タイムアウトの設定
- Amazon SQS でデッドレターキューを使用する

Amazon SQS メッセージキューの使用

メッセージキュー は、Amazon SQS でメッセージを確実に送信するために使用する論理コンテナです。キューには、標準と先入れ先出し (FIFO) の 2 種類があります。キューの詳細とキュータイプ間の相違点については、「Amazon Simple Queue Service デベロッパーガイド」を参照してください。

これらの C++ の例では、AWS SDK for C++ を使用して Amazon SQS キューの作成、一覧取得、削除、URL 取得を行う方法を示しています。

前提条件

作業を始める前に「AWS SDK for C++ の開始方法」を読むことをお勧めします。

コード例をダウンロードし、「コード例の開始方法」の説明に従ってソリューションをビルドします。

例を実行するには、リクエストに使用するユーザープロファイルに、AWS のサービスとアクションに対する適切なアクセス許可が付与されている必要があります。詳細については、「AWS 認証情報の提供」を参照してください。

キューの作成

SQSClient クラスの `CreateQueue` メンバー関数の使用時に、キューのパラメータを CreateQueueRequest オブジェクトで指定して渡します。

含まれるもの:

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/CreateQueueRequest.h>
#include <iostream>
```

コード

```
    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;
    }
```

完全な例をご覧ください。

キューを一覧表示する

アカウントの Amazon SQS キューを一覧表示するには、SQSClient クラスの ListQueues メンバー関数の呼び出しで、ListQueuesRequest オブジェクトを渡します。

含まれるもの:

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/ListQueuesRequest.h>
#include <iostream>
```

コード

```
    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;
    }
```

完全な例をご覧ください。

キューの URL を取得する

既存の Amazon SQS キューの URL を取得するには、SQSClient クラスの `GetQueueUrl` メンバー関数を呼び出します。

含まれるもの:

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/GetQueueUrlRequest.h>
#include <iostream>
```

コード

```
    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;
    }
```

完全な例をご覧ください。

キューの削除

SQSClient クラスの DeleteQueue メンバー関数に URL を渡します。

含まれるもの:

```
#include <aws/core/Aws.h>
#include <aws/core/client/DefaultRetryStrategy.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/DeleteQueueRequest.h>
#include <iostream>
```

コード

```
    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;
    }
```

完全な例をご覧ください。

詳細

- 「Amazon Simple Queue Service デベロッパーガイド」の「Amazon SQS キューの仕組み」

- 「Amazon Simple Queue Service API リファレンス」の「CreateQueue」
- 「Amazon Simple Queue Service API リファレンス」の「GetQueueUrl」
- 「Amazon Simple Queue Service API リファレンス」の「ListQueues」
- 「Amazon Simple Queue Service API リファレンス」の「DeleteQueues」

Amazon SQS メッセージの送信、受信、削除

メッセージは、常に SQS キューを使用して提供されます。これらの C++ の例では、AWS SDK for C++ を使用して Amazon SQS キューへのメッセージの送信、およびキューからのメッセージの受信、削除を行う方法を示しています。

前提条件

作業を始める前に「AWS SDK for C++ の開始方法」を読むことをお勧めします。

コード例をダウンロードし、「コード例の開始方法」の説明に従ってソリューションをビルドします。

例を実行するには、リクエストに使用するユーザープロファイルに、AWS のサービスとアクションに対する適切なアクセス許可が付与されている必要があります。詳細については、「AWS 認証情報の提供」を参照してください。

メッセージの送信

SQSClient クラスの SendMessage メンバー関数を呼び出すことで、Amazon SQS キューに 1 つのメッセージを追加できます。SendMessage には、キューの URL、メッセージ本文、オプションで遅延値 (秒単位) を SendMessageRequest オブジェクトで指定して渡します。

含まれるもの:

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/SendMessageRequest.h>
#include <iostream>
```

コード

```
    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;
    }
```

完全な例をご覧ください。

メッセージを受信する

キューに現在含まれているメッセージを取得するには、SQSClient クラスの ReceiveMessage メソッドの呼び出しで、キューの URL を渡します。メッセージは、Message オブジェクトのリストとして返されます。

含まれるもの:

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/ReceiveMessageRequest.h>
#include <iostream>
```

コード

```
    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;
    }
```

完全な例をご覧ください。

受信後にメッセージを削除する

メッセージを受信し、その内容を処理した後で、メッセージをキューから削除するには、メッセージの受信ハンドルとキュー URL を SQSClient クラスの DeleteMessage メンバー関数に送信します。

含まれるもの:

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/DeleteMessageRequest.h>
#include <iostream>
```

コード

```
    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;
    }
```

完全な例をご覧ください。

詳細

- 「Amazon Simple Queue Service デベロッパーガイド」の「Amazon SQS キューの仕組み」
- 「Amazon Simple Queue Service API リファレンス」の「SendMessage」
- 「Amazon Simple Queue Service API リファレンス」の「SendMessageBatch」
- 「Amazon Simple Queue Service API リファレンス」の「ReceiveMessage」
- 「Amazon Simple Queue Service API リファレンス」の「DeleteMessage」

Amazon SQS メッセージキューでロングポーリングを有効にする

Amazon SQS はデフォルトでショートポーリングを使用して、サーバーのサブセットだけに対して (重み付けされたランダム分散に基づいて) クエリを実行し、レスポンスに含めることができるメッセージがあるかどうかを調べます。

ロングポーリングは、Amazon SQS キューに送信された ReceiveMessage リクエストに応答して返信するメッセージがない場合に、偽の空のレスポンスを排除して空のレスポンスの数を減らすことで、Amazon SQS の使用コストを削減します。1~20 秒でロングポーリング頻度を設定できます。

前提条件

作業を始める前に「AWS SDK for C++ の開始方法」を読むことをお勧めします。

コード例をダウンロードし、「コード例の開始方法」の説明に従ってソリューションをビルドします。

例を実行するには、リクエストに使用するユーザープロファイルに、AWS のサービスとアクションに対する適切なアクセス許可が付与されている必要があります。詳細については、「AWS 認証情報の提供」を参照してください。

キューの作成時にロングポーリングを有効にする

Amazon SQS キューの作成時にロングポーリングを有効にするには、SQSClient クラスの `CreateQueue` メンバー関数を呼び出す前に、CreateQueueRequest オブジェクトの `ReceiveMessageWaitTimeSeconds` 属性を設定します。

含まれるもの:

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/CreateQueueRequest.h>
#include <iostream>
```

コード

```
    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;
    }
```

完全な例をご覧ください。

既存のキューでロングポーリングを有効にする

キュー作成時にロングポーリングを有効にすることに加えて、SQSClient クラスの `SetQueueAttributes` メンバー関数を呼び出す前に SetQueueAttributesRequest で `ReceiveMessageWaitTimeSeconds` を設定することで、既存のキューでも有効にすることができます。

含まれるもの:

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/SetQueueAttributesRequest.h>
#include <iostream>
```

コード

```
    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;
    }
```

完全な例をご覧ください。

メッセージ受信時のロングポーリングを有効化する

メッセージの受信時にロングポーリングを有効にするには、SQSClient クラスの ReceiveMessage メンバー関数に渡す ReceiveMessageRequest で待機時間 (秒単位) を設定します。

> **Note**
>
> 次のポーリングイベントの待機中に ReceiveMessage リクエストがタイムアウトしないように、AWS クライアントのリクエストのタイムアウトがロングポーリングの最大値 (20 秒) より長いことを確認します。

含まれるもの:

```
#include <aws/core/Aws.h>
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/ReceiveMessageRequest.h>
```

コード

```
    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;
```

```
    }
```

完全な例をご覧ください。

詳細

- 「Amazon Simple Queue Service デベロッパーガイド」の「Amazon SQS ロングポーリング」
- 「Amazon Simple Queue Service API リファレンス」の「CreateQueue」
- 「Amazon Simple Queue Service API リファレンス」の「ReceiveMessage」
- 「Amazon Simple Queue Service API リファレンス」の「SetQueueAttributes 」

Amazon SQS での可視性タイムアウトの設定

Amazon SQS でメッセージを受信すると、確実な受信のために、そのメッセージは削除されるまでキューに残ります。削除されなかった受信メッセージは、指定された可視性タイムアウトの後に以降のリクエストで使用でき、メッセージが処理および削除される前に複数回受信することを防ぎます。

標準キューを使用している場合、可視性タイムアウトはメッセージを 2 回受信しない保証にはなりません。標準キューを使用している場合は、同じメッセージが複数回配信されるケースをコードが処理できることを確認してください。

前提条件

作業を始める前に「AWS SDK for C++ の開始方法」を読むことをお勧めします。

コード例をダウンロードし、「コード例の開始方法」の説明に従ってソリューションをビルドします。

例を実行するには、リクエストに使用するユーザープロファイルに、AWS のサービスとアクションに対する適切なアクセス許可が付与されている必要があります。詳細については、「AWS 認証情報の提供」を参照してください。

メッセージ受信時の可視性タイムアウトを設定する

メッセージを受信したら、その受信ハンドルを ChangeMessageVisibilityRequest に指定し、そのリクエストを SQSClient クラスの `ChangeMessageVisibility` メンバー関数に渡すことで、可視性タイムアウトを変更できます。

含まれるもの:

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

```
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/ChangeMessageVisibilityRequest.h>
#include <aws/sqs/model/ReceiveMessageRequest.h>
#include <iostream>
```

コード

```
    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;
    }
```

完全な例をご覧ください。

詳細

- Amazon Simple Queue Service デベロッパーガイドの「可視性タイムアウト」
- 「Amazon Simple Queue Service API リファレンス」の「SetQueueAttributes 」
- 「Amazon Simple Queue Service API リファレンス」の「GetQueueAttributes」
- 「Amazon Simple Queue Service API リファレンス」の「ReceiveMessage」
- 「Amazon Simple Queue Service API リファレンス」の「ChangeMessageVisibility」
- 「Amazon Simple Queue Service API リファレンス」の「ChangeMessageVisibilityBatch」

Amazon SQS でデッドレターキューを使用する

Amazon SQS では、デッドレターキューがサポートされます。デッドレターキューは、正常に処理できないメッセージの送信先として他のキューが使用できるキューです。これらのメッセージは、処理が成功しなかった理由を判断するためにデッドレターキューに分離できます。

デッドレターキューを作成するには、まず再処理ポリシーを作成し、次にキューの属性でそのポリシーを設定します。

> **Important**
>
> デッドレターキューは、ソースキューと同じタイプのキュー (FIFO または標準) である必要があります。また、ソースキューと同じ AWS アカウント および AWS リージョン を使用して作成する必要があります。

前提条件

作業を始める前に「AWS SDK for C++ の開始方法」を読むことをお勧めします。

コード例をダウンロードし、「コード例の開始方法」の説明に従ってソリューションをビルドします。

例を実行するには、リクエストに使用するユーザープロファイルに、AWS のサービスとアクションに対する適切なアクセス許可が付与されている必要があります。詳細については、「AWS 認証情報の提供」を参照してください。

再処理ポリシーを作成する

再処理ポリシーは JSON 形式で指定します。その作成には、AWS SDK for C++ で提供されている JSON ユーティリティクラスを使用できます。

ここで示しているのは、再処理ポリシーを作成する関数の一例です。この関数では、対象となるデッドレターキューの ARN と、メッセージが受信されても処理されなかった場合に、デッドレターキューへ送られるまでの最大試行回数を指定しています。

含まれるもの:

```
#include <aws/core/Aws.h>
#include <aws/core/utils/json/JsonSerializer.h>
```

コード

```
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();
}
```

完全な例をご覧ください。

ソースキューに再処理ポリシーを設定する

デッドレターキューの設定を完了するには、SQSClient クラスの SetQueueAttributes メンバー関数の呼び出しで、JSON 形式の再処理ポリシーを RedrivePolicy 属性として設定した SetQueueAttributesRequest オブジェクトを渡します。

含まれるもの:

```
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/SetQueueAttributesRequest.h>
#include <iostream>
```

コード

```
    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;
    }
```

完全な例をご覧ください。

詳細

- 「Amazon Simple Queue Service デベロッパーガイド」の「Amazon SQS デッドレターキューの使用」
- 「Amazon Simple Queue Service API リファレンス」の「SetQueueAttributes 」

SDK for C++ のコード例

このトピックのコード例は、 AWS SDK for C++ で を使用する方法を示しています AWS。

基本 は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

シナリオは、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

一部のサービスには、サービス固有のライブラリや関数の活用方法を示す追加のカテゴリ例が含まれています。

サービス

- SDK for C++ を使用した ACM の例
- SDK for C++ を使用した API Gateway の例
- SDK for C++ を使用した Aurora の例
- SDK for C++ を使用した自動スケーリングの例
- SDK for C++ を使用した CloudTrail の例
- SDK for C++ を使用した CloudWatch の例
- SDK for C++ を使用した CloudWatch Logs の例
- SDK for C++ を使用した CodeBuild の例
- SDK for C++ を使用する Amazon Cognito ID プロバイダーの例
- SDK for C++ を使用した DynamoDB の例
- SDK for C++ を使用した Amazon EC2 の例
- SDK for C++ を使用した EventBridge の例
- AWS Glue SDK for C++ を使用した例
- SDK for C++ を使用した HealthImaging の例
- SDK for C++ を使用する IAM の例
- AWS IoT SDK for C++ を使用した例

- AWS IoT data SDK for C++ を使用した例
- SDK for C++ を使用した Lambda の例
- SDK for C++ を使用した MediaConvert の例
- SDK for C++ を使用した Amazon RDS の例
- SDK for C++ を使用した Amazon RDS Data Service の例
- SDK for C++ を使用した Amazon Rekognition の例
- SDK for C++ を使用した Amazon S3 の例
- SDK for C++ を使用した Secrets Manager の例
- SDK for C++ を使用した Amazon SES の例
- SDK for C++ を使用した Amazon SNS の例
- SDK for C++ を使用した Amazon SQS の例
- AWS STS SDK for C++ を使用した例
- SDK for C++ を使用した Amazon Transcribe Streaming の例

SDK for C++ を使用した ACM の例

次のコード例は、ACM AWS SDK for C++ で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- アクション

アクション

`AddTagsToCertificate`

次のコード例は、`AddTagsToCertificate` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「AddTagsToCertificate」を参照してください。

`DeleteCertificate`

次のコード例は、`DeleteCertificate` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteCertificate」を参照してください。

`DescribeCertificate`

次のコード例は、`DescribeCertificate` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、「AWS SDK for C++ API リファレンス」の「DescribeCertificate」を参照してください。

`ExportCertificate`

次のコード例は、`ExportCertificate` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスのExportCertificate」を参照してください。

GetCertificate

次のコード例は、GetCertificate を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「GetCertificate」を参照してください。

`ImportCertificate`

次のコード例は、`ImportCertificate` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「ImportCertificate」を参照してください。

`ListCertificates`

次のコード例は、`ListCertificates` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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);

       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 の詳細については、AWS SDK for C++ API リファレンスの「ListCertificates」を参照してください。

`ListTagsForCertificate`

次のコード例は、`ListTagsForCertificate` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「ListTagsForCertificate」を参照してください。

`RemoveTagsFromCertificate`

次のコード例は、`RemoveTagsFromCertificate` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「RemoveTagsFromCertificate」を参照してください。

`RenewCertificate`

次のコード例は、`RenewCertificate` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「RenewCertificate」を参照してください。

`RequestCertificate`

次のコード例は、`RequestCertificate` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「RequestCertificate」を参照してください。

`ResendValidationEmail`

次のコード例は、`ResendValidationEmail` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「ResendValidationEmail」を参照してください。

UpdateCertificateOptions

次のコード例は、UpdateCertificateOptions を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「UpdateCertificateOptions」を参照してください。

SDK for C++ を使用した API Gateway の例

次のコード例は、 API Gateway AWS SDK for C++ で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

シナリオは、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- シナリオ

シナリオ

サーバーレスアプリケーションを作成して写真を管理する

次のコード例では、ユーザーがラベルを使用して写真を管理できるサーバーレスアプリケーションを作成する方法について示しています。

SDK for C++

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードと設定および実行の手順については、GitHub で完全な例を参照してください。

この例のソースについて詳しくは、AWS コミュニティでブログ投稿を参照してください。

この例で使用されているサービス

- API ゲートウェイ
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

SDK for C++ を使用した Aurora の例

次のコード例は、Aurora AWS SDK for C++ で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

基本 は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

シナリオは、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- はじめに
- 基本
- アクション
- シナリオ

はじめに

Hello Aurora

次のコード例は、Aurora の使用を開始する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

CMakeLists.txt CMake ファイルのコード。

```
# 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 ソースファイルのコード。

```
#include <aws/core/Aws.h>
#include <aws/rds/RDSClient.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::RDSClient 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());
                }
                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 の詳細については、AWS SDK for C++ API リファレンスの「DescribeDBClusters」を参照してください。

基本

基本を学ぶ

次のコード例は、以下の操作方法を示しています。

- カスタム Aurora DB クラスターパラメータグループを作成し、パラメータ値を設定します。
- パラメータグループを使用する DB クラスターを作成する
- データベースを含む DB インスタンスを作成します。
- DB クラスターのスナップショットを作成して、リソースをクリーンアップします。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 ofter "
                      << 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 ofter "
                      << 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 ofter "
                         << 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.
/*!
 \sa 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.
/*!
 \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;
}

//! 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 ofter " << 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 の詳細については、「AWS SDK for C++ API リファレンス」の以下のトピックを参照してください。
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - CreateDBClusterSnapshot
  - CreateDBInstance
  - DeleteDBCluster
  - DeleteDBClusterParameterGroup
  - DeleteDBInstance
  - DescribeDBClusterParameterGroups
  - DescribeDBClusterParameters
  - DescribeDBClusterSnapshots
  - DescribeDBClusters
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeOrderableDBInstanceOptions
  - ModifyDBClusterParameterGroup

アクション

`CreateDBCluster`

次のコード例は、`CreateDBCluster` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「CreateDBCluster」を参照してください。

`CreateDBClusterParameterGroup`

次のコード例は、`CreateDBClusterParameterGroup` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「CreateDBClusterParameterGroup」を参照してください。

`CreateDBClusterSnapshot`

次のコード例は、`CreateDBClusterSnapshot` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、「AWS SDK for C++ API リファレンス」の「CreateDBClusterSnapshot」を参照してください。

`CreateDBInstance`

次のコード例は、`CreateDBInstance` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「CreateDBInstance」を参照してください。

DeleteDBCluster

次のコード例は、DeleteDBCluster を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteDBCluster」を参照してください。

DeleteDBClusterParameterGroup

次のコード例は、DeleteDBClusterParameterGroup を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteDBClusterParameterGroup」を参照してください。

`DeleteDBInstance`

次のコード例は、`DeleteDBInstance` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteDBInstance」を参照してください。

DescribeDBClusterParameterGroups

次のコード例は、DescribeDBClusterParameterGroups を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「DescribeDBClusterParameterGroups」を参照してください。

`DescribeDBClusterParameters`

次のコード例は、`DescribeDBClusterParameters` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「DescribeDBClusterParameters」を参照してください。

DescribeDBClusterSnapshots

次のコード例は、DescribeDBClusterSnapshots を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「DescribeDBClusterSnapshots」を参照してください。

DescribeDBClusters

次のコード例は、DescribeDBClusters を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「DescribeDBClusters」を参照してください。

`DescribeDBEngineVersions`

次のコード例は、`DescribeDBEngineVersions` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「DescribeDBEngineVersions」を参照してください。

`DescribeDBInstances`

次のコード例は、`DescribeDBInstances` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 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;
}
```

- API の詳細については、AWS SDK for C++ API リファレンスの「DescribeDBInstances」を参照してください。

`DescribeOrderableDBInstanceOptions`

次のコード例は、`DescribeOrderableDBInstanceOptions` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「DescribeOrderableDBInstanceOptions」を参照してください。

`ModifyDBClusterParameterGroup`

次のコード例は、`ModifyDBClusterParameterGroup` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、「AWS SDK for C++ API リファレンス」の「ModifyDBClusterParameterGroup」を参照してください。

シナリオ

Aurora Serverless 作業項目トラッカーの作成

次のコード例は、Amazon Aurora Serverless データベースの作業項目を追跡し、Amazon Simple Email Service (Amazon SES) を使用してレポートを送信するウェブアプリケーションを作成する方法を示しています。

SDK for C++

Amazon Aurora Serverless データベースに保存されている作業項目を追跡して報告するウェブアプリケーションを作成する方法を説明します。

Amazon Aurora Serverless データをクエリし、React アプリケーションで使用するための C++ REST API の完全なソースコードと設定方法については、GitHub にある完全な例を参照してください。

この例で使用されているサービス

- Aurora

- Amazon RDS
- Amazon RDS データサービス
- Amazon SES

SDK for C++ を使用した自動スケーリングの例

次のコード例は、Auto Scaling AWS SDK for C++ で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

基本 は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- はじめに
- 基本
- アクション

はじめに

こんにちは、Auto Scaling

次のコード例は、Auto Scaling の使用を開始する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

CMakeLists.txt CMake ファイルのコード。

```
# 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 ソースファイルのコード。

```
#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 の詳細については、AWS SDK for C++ API リファレンスの「DescribeAutoScalingGroups」を参照してください。

基本

基本を学ぶ

次のコード例は、以下の操作方法を示しています。

- 起動テンプレートとアベイラビリティーゾーンを使用して、Amazon EC2 Auto Scaling グループを作成し、実行中のインスタンスに関する情報を取得します。
- Amazon CloudWatch メトリクスの収集を有効にします。
- グループの希望するキャパシティを更新し、インスタンスが起動するのを待ちます。
- グループ内の最も古いインスタンスを削除します。
- ユーザーのリクエストやキャパシティの変更に応じて発生するスケーリングアクティビティを一覧表示します。
- CloudWatch メトリクスの統計を取得して、リソースをクリーンアップします。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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.AddLaunchTemplateNames(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 の詳細については、「AWS SDK for C++ API リファレンス」の以下のトピックを参照してください。
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

アクション

`CreateAutoScalingGroup`

次のコード例は、`CreateAutoScalingGroup` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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;

        }
```

- 詳細については、AWS SDK for C++ API リファレンスの「CreateAutoScalingGroup」を参照してください。

DeleteAutoScalingGroup

次のコード例は、DeleteAutoScalingGroup を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    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;
        }
    }
```

- API の詳細については、AWS SDK for C++ API リファレンスの「DeleteAutoScalingGroup」を参照してください。

`DescribeAutoScalingGroups`

次のコード例は、`DescribeAutoScalingGroups` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    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;
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「DescribeAutoScalingGroups」を参照してください。

`DescribeAutoScalingInstances`

次のコード例は、DescribeAutoScalingInstances を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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;
        }
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「DescribeAutoScalingInstances」を参照してください。

`DescribeScalingActivities`

次のコード例は、`DescribeScalingActivities` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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;
        }
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「DescribeScalingActivities」を参照してください。

`DisableMetricsCollection`

次のコード例は、`DisableMetricsCollection` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    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;

    }
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「DisableMetricsCollection」を参照してください。

`EnableMetricsCollection`

次のコード例は、`EnableMetricsCollection` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    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;

        }
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「EnableMetricsCollection」を参照してください。

SetDesiredCapacity

次のコード例は、SetDesiredCapacity を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    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;

        }
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「SetDesiredCapacity」を参照してください。

`TerminateInstanceInAutoScalingGroup`

次のコード例は、TerminateInstanceInAutoScalingGroup を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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;

        }
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「TerminateInstanceInAutoScalingGroup」を参照してください。

UpdateAutoScalingGroup

次のコード例は、UpdateAutoScalingGroup を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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;
```

```
        }
```

- API の詳細については、AWS SDK for C++ API リファレンスの「UpdateAutoScalingGroup」を参照してください。

SDK for C++ を使用した CloudTrail の例

次のコード例は、CloudTrail AWS SDK for C++ で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- アクション

アクション

`CreateTrail`

次のコード例は、`CreateTrail` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
// 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 の詳細については、AWS SDK for C++ API リファレンスの「CreateTrail」を参照してください。

`DeleteTrail`

次のコード例は、`DeleteTrail` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
// 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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteTrail」を参照してください。

`DescribeTrail`

次のコード例は、`DescribeTrail` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
// 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 の詳細については、AWS SDK for C++ API リファレンスの「DescribeTrail」を参照してください。

`LookupEvents`

次のコード例は、`LookupEvents` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
// 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 の詳細については、AWS SDK for C++ API リファレンスの「LookupEvents」を参照してください。

SDK for C++ を使用した CloudWatch の例

次のコード例は、CloudWatch AWS SDK for C++ で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- アクション

アクション

`DeleteAlarms`

次のコード例は、`DeleteAlarms` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。完全な例を見つけて、AWS コード例リポジトリでの設定と実行の方法を確認してください。

必要なファイルを含めます。

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

アラームを削除します。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteAlarms」を参照してください。

`DescribeAlarmsForMetric`

次のコード例は、`DescribeAlarmsForMetric` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。完全な例を見つけて、AWS コード例リポジトリでの設定と実行の方法を確認してください。

必要なファイルを含めます。

```
#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>
```

アラームを記述します。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「DescribeAlarmsForMetric」を参照してください。

`DisableAlarmActions`

次のコード例は、`DisableAlarmActions` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。完全な例を見つけて、AWS コード例リポジトリでの設定と実行の方法を確認してください。

必要なファイルを含めます。

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

アラームアクションを無効化します。

```
    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 詳細については、AWS SDK for C++ API リファレンスの「DisableAlarmActions」を参照してください。

`EnableAlarmActions`

次のコード例は、`EnableAlarmActions` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。完全な例を見つけて、AWS コード例リポジトリでの設定と実行の方法を確認してください。

必要なファイルを含めます。

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

アラームアクションを有効化します。

```
    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 の詳細については、AWS SDK for C++ API リファレンスの「EnableAlarmActions」を参照してください。

`ListMetrics`

次のコード例は、`ListMetrics` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。完全な例を見つけて、AWS コード例リポジトリでの設定と実行の方法を確認してください。

必要なファイルを含めます。

```
#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>
```

メトリクスを一覧表示します。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「ListMetrics」を参照してください。

`PutMetricAlarm`

次のコード例は、`PutMetricAlarm` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。完全な例を見つけて、AWS コード例リポジトリでの設定と実行の方法を確認してください。

必要なファイルを含めます。

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

メトリクスを監視するアラームを作成します。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「PutMetricAlarm」を参照してください。

`PutMetricData`

次のコード例は、`PutMetricData` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。完全な例を見つけて、AWS コード例リポジトリでの設定と実行の方法を確認してください。

必要なファイルを含めます。

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

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

メトリクスにデータを配置します。

```
        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 の詳細については、「AWS SDK for C++ API リファレンス」の「PutMetricData」を参照してください。

SDK for C++ を使用した CloudWatch Logs の例

次のコード例は、CloudWatch Logs AWS SDK for C++ で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- アクション

アクション

`DeleteSubscriptionFilter`

次のコード例は、`DeleteSubscriptionFilter` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。完全な例を見つけて、AWS コード例リポジトリでの設定と実行の方法を確認してください。

必要なファイルを含めます。

```
#include <aws/core/Aws.h>
#include <aws/core/utils/Outcome.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/DeleteSubscriptionFilterRequest.h>
#include <iostream>
```

サブスクリプションフィルターを削除します。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteSubscriptionFilter」を参照してください。

`DescribeSubscriptionFilters`

次のコード例は、`DescribeSubscriptionFilters` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。完全な例を見つけて、AWS コード例リポジトリでの設定と実行の方法を確認してください。

必要なファイルを含めます。

```
#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>
```

サブスクリプションフィルターを一覧表示します。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「DescribeSubscriptionFilters」を参照してください。

`PutSubscriptionFilter`

次のコード例は、`PutSubscriptionFilter` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。完全な例を見つけて、AWS コード例リポジトリでの設定と実行の方法を確認してください。

必要なファイルを含めます。

```
#include <aws/core/Aws.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/PutSubscriptionFilterRequest.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

サブスクリプションフィルターを作成します。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「PutSubscriptionFilter」を参照してください。

SDK for C++ を使用した CodeBuild の例

次のコード例は、CodeBuild AWS SDK for C++ で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック
- アクション

アクション

`ListBuilds`

次のコード例は、`ListBuilds` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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());
                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;
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「ListBuilds」を参照してください。

`ListProjects`

次のコード例は、`ListProjects` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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();
        }

        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 の詳細については、「AWS SDK for C++ API リファレンス」の「ListProjects」を参照してください。

StartBuild

次のコード例は、StartBuild を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、「AWS SDK for C++ API リファレンス」の「StartBuild」を参照してください。

SDK for C++ を使用する Amazon Cognito ID プロバイダーの例

次のコード例は、Amazon Cognito ID プロバイダー AWS SDK for C++ で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

シナリオは、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- はじめに
- アクション
- シナリオ

はじめに

Hello Amazon Cognito

次のコード例は、Amazon Cognito の使用を開始する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

CMakeLists.txt CMake ファイルのコード。

```
# 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 ソースファイルのコード。

```
#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 の詳細については、AWS SDK for C++ API リファレンスの「ListUserPools」を参照してください。

アクション

`AdminGetUser`

次のコード例は、`AdminGetUser` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「AdminGetUser」を参照してください。

`AdminInitiateAuth`

次のコード例は、`AdminInitiateAuth` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「AdminInitiateAuth」を参照してください。

`AdminRespondToAuthChallenge`

次のコード例は、AdminRespondToAuthChallenge を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
       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 の詳細については、AWS SDK for C++ API リファレンスの「AdminRespondToAuthChallenge」を参照してください。

AssociateSoftwareToken

次のコード例は、AssociateSoftwareToken を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「AssociateSoftwareToken」を参照してください。

`ConfirmSignUp`

次のコード例は、ConfirmSignUp を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「ConfirmSignUp」を参照してください。

`DeleteUser`

次のコード例は、`DeleteUser` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteUser」を参照してください。

`ResendConfirmationCode`

次のコード例は、`ResendConfirmationCode` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「ResendConfirmationCode」を参照してください。

SignUp

次のコード例は、SignUp を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「SignUp」を参照してください。

`VerifySoftwareToken`

次のコード例は、`VerifySoftwareToken` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、「AWS SDK for C++ API リファレンス」の「VerifySoftwareToken」を参照してください。

シナリオ

MFA を必要とするユーザープールでユーザーをサインアップする

次のコード例は、以下の操作方法を示しています。

- ユーザー名、パスワード、E メールアドレスでサインアップしてユーザーを確認します。
- MFA アプリケーションをユーザーに関連付けて、多要素認証を設定します。
- パスワードと MFA コードを使用してサインインします。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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;
        }
        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 の詳細については、「AWS SDK for C++ API リファレンス」の以下のトピックを参照してください。
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp

- VerifySoftwareToken

SDK for C++ を使用した DynamoDB の例

次のコード例は、DynamoDB AWS SDK for C++ で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

基本 は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

シナリオは、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- はじめに
- 基本
- アクション
- シナリオ

はじめに

Hello DynamoDB

次のコード例は、DynamoDB の使用を開始する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

CMakeLists.txt CMake ファイルのコード。

```
# 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 ソースファイルのコード。

```
#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 の詳細については、AWS SDK for C++ API リファレンスの「ListTables」を参照してください。

基本

基本を学ぶ

次のコード例は、以下の操作方法を示しています。

- 映画データを保持できるテーブルを作成します。
- テーブルに 1 つの映画を入れ、取得して更新する。
- サンプル JSON ファイルから映画データをテーブルに書き込む。
- 特定の年にリリースされた映画を照会する。
- 一定期間内に公開された映画をスキャンします。
- テーブルから映画を削除し、テーブルを削除します。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    {
        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 の詳細については、「AWS SDK for C++ API リファレンス」の以下のトピックを参照してください。
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem

- PutItem
- Query
- Scan
- UpdateItem

アクション

`BatchExecuteStatement`

次のコード例は、BatchExecuteStatement を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

INSERT ステートメントのバッチを使用して項目を追加します。

```
    // 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 ステートメントのバッチを使用して項目を取得します。

```
    // 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 ステートメントのバッチを使用して項目を更新します。

```
    // 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 ステートメントのバッチを使用して項目を削除します。

```
    // 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 の詳細については、AWS SDK for C++ API リファレンスの「BatchExecuteStatement」を参照してください。

BatchGetItem

次のコード例は、BatchGetItem を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「BatchGetItem」を参照してください。

`BatchWriteItem`

次のコード例は、`BatchWriteItem` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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::JsonView> 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::JsonView> &requestsJson,

 Aws::Vector<Aws::DynamoDB::Model::WriteRequest> &writeRequests) {
    for (size_t i = 0; i < requestsJson.GetLength(); ++i) {
        const Aws::Utils::Json::JsonView &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::JsonView> requestsMap =
 requestsEntry.GetAllObjects();

        for (const auto &request: requestsMap) {
            const Aws::String &requestType = request.first;
            const Aws::Utils::Json::JsonView &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 の詳細については、AWS SDK for C++ API リファレンスの「BatchWriteItem」を参照してください。

`CreateTable`

次のコード例は、`CreateTable` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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);
}
```

テーブルがアクティブになるまで待機するコード。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「CreateTable」を参照してください。

DeleteItem

次のコード例は、DeleteItem を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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);
}
```

テーブルがアクティブになるまで待機するコード。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteItem」を参照してください。

`DeleteTable`

次のコード例は、`DeleteTable` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteTable」を参照してください。

`DescribeTable`

次のコード例は、`DescribeTable` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「DescribeTable」を参照してください。

`ExecuteStatement`

次のコード例は、`ExecuteStatement` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

INSERT ステートメントを使用して項目を追加します。

```
    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 ステートメントを使用して項目を取得します。

```
    //  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 ステートメントを使用して項目を更新します。

```
    //  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 ステートメントを使用して項目を削除します。

```
    // 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 の詳細については、AWS SDK for C++ API リファレンスの「ExecuteStatement」を参照してください。

`GetItem`

次のコード例は、`GetItem` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「GetItem」を参照してください。

ListTables

次のコード例は、ListTables を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「ListTables」を参照してください。

`PutItem`

次のコード例は、`PutItem` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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);
}
```

テーブルがアクティブになるまで待機するコード。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスのの「PutItem」を参照してください。

Query

次のコード例は、Query を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「Query」を参照してください。

Scan

次のコード例は、Scan を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「Scan」を参照してください。

`UpdateItem`

次のコード例は、`UpdateItem` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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);
}
```

テーブルがアクティブになるまで待機するコード。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「UpdateItem」を参照してください。

`UpdateTable`

次のコード例は、UpdateTable を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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);
}
```

テーブルがアクティブになるまで待機するコード。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「UpdateTable」を参照してください。

シナリオ

サーバーレスアプリケーションを作成して写真を管理する

次のコード例では、ユーザーがラベルを使用して写真を管理できるサーバーレスアプリケーションを作成する方法について示しています。

SDK for C++

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードと設定および実行の手順については、GitHub で完全な例を参照してください。

この例のソースについて詳しくは、AWS コミュニティでブログ投稿を参照してください。

この例で使用されているサービス

- API ゲートウェイ
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PartiQL ステートメントのバッチを使用してテーブルに対してクエリを実行する

次のコードサンプルは、以下の操作方法を示しています。

- 複数の SELECT ステートメントを実行して、項目のバッチを取得します。
- 複数の INSERT ステートメントを実行して、項目のバッチを追加する。
- 複数の UPDATE ステートメントを実行して、項目のバッチを更新する。
- 複数の DELETE ステートメントを実行して、項目のバッチを削除します。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、「AWS SDK for C++ API リファレンス」の「BatchExecuteStatement」を参照してください。

PartiQL を使用してテーブルに対してクエリを実行する

次のコードサンプルは、以下の操作方法を示しています。

- SELECT ステートメントを実行して項目を取得します。
- INSERT 文を実行して項目を追加する。
- UPDATE ステートメントを使用して項目を更新する。
- DELETE ステートメントを実行して項目を削除します。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        //  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 の詳細については、AWS SDK for C++ API リファレンスの「ExecuteStatement」を参照してください。

SDK for C++ を使用した Amazon EC2 の例

次のコード例は、Amazon EC2 AWS SDK for C++ で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- はじめに
- アクション

はじめに

Hello Amazon EC2

次のコード例は、Amazon EC2 の使用を開始する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

CMakeLists.txt CMake ファイルのコード。

```
# 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 ソースファイルのコード。

```
#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 の詳細については、AWS SDK for C++ API リファレンスの「DescribeSecurityGroups」を参照してください。

アクション

`AllocateAddress`

次のコード例は、`AllocateAddress` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「AllocateAddress」を参照してください。

`AssociateAddress`

次のコード例は、`AssociateAddress` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    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 の詳細については、AWS SDK for C++ API リファレンスの「AssociateAddress」を参照してください。

AuthorizeSecurityGroupIngress

次のコード例は、AuthorizeSecurityGroupIngress を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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();
}
```

進入ルールを構築するユーティリティ関数。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「AuthorizeSecurityGroupIngress」を参照してください。

CreateKeyPair

次のコード例は、CreateKeyPair を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「CreateKeyPair」を参照してください。

`CreateSecurityGroup`

次のコード例は、`CreateSecurityGroup` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「CreateSecurityGroup」を参照してください。

`CreateTags`

次のコード例は、`CreateTags` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、「AWS SDK for C++ API リファレンス」の「 CreateTags」を参照してください。

`DeleteKeyPair`

次のコード例は、`DeleteKeyPair` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteKeyPair」を参照してください。

DeleteSecurityGroup

次のコード例は、DeleteSecurityGroup を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteSecurityGroup」を参照してください。

DescribeAddresses

次のコード例は、DescribeAddresses を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、「AWS SDK for C++ API リファレンス」の「DescribeAddresses」を参照してください。

`DescribeAvailabilityZones`

次のコード例は、`DescribeAvailabilityZones` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「DescribeAvailabilityZones」を参照してください。

`DescribeInstances`

次のコード例は、`DescribeInstances` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「DescribeInstances」を参照してください。

DescribeKeyPairs

次のコード例は、DescribeKeyPairs を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「DescribeKeyPairs」を参照してください。

`DescribeRegions`

次のコード例は、`DescribeRegions` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「DescribeRegions」を参照してください。

`DescribeSecurityGroups`

次のコード例は、`DescribeSecurityGroups` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「DescribeSecurityGroups」を参照してください。

MonitorInstances

次のコード例は、MonitorInstances を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「MonitorInstances」を参照してください。

`RebootInstances`

次のコード例は、RebootInstances を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「RebootInstances」を参照してください。

`ReleaseAddress`

次のコード例は、`ReleaseAddress` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「ReleaseAddress」を参照してください。

`RunInstances`

次のコード例は、`RunInstances` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「RunInstances」を参照してください。

`StartInstances`

次のコード例は、`StartInstances` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「StartInstances」を参照してください。

StopInstances

次のコード例は、StopInstances を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「StopInstances」を参照してください。

`TerminateInstances`

次のコード例は、`TerminateInstances` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「TerminateInstances」を参照してください。

`UnmonitorInstances`

次のコード例は、`UnmonitorInstances` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「UnmonitorInstances」を参照してください。

SDK for C++ を使用した EventBridge の例

次のコード例は、EventBridge AWS SDK for C++ で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- アクション

アクション

`PutEvents`

次のコード例は、`PutEvents` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。完全な例を見つけて、AWS コード例リポジトリでの設定と実行の方法を確認してください。

必要なファイルを含めます。

```
#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>
```

イベントを送信します。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「PutEvents」を参照してください。

`PutRule`

次のコード例は、`PutRule` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。完全な例を見つけて、AWS コード例リポジトリでの設定と実行の方法を確認してください。

必要なファイルを含めます。

```
#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>
```

ルールを作成します。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「PutRule」を参照してください。

PutTargets

次のコード例は、PutTargets を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。完全な例を見つけて、AWS コード例リポジトリでの設定と実行の方法を確認してください。

必要なファイルを含めます。

```
#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>
```

ターゲットとして追加します。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「PutTargets」を参照してください。

AWS Glue SDK for C++ を使用した例

次のコード例は、 AWS SDK for C++ で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています AWS Glue。

基本 は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- はじめに
- 基本
- アクション

はじめに

こんにち AWS Glueは

次のコード例は、 AWS Glueの使用を開始する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

CMakeLists.txt CMake ファイルのコード。

```
# 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 ソースファイルのコード。

```
#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 の詳細については、AWS SDK for C++ API リファレンスの「ListJobs」を参照してください。

基本

基本を学ぶ

次のコード例は、以下の操作方法を示しています。

- パブリック Amazon S3 バケットをクロールし、CSV 形式のメタデータのデータベースを生成するクローラーを作成します。
- のデータベースとテーブルに関する情報を一覧表示します AWS Glue Data Catalog。

- S3 バケットから CSV 形式のデータを抽出するジョブを作成し、そのデータを変換して JSON 形式の出力を別の S3 バケットにロードする。
- ジョブ実行に関する情報を一覧表示し、変換されたデータを表示してリソースをクリーンアップします。

詳細については、「チュートリアル: AWS Glue Studio の開始方法」を参照してください。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、「AWS SDK for C++ API リファレンス」の以下のトピックを参照してください。
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase

- DeleteJob
- DeleteTable
- GetCrawler
- GetDatabase
- GetDatabases
- GetJob
- GetJobRun
- GetJobRuns
- GetTables
- ListJobs
- StartCrawler
- StartJobRun

アクション

`CreateCrawler`

次のコード例は、`CreateCrawler` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「CreateCrawler」を参照してください。

`CreateJob`

次のコード例は、`CreateJob` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「CreateJob」を参照してください。

DeleteCrawler

次のコード例は、DeleteCrawler を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteCrawler」を参照してください。

`DeleteDatabase`

次のコード例は、`DeleteDatabase` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteDatabase」を参照してください。

`DeleteJob`

次のコード例は、`DeleteJob` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteJob」を参照してください。

`GetCrawler`

次のコード例は、`GetCrawler` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「GetCrawler」を参照してください。

GetDatabase

次のコード例は、GetDatabase を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
       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 の詳細については、AWS SDK for C++ API リファレンスの「GetDatabase」を参照してください。

GetJobRun

次のコード例は、GetJobRun を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「GetJobRun」を参照してください。

`GetJobRuns`

次のコード例は、`GetJobRuns` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「GetJobRuns」を参照してください。

GetTables

次のコード例は、GetTables を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
       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 の詳細については、AWS SDK for C++ API リファレンスの「GetTables」を参照してください。

`ListJobs`

次のコード例は、`ListJobs` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「ListJobs」を参照してください。

`StartCrawler`

次のコード例は、`StartCrawler` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
       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 の詳細については、AWS SDK for C++ API リファレンスの「StartCrawler」を参照してください。

`StartJobRun`

次のコード例は、`StartJobRun` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「StartJobRun」を参照してください。

SDK for C++ を使用した HealthImaging の例

次のコード例は、HealthImaging AWS SDK for C++ で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

シナリオは、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- はじめに

- アクション
- シナリオ

はじめに

Hello HealthImaging

次のコード例は、HealthImaging の使用を開始する方法を示しています。

SDK for C++

CMakeLists.txt CMake ファイルのコード。

```
# 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_CPY_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 ソースファイルのコード。

```
#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;
}
```

- API の詳細については、AWS SDK for C++ API リファレンスの「ListDatastores」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

アクション

`DeleteImageSet`

次のコード例は、DeleteImageSet を使用する方法を示しています。

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();
}
```

- API の詳細については、AWS SDK for C++ API リファレンスの「DeleteImageSet」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

GetDICOMImportJob

次のコード例は、GetDICOMImportJob を使用する方法を示しています。

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;
}
```

- API の詳細については、AWS SDK for C++ API リファレンスの「GetDICOMImportJob」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

`GetImageFrame`

次のコード例は、`GetImageFrame` を使用する方法を示しています。

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();
}
```

- API の詳細については、AWS SDK for C++ API リファレンスの「GetImageFrame」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

`GetImageSetMetadata`

次のコード例は、`GetImageSetMetadata` を使用する方法を示しています。

SDK for C++

イメージセットのメタデータを取得するためのユーティリティ関数。

```
//! 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();
}
```

イメージセットのメタデータをバージョンなしで取得します。

```
        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;
        }
```

イメージセットのメタデータをバージョン付きで取得します。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「GetImageSetMetadata」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

`SearchImageSets`

次のコード例は、`SearchImageSets` を使用する方法を示しています。

SDK for C++

画像セットを検索するためのユーティリティ関数。

```
//! 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;
}
```

ユースケース #1: EQUAL 演算子。

```
        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::Operato

 .WithValues({Aws::MedicalImaging::Model::SearchByAttributeValue().WithDICOMPatientId(patien
        };

        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;
            }
        }
```

ユースケース #2: DICOMStudyDate と DICOMStudyTime を使用する BETWEEN 演算子。

```
         Aws::MedicalImaging::Model::SearchByAttributeValue useCase2StartDate;

 useCase2StartDate.SetDICOMStudyDateAndTime(Aws::MedicalImaging::Model::DICOMStudyDateAndTim
        .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;
            }
        }
```

ユースケース #3: createdAt を使用する BETWEEN 演算子。タイムスタディは以前に永続化されています。

```
        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;
            }
        }
```

ユースケース 4: DICOMSeriesInstanceUID で EQUAL 演算子を使用し、updatedAt で BETWEEN 演算子を使用して、updatedAt フィールドで ASC 順序にレスポンスをソートします。

```
        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;
            }
        }
```

- API の詳細については、AWS SDK for C++ API リファレンスの「SearchImageSets」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

StartDICOMImportJob

次のコード例は、StartDICOMImportJob を使用する方法を示しています。

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();
}
```

- API の詳細については、AWS SDK for C++ API リファレンスの「StartDICOMImportJob」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

シナリオ

画像セットと画像フレームを使い始める

次のコード例は、HealthImaging で DICOM ファイルをインポートし、画像フレームをダウンロードする方法を示しています。

実装はコマンドラインアプリケーションとして構造化されています。

- DICOM インポート用にリソースをセットアップします。
- DICOM ファイルをデータストアへのインポート。
- インポートジョブの画像セット ID の取得。
- インポートジョブの画像フレーム ID の取得。
- イメージフレームをダウンロード、デコード、および検証します。
- リソースをクリーンアップします。

SDK for C++

必要なリソースを持つ CloudFormation スタックを作成します。

```
   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: ");
    }
```

DICOM ファイルを Amazon S3 インポートバケットにコピーします。

```
    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;
    }
```

DICOM ファイルを Amazon S3 データストアのにインポートします。

```
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 インポートジョブによって作成された画像セットを取得します。

```
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;
}
```

画像セットの画像フレーム情報を取得します。

```
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 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();
}
```

画像フレームをダウンロード、デコード、および検証します。

```
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;
}
```

リソースをクリーンアップします。

```
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 の詳細については、「AWS SDK for C++ API リファレンス」の以下のトピックを参照してください。
  - DeleteImageSet
  - GetDICOMImportJob
  - GetImageFrame
  - GetImageSetMetadata
  - SearchImageSets
  - StartDICOMImportJob

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

SDK for C++ を使用する IAM の例

次のコード例は、IAM AWS SDK for C++ で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

基本 は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- はじめに
- 基本
- アクション

はじめに

IAM へようこそ

次のコード例は、IAM の使用を開始する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

CMakeLists.txt CMake ファイルのコード。

```
# 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 ソースファイルのコード。

```
#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 の詳細については、AWS SDK for C++ API リファレンスの「ListPolicies」を参照してください。

基本

基本を学ぶ

次のコード例は、ユーザーを作成してロールを割り当てる方法を示しています。

> **Warning**
>
> セキュリティリスクを避けるため、専用ソフトウェアを開発するときや実際のデータを扱うときは、IAM ユーザーを認証に使用しないでください。代わりに、AWS IAM アイデンティティセンター などの ID プロバイダーとのフェデレーションを使用してください。

- 権限のないユーザーを作成します。
- アカウントの Amazon S3 バケットを一覧表示する権限を付与するロールを作成します。
- ユーザーがロールを引き受けられるようにポリシーを追加します。
- ロールを引き受け、一時的な認証情報を使用して S3 バケットを一覧表示し、リソースをクリーンアップします。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、「AWS SDK for C++ API リファレンス」の以下のトピックを参照してください。
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy

- CreateRole
- CreateUser
- DeleteAccessKey
- DeletePolicy
- DeleteRole
- DeleteUser
- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

アクション

`AttachRolePolicy`

次のコード例は、`AttachRolePolicy` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「AttachRolePolicy」を参照してください。

CreateAccessKey

次のコード例は、CreateAccessKey を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「CreateAccessKey」を参照してください。

CreateAccountAlias

次のコード例は、CreateAccountAlias を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「CreateAccountAlias」を参照してください。

CreatePolicy

次のコード例は、CreatePolicy を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「CreatePolicy」を参照してください。

`CreateRole`

次のコード例は、`CreateRole` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「CreateRole」を参照してください。

`CreateUser`

次のコード例は、`CreateUser` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    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 の詳細については、AWS SDK for C++ API リファレンスの「CreateUser」を参照してください。

`DeleteAccessKey`

次のコード例は、`DeleteAccessKey` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteAccessKey」を参照してください。

`DeleteAccountAlias`

次のコード例は、`DeleteAccountAlias` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteAccountAlias」を参照してください。

`DeletePolicy`

次のコード例は、`DeletePolicy` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「DeletePolicy」を参照してください。

`DeleteServerCertificate`

次のコード例は、`DeleteServerCertificate` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteServerCertificate」を参照してください。

`DeleteUser`

次のコード例は、`DeleteUser` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteUser」を参照してください。

`DetachRolePolicy`

次のコード例は、`DetachRolePolicy` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「DetachRolePolicy」を参照してください。

GetAccessKeyLastUsed

次のコード例は、GetAccessKeyLastUsed を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「GetAccessKeyLastUsed」を参照してください。

GetPolicy

次のコード例は、GetPolicy を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「GetPolicy」を参照してください。

GetServerCertificate

次のコード例は、GetServerCertificate を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「GetServerCertificate」を参照してください。

`ListAccessKeys`

次のコード例は、`ListAccessKeys` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「ListAccessKeys」を参照してください。

ListAccountAliases

次のコード例は、ListAccountAliases を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「ListAccountAliases」を参照してください。

`ListPolicies`

次のコード例は、`ListPolicies` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「ListPolicies」を参照してください。

`ListServerCertificates`

次のコード例は、`ListServerCertificates` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「ListServerCertificates」を参照してください。

`ListUsers`

次のコード例は、`ListUsers` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「ListUsers」を参照してください。

PutRolePolicy

次のコード例は、PutRolePolicy を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「PutRolePolicy」を参照してください。

`UpdateAccessKey`

次のコード例は、`UpdateAccessKey` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「UpdateAccessKey」を参照してください。

`UpdateServerCertificate`

次のコード例は、`UpdateServerCertificate` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「UpdateServerCertificate」を参照してください。

UpdateUser

次のコード例は、UpdateUser を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「UpdateUser」を参照してください。

AWS IoT SDK for C++ を使用した例

次のコード例は、 AWS SDK for C++ で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています AWS IoT。

基本 は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- はじめに
- 基本
- アクション

はじめに

こんにち AWS IoTは

次のコード例は、 AWS IoTの使用を開始する方法を示しています。

SDK for C++

CMakeLists.txt CMake ファイルのコード。

```
# 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 ソースファイルのコード。

```
#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 の詳細については、AWS SDK for C++ API リファレンスの「listThings」を参照してください。

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

基本

基本を学ぶ

次のコード例は、以下の操作方法を示しています。

- AWS IoT モノを作成します。
- デバイス証明書を生成します。
- 属性を使用して AWS IoT モノを更新します。
- 一意のエンドポイントを返します。
- AWS IoT 証明書を一覧表示します。
- AWS IoT シャドウを更新します。
- 状態情報を書き込みます。
- ルールを作成する。
- ルールを一覧表示します。
- モノの名前を使用してモノを検索します。
- AWS IoT モノを削除します。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

AWS IoT モノを作成します。

```
    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;
    }
```

```
//! 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();
}
```

デバイス証明書を生成してアタッチします。

```
    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 モノに対してさまざまなオペレーションを実行します。

```
    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);
```

```
        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;
}
```

リソースをクリーンアップします。

```
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();
}
```

アクション

`AttachThingPrincipal`

次のコード例は、`AttachThingPrincipal` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「AttachThingPrincipal」を参照してください。

`CreateKeysAndCertificate`

次のコード例は、`CreateKeysAndCertificate` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「CreateKeysAndCertificate」を参照してください。

`CreateThing`

次のコード例は、`CreateThing` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「CreateThing」を参照してください。

`CreateTopicRule`

次のコード例は、`CreateTopicRule` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「CreateTopicRule」を参照してください。

DeleteCertificate

次のコード例は、DeleteCertificate を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteCertificate」を参照してください。

`DeleteThing`

次のコード例は、DeleteThing を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteThing」を参照してください。https://docs.aws.amazon.com/goto/SdkForCpp/iot-2015-05-28/DeleteThing

DeleteTopicRule

次のコード例は、DeleteTopicRule を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、「AWS SDK for C++ API リファレンス」の「DeleteTopicRule」を参照してください。

`DescribeEndpoint`

次のコード例は、DescribeEndpoint を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「DescribeEndpoint」を参照してください。

`DescribeThing`

次のコード例は、`DescribeThing` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「DescribeThing」を参照してください。

`DetachThingPrincipal`

次のコード例は、`DetachThingPrincipal` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「DetachThingPrincipal」を参照してください。

`ListCertificates`

次のコード例は、`ListCertificates` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「ListCertificates」を参照してください。

`SearchIndex`

次のコード例は、`SearchIndex` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「SearchIndex」を参照してください。

`UpdateIndexingConfiguration`

次のコード例は、`UpdateIndexingConfiguration` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、「AWS SDK for C++ API リファレンス」の「UpdateIndexingConfiguration」を参照してください。

`UpdateThing`

次のコード例は、UpdateThing を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「UpdateThing」を参照してください。

AWS IoT data SDK for C++ を使用した例

次のコード例は、 AWS SDK for C++ で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています AWS IoT data。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- アクション

アクション

`GetThingShadow`

次のコード例は、`GetThingShadow` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、「 API リファレンスAWS SDK for C++ 」の「GetThingShadow」を参照してください。

UpdateThingShadow

次のコード例は、UpdateThingShadow を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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();
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「UpdateThingShadow」を参照してください。

SDK for C++ を使用した Lambda の例

次のコード例は、Lambda AWS SDK for C++ で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

基本 は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

シナリオは、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- はじめに
- 基本
- アクション
- シナリオ

はじめに

Hello Lambda

次のコード例は、Lambda の使用を開始する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

CMakeLists.txt CMake ファイルのコード。

```
# 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 ソースファイルのコード。

```
#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 の詳細については、AWS SDK for C++ API リファレンスの「ListFunctions」を参照してください。

基本

基本を学ぶ

次のコード例は、以下の操作方法を示しています。

- IAM ロールと Lambda 関数を作成し、ハンドラーコードをアップロードします。
- 1 つのパラメータで関数を呼び出して、結果を取得します。
- 関数コードを更新し、環境変数で設定します。
- 新しいパラメータで関数を呼び出して、結果を取得します。返された実行ログを表示します。
- アカウントの関数を一覧表示し、リソースをクリーンアップします。

詳細については、「コンソールで Lambda 関数を作成する」を参照してください。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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.
#if 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.
#if  defined(__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::JsonView> 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;
```

```
#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
            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 の詳細については、「AWS SDK for C++ API リファレンス」の以下のトピックを参照してください。
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

アクション

`CreateFunction`

次のコード例は、`CreateFunction` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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.
#if 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.
#if  defined(__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 {
            std::cerr << "Error with CreateFunction. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            deleteIamRole(clientConfig);
            return false;
        }
```

- API の詳細については、AWS SDK for C++ API リファレンスの「CreateFunction」を参照してください。

`DeleteFunction`

次のコード例は、`DeleteFunction` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteFunction」を参照してください。

GetFunction

次のコード例は、GetFunction を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「GetFunction」を参照してください。

`Invoke`

次のコード例は、Invoke を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「Invoke」を参照してください。

`ListFunctions`

次のコード例は、`ListFunctions` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「ListFunctions」を参照してください。

UpdateFunctionCode

次のコード例は、UpdateFunctionCode を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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;

#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
            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 の詳細については、AWS SDK for C++ API リファレンスの「UpdateFunctionCode」を参照してください。

`UpdateFunctionConfiguration`

次のコード例は、UpdateFunctionConfiguration を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「UpdateFunctionConfiguration」を参照してください。

シナリオ

サーバーレスアプリケーションを作成して写真を管理する

次のコード例では、ユーザーがラベルを使用して写真を管理できるサーバーレスアプリケーションを作成する方法について示しています。

SDK for C++

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードと設定および実行の手順については、GitHub で完全な例を参照してください。

この例のソースについて詳しくは、AWS コミュニティでブログ投稿を参照してください。

この例で使用されているサービス

- API ゲートウェイ
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

SDK for C++ を使用した MediaConvert の例

次のコード例は、MediaConvert AWS SDK for C++ で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- アクション

アクション

`CreateJob`

次のコード例は、`CreateJob` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「CreateJob」を参照してください。

`GetJob`

次のコード例は、`GetJob` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「GetJob」を参照してください。

`ListJobs`

次のコード例は、`ListJobs` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「ListJobs」を参照してください。

SDK for C++ を使用した Amazon RDS の例

次のコード例は、Amazon RDS AWS SDK for C++ で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

基本 は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

シナリオは、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- はじめに
- 基本
- アクション
- シナリオ

はじめに

Hello Amazon RDS

次のコード例は、Amazon RDS の使用を開始する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

CMakeLists.txt CMake ファイルのコード。

```
# 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 ソースファイルのコード。

```
#include <aws/core/Aws.h>
#include <aws/rds/RDSClient.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 の詳細については、AWS SDK for C++ API リファレンスの「DescribeDBInstances」を参照してください。

基本

基本を学ぶ

次のコード例は、以下の操作方法を示しています。

- カスタム DB パラメータグループを作成し、パラメータ値を設定します。
- パラメータグループを使用するように設定した DB インスタンスを作成します。DB インスタンスにはデータベースも含まれています。
- インスタンスのスナップショットを取得します。
- インスタンスとパラメータグループを削除します。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 ofter "
                      << 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 ofter "
                          << 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 dbInstanceIdentifier: 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 &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 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 ofter " <<
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 の詳細については、「AWS SDK for C++ API リファレンス」の以下のトピックを参照してください。
  - CreateDBInstance
  - CreateDBParameterGroup
  - CreateDBSnapshot
  - DeleteDBInstance
  - DeleteDBParameterGroup
  - DescribeDBEngineVersions
  - DescribeDBInstances
  - DescribeDBParameterGroups
  - DescribeDBParameters
  - DescribeDBSnapshots
  - DescribeOrderableDBInstanceOptions
  - ModifyDBParameterGroup

アクション

`CreateDBInstance`

次のコード例は、`CreateDBInstance` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    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 の詳細については、AWS SDK for C++ API リファレンスの「CreateDBInstance」を参照してください。

`CreateDBParameterGroup`

次のコード例は、`CreateDBParameterGroup` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    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;
    }
```

- 詳細については、「AWS SDK for C++ API リファレンス」の「CreateDBParameterGroup」を参照してください。

`CreateDBSnapshot`

次のコード例は、`CreateDBSnapshot` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、「AWS SDK for C++ API リファレンス」の「CreateDBSnapshot」を参照してください。

`DeleteDBInstance`

次のコード例は、DeleteDBInstance を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteDBInstance」を参照してください。

`DeleteDBParameterGroup`

次のコード例は、`DeleteDBParameterGroup` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
    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;
    }
```

- 詳細については、「AWS SDK for C++ API リファレンス」の「DeleteDBParameterGroup」を参照してください。

DescribeDBEngineVersions

次のコード例は、DescribeDBEngineVersions を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「DescribeDBEngineVersions」を参照してください。

DescribeDBInstances

次のコード例は、DescribeDBInstances を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 dbInstanceIdentifier: 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 &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 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 の詳細については、AWS SDK for C++ API リファレンスの「DescribeDBInstances」を参照してください。

`DescribeDBParameterGroups`

次のコード例は、`DescribeDBParameterGroups` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、「AWS SDK for C++ API リファレンス」の「DescribeDBParameterGroups」を参照してください。

`DescribeDBParameters`

次のコード例は、`DescribeDBParameters` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、「AWS SDK for C++ API リファレンス」の「DescribeDBParameters」を参照してください。

`DescribeDBSnapshots`

次のコード例は、`DescribeDBSnapshots` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、「AWS SDK for C++ API リファレンス」の「DescribeDBSnapshots」を参照してください。

`DescribeOrderableDBInstanceOptions`

次のコード例は、`DescribeOrderableDBInstanceOptions` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「DescribeOrderableDBInstanceOptions」を参照してください。

`ModifyDBParameterGroup`

次のコード例は、`ModifyDBParameterGroup` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、「AWS SDK for C++ API リファレンス」の「ModifyDBParameterGroup」を参照してください。

シナリオ

Aurora Serverless 作業項目トラッカーの作成

次のコード例は、Amazon Aurora Serverless データベースの作業項目を追跡し、Amazon Simple Email Service (Amazon SES) を使用してレポートを送信するウェブアプリケーションを作成する方法を示しています。

SDK for C++

Amazon Aurora Serverless データベースに保存されている作業項目を追跡して報告するウェブアプリケーションを作成する方法を説明します。

Amazon Aurora Serverless データをクエリし、React アプリケーションで使用するための C++ REST API の完全なソースコードと設定方法については、GitHub にある完全な例を参照してください。

この例で使用されているサービス

- Aurora
- Amazon RDS
- Amazon RDS データサービス
- Amazon SES

SDK for C++ を使用した Amazon RDS Data Service の例

次のコード例は、Amazon RDS Data Service AWS SDK for C++ で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

シナリオは、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- シナリオ

シナリオ

Aurora Serverless 作業項目トラッカーの作成

次のコード例は、Amazon Aurora Serverless データベースの作業項目を追跡し、Amazon Simple Email Service (Amazon SES) を使用してレポートを送信するウェブアプリケーションを作成する方法を示しています。

SDK for C++

Amazon Aurora Serverless データベースに保存されている作業項目を追跡して報告するウェブアプリケーションを作成する方法を説明します。

Amazon Aurora Serverless データをクエリし、React アプリケーションで使用するための C++ REST API の完全なソースコードと設定方法については、GitHub にある完全な例を参照してください。

この例で使用されているサービス

- Aurora
- Amazon RDS
- Amazon RDS データサービス
- Amazon SES

SDK for C++ を使用した Amazon Rekognition の例

次のコード例は、Amazon Rekognition AWS SDK for C++ で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

シナリオは、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- はじめに
- アクション
- シナリオ

はじめに

Hello Amazon Rekognition

以下のコード例では、Amazon Rekognition の使用を開始する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

CMakeLists.txt CMake ファイルのコード。

```
# 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 ソースファイルのコード。

```
#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;
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「ListCollections」を参照してください。

アクション

`DetectLabels`

次のコード例は、DetectLabels を使用する方法を示しています。

詳細については、「イメージ内のラベルを検出する」を参照してください。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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();
}
```

- API の詳細については、「AWS SDK for C++ API リファレンス」の「DetectLabels」を参照してください。

シナリオ

サーバーレスアプリケーションを作成して写真を管理する

次のコード例では、ユーザーがラベルを使用して写真を管理できるサーバーレスアプリケーションを作成する方法について示しています。

SDK for C++

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードと設定および実行の手順については、GitHub で完全な例を参照してください。

この例のソースについて詳しくは、AWS コミュニティでブログ投稿を参照してください。

この例で使用されているサービス

- API ゲートウェイ
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

SDK for C++ を使用した Amazon S3 の例

次のコード例は、Amazon S3 AWS SDK for C++ で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

基本 は、重要なオペレーションをサービス内で実行する方法を示すコード例です。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

シナリオは、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- はじめに
- 基本
- アクション
- シナリオ

はじめに

Hello Amazon S3

次のコード例は、Amazon S3 の使用を開始する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

CMakeLists.txt CMake ファイルのコード。

```
# 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 ソースファイルのコード。

```
#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 の詳細については、AWS SDK for C++ API リファレンスの「ListBuckets」を参照してください。

基本

基本を学ぶ

次のコード例は、以下の操作方法を示しています。

- バケットを作成し、そこにファイルをアップロードします。
- バケットからオブジェクトをダウンロードします。
- バケット内のサブフォルダにオブジェクトをコピーします。
- バケット内のオブジェクトを一覧表示します。
- バケットオブジェクトとバケットを削除します。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
#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 の詳細については、「AWS SDK for C++ API リファレンス」の以下のトピックを参照してください。
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

アクション

`AbortMultipartUpload`

次のコード例は、`AbortMultipartUpload` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、「AWS SDK for C++ API リファレンス」の「AbortMultipartUpload」を参照してください。

`CompleteMultipartUpload`

次のコード例は、CompleteMultipartUpload を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、「AWS SDK for C++ API リファレンス」の「CompleteMultipartUpload」を参照してください。

CopyObject

次のコード例は、CopyObject を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「CopyObject」を参照してください。

`CreateBucket`

次のコード例は、`CreateBucket` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「CreateBucket」を参照してください。

CreateMultipartUpload

次のコード例は、CreateMultipartUpload を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、「AWS SDK for C++ API リファレンス」の「CreateMultipartUpload」を参照してください。

`DeleteBucket`

次のコード例は、DeleteBucket を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteBucket」を参照してください。

`DeleteBucketPolicy`

次のコード例は、`DeleteBucketPolicy` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteBucketPolicy」を参照してください。

DeleteBucketWebsite

次のコード例は、DeleteBucketWebsite を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteBucketWebsite」を参照してください。

DeleteObject

次のコード例は、DeleteObject を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteObject」を参照してください。

`DeleteObjects`

次のコード例は、`DeleteObjects` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteObjects」を参照してください。

`GetBucketAcl`

次のコード例は、`GetBucketAcl` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「GetBucketAcl」を参照してください。

GetBucketPolicy

次のコード例は、GetBucketPolicy を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「GetBucketPolicy」を参照してください。

GetBucketWebsite

次のコード例は、GetBucketWebsite を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「GetBucketWebsite」を参照してください。

`GetObject`

次のコード例は、`GetObject` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「GetObject」を参照してください。

`GetObjectAcl`

次のコード例は、`GetObjectAcl` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、「AWS SDK for C++ API リファレンス」の「GetObjectAcl」を参照してください。

`GetObjectAttributes`

次のコード例は、`GetObjectAttributes` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
// ! 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 の詳細については、「AWS SDK for C++ API リファレンス」の「GetObjectAttributes」を参照してください。

`ListBuckets`

次のコード例は、`ListBuckets` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「ListBuckets」を参照してください。

`ListObjectsV2`

次のコード例は、`ListObjectsV2` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「ListObjectsV2」を参照してください。

PutBucketAcl

次のコード例は、PutBucketAcl を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「PutBucketAcl」を参照してください。

`PutBucketPolicy`

次のコード例は、PutBucketPolicy を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「PutBucketPolicy」を参照してください。

PutBucketWebsite

次のコード例は、PutBucketWebsite を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
bool AwsDoc::S3::putWebsiteConfig(const Aws::String &bucketName,
                                  const Aws::String &indexPage, const Aws::String
 &errorPage,
                                  const Aws::S3::S3ClientConfiguration
 &clientConfig) {
    Aws::S3::S3Client client(clientConfig);

    Aws::S3::Model::IndexDocument indexDocument;
    indexDocument.SetSuffix(indexPage);

    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 の詳細については、AWS SDK for C++ API リファレンスの「PutBucketWebsite」を参照してください。https://docs.aws.amazon.com/goto/SdkForCpp/s3-2006-03-01/PutBucketWebsite

`PutObject`

次のコード例は、`PutObject` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「PutObject」を参照してください。

PutObjectAcl

次のコード例は、PutObjectAcl を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、「AWS SDK for C++ API リファレンス」の「PutObjectAcl」を参照してください。

`UploadPart`

次のコード例は、`UploadPart` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、「AWS SDK for C++ API リファレンス」の「UploadPart」を参照してください。

シナリオ

署名付き URL を作成する

次のコード例は、Amazon S3 の署名付き URL を作成し、オブジェクトをアップロードする方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

オブジェクトをダウンロードするための署名付き URL を生成します。

```
//! 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);
}
```

libcurl を使用してダウンロードします。

```
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;
}
```

オブジェクトをアップロードするための署名付き URL を生成します。

```
//! 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);
}
```

libcurl を使用してアップロードします。

```
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;
    }
}
```

サーバーレスアプリケーションを作成して写真を管理する

次のコード例では、ユーザーがラベルを使用して写真を管理できるサーバーレスアプリケーションを作成する方法について示しています。

SDK for C++

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードと設定および実行の手順については、GitHub で完全な例を参照してください。

この例のソースについて詳しくは、AWS コミュニティでブログ投稿を参照してください。

この例で使用されているサービス

- API ゲートウェイ
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Amazon S3 オブジェクトの整合性の操作

次のコード例は、S3 オブジェクトの整合性機能を操作する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

Amazon S3 オブジェクトの整合性機能を示すインタラクティブなシナリオを実行します。

```
//! 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();
    }
}

//! Routine which retrieves the string for a HASH_METHOD constant.
/*!
    \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";
    }
}

//! Routine that returns the ChecksumAlgorithm for a HASH_METHOD constant.
/*!
    \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 の詳細については、「AWS SDK for C++ API リファレンス」の以下のトピックを参照してください。
  - AbortMultipartUpload
  - CompleteMultipartUpload
  - CreateMultipartUpload
  - DeleteObject
  - GetObjectAttributes
  - PutObject
  - UploadPart

SDK for C++ を使用した Secrets Manager の例

次のコード例は、Secrets Manager AWS SDK for C++ で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- アクション

アクション

`GetSecretValue`

次のコード例は、`GetSecretValue` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、「AWS SDK for C++ API リファレンス」の「GetSecretValue」を参照してください。

SDK for C++ を使用した Amazon SES の例

次のコード例は、Amazon SES AWS SDK for C++ で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

シナリオは、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- アクション

- シナリオ

アクション

CreateReceiptFilter

次のコード例は、CreateReceiptFilter を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、「AWS SDK for C++ API リファレンス」の「CreateReceiptFilter」を参照してください。

`CreateReceiptRule`

次のコード例は、`CreateReceiptRule` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、「AWS SDK for C++ API リファレンス」の「CreateReceiptRule」を参照してください。

CreateReceiptRuleSet

次のコード例は、CreateReceiptRuleSet を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、「AWS SDK for C++ API リファレンス」の「CreateReceiptRuleSet」を参照してください。

`CreateTemplate`

次のコード例は、`CreateTemplate` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「CreateTemplate」を参照してください。

`DeleteIdentity`

次のコード例は、`DeleteIdentity` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteIdentity」を参照してください。

`DeleteReceiptFilter`

次のコード例は、`DeleteReceiptFilter` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、「AWS SDK for C++ API リファレンス」の「DeleteReceiptFilter」を参照してください。

DeleteReceiptRule

次のコード例は、DeleteReceiptRule を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、「AWS SDK for C++ API リファレンス」の「DeleteReceiptRule」を参照してください。

`DeleteReceiptRuleSet`

次のコード例は、DeleteReceiptRuleSet を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、「AWS SDK for C++ API リファレンス」の「DeleteReceiptRuleSet」を参照してください。

`DeleteTemplate`

次のコード例は、DeleteTemplate を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスのの「DeleteTemplate」を参照してください。

`GetTemplate`

次のコード例は、`GetTemplate` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、「AWS SDK for C++ API リファレンス」の「GetTemplate」を参照してください。

`ListIdentities`

次のコード例は、`ListIdentities` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「ListIdentities」を参照してください。

ListReceiptFilters

次のコード例は、ListReceiptFilters を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、「AWS SDK for C++ API リファレンス」の「ListReceiptFilters」を参照してください。

SendEmail

次のコード例は、SendEmail を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「SendEmail」を参照してください。

`SendTemplatedEmail`

次のコード例は、`SendTemplatedEmail` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、「AWS SDK for C++ API リファレンス」の「SendTemplatedEmail」を参照してください。

`UpdateTemplate`

次のコード例は、UpdateTemplate を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、「AWS SDK for C++ API リファレンス」の「UpdateTemplate」を参照してください。

`VerifyEmailIdentity`

次のコード例は、`VerifyEmailIdentity` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「VerifyEmailIdentity」を参照してください。

シナリオ

Aurora Serverless 作業項目トラッカーの作成

次のコード例は、Amazon Aurora Serverless データベースの作業項目を追跡し、Amazon Simple Email Service (Amazon SES) を使用してレポートを送信するウェブアプリケーションを作成する方法を示しています。

SDK for C++

Amazon Aurora Serverless データベースに保存されている作業項目を追跡して報告するウェブアプリケーションを作成する方法を説明します。

Amazon Aurora Serverless データをクエリし、React アプリケーションで使用するための C++ REST API の完全なソースコードと設定方法については、GitHub にある完全な例を参照してください。

この例で使用されているサービス

- Aurora
- Amazon RDS
- Amazon RDS データサービス
- Amazon SES

SDK for C++ を使用した Amazon SNS の例

次のコード例は、Amazon SNS AWS SDK for C++ で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

シナリオは、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- はじめに
- アクション
- シナリオ

はじめに

Hello Amazon SNS

次のコード例は、Amazon SNS の使用を開始する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

CMakeLists.txt CMake ファイルのコード。

```
# 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 ソースファイルのコード。

```
#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 の詳細については、「AWS SDK for C++ API リファレンス」の「ListTopics」を参照してください。

アクション

`CreateTopic`

次のコード例は、`CreateTopic` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「CreateTopic」を参照してください。

`DeleteTopic`

次のコード例は、`DeleteTopic` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteTopic」を参照してください。

GetSMSAttributes

次のコード例は、GetSMSAttributes を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「GetSMSAttributes」を参照してください。

GetTopicAttributes

次のコード例は、GetTopicAttributes を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「GetTopicAttributes」を参照してください。

`ListSubscriptions`

次のコード例は、`ListSubscriptions` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「ListSubscriptions」を参照してください。

`ListTopics`

次のコード例は、`ListTopics` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「ListTopics」を参照してください。

Publish

次のコード例は、Publish を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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();
}
```

属性を指定してメッセージを発行します。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「Publish」を参照してください。

SetSMSAttributes

次のコード例は、SetSMSAttributes を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

Amazon SNS を使用して DefaultSMSType 属性を設定する方法。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「SetSMSAttributes」を参照してください。

Subscribe

次のコード例は、Subscribe を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

E メールアドレスをトピックにサブスクライブします。

```
//! 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();
}
```

モバイルアプリケーションをトピックにサブスクライブします。

```
//! 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 関数をトピックにサブスクライブします。

```
//! 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 キューをトピックにサブスクライブします。

```
        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;
            }
```

トピックにフィルターでサブスクライブします。

```
        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::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 ostringstream;
                ostringstream << "Filter messages for \"" << queueName
                              << "\"'s subscription to the topic \""
                              << topicName << "\"?  (y/n)";

                // Add filter if user answers yes.
                if (askYesNoQuestion(ostringstream.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 の詳細については、AWS SDK for C++ API リファレンスの「Subscribe」を参照してください。

`Unsubscribe`

次のコード例は、`Unsubscribe` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
//! 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 の詳細については、AWS SDK for C++ API リファレンスの「Unsubscribe」を参照してください。

シナリオ

サーバーレスアプリケーションを作成して写真を管理する

次のコード例では、ユーザーがラベルを使用して写真を管理できるサーバーレスアプリケーションを作成する方法について示しています。

SDK for C++

Amazon Rekognition を使用して画像内のラベルを検出し、保存して後で取得できるようにする写真アセット管理アプリケーションの開発方法を示します。

完全なソースコードと設定および実行の手順については、GitHub で完全な例を参照してください。

この例のソースについて詳しくは、AWS コミュニティでブログ投稿を参照してください。

この例で使用されているサービス

- API ゲートウェイ
- DynamoDB
- Lambda
- Amazon Rekognition

- Amazon S3
- Amazon SNS

SMS テキストメッセージを発行する

次のコードサンプルは、Amazon SNS を使用して SMS メッセージを発行する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
/**
 * 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 の詳細については、AWS SDK for C++ API リファレンスの「Publish」を参照してください。

メッセージをキューに発行する

次のコード例は、以下の操作方法を示しています。

- トピック (FIFO または非 FIFO) を作成します。
- フィルターを適用するオプションを使用して、複数のキューをトピックにサブスクライブします。
- メッセージをトピックに発行します。
- 受信したメッセージのキューをポーリングします。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 ostringstream;
                ostringstream << "Filter messages for \"" << queueName
                            << "\"'s subscription to the topic \""
                            << topicName << "\"?  (y/n)";

                // Add filter if user answers yes.
                if (askYesNoQuestion(ostringstream.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 の詳細については、「AWS SDK for C++ API リファレンス」の以下のトピックを参照してください。
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

SDK for C++ を使用した Amazon SQS の例

次のコード例は、Amazon SQS AWS SDK for C++ で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

シナリオは、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コンテキスト内でコードを設定および実行する方法の手順を確認できます。

トピック

- はじめに
- アクション
- シナリオ

はじめに

Hello Amazon SQS

次のコード例は、Amazon SQS の使用を開始する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

CMakeLists.txt CMake ファイルのコード。

```
# 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_CPY_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 ソースファイルのコード。

```
#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 の詳細については、「AWS SDK for C++ API リファレンス」の「ListQueues」を参照してください。

アクション

`ChangeMessageVisibility`

次のコード例は、`ChangeMessageVisibility` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「ChangeMessageVisibility」を参照してください。

`CreateQueue`

次のコード例は、`CreateQueue` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「CreateQueue」を参照してください。

`DeleteMessage`

次のコード例は、`DeleteMessage` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteMessage」を参照してください。

`DeleteMessageBatch`

次のコード例は、`DeleteMessageBatch` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteMessageBatch」を参照してください。

`DeleteQueue`

次のコード例は、`DeleteQueue` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「DeleteQueue」を参照してください。

`GetQueueAttributes`

次のコード例は、`GetQueueAttributes` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、「AWS SDK for C++ API リファレンス」の「GetQueueAttributes」を参照してください。

GetQueueUrl

次のコード例は、GetQueueUrl を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「GetQueueUrl」を参照してください。

`ListQueues`

次のコード例は、`ListQueues` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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;
}
```

- API の詳細については、AWS SDK for C++ API リファレンスの「ListQueues」を参照してください。

`ReceiveMessage`

次のコード例は、`ReceiveMessage` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「ReceiveMessage」を参照してください。

SendMessage

次のコード例は、SendMessage を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 の詳細については、AWS SDK for C++ API リファレンスの「SendMessage」を参照してください。

`SetQueueAttributes`

次のコード例は、`SetQueueAttributes` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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();
}
```

デッドレターキューを設定します。

```
        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();
}
```

ロングポーリングを使用するように Amazon SQS キューを設定します。

```
        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 の詳細については、「AWS SDK for C++ API リファレンス」の「SetQueueAttributes」を参照してください。

シナリオ

メッセージをキューに発行する

次のコード例は、以下の操作方法を示しています。

- トピック (FIFO または非 FIFO) を作成します。
- フィルターを適用するオプションを使用して、複数のキューをトピックにサブスクライブします。
- メッセージをトピックに発行します。
- 受信したメッセージのキューをポーリングします。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
        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 ostringstream;
                ostringstream << "Filter messages for \"" << queueName
                              << "\"'s subscription to the topic \""
                              << topicName << "\"?  (y/n)";
```

```
                // Add filter if user answers yes.
                if (askYesNoQuestion(ostringstream.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 の詳細については、「AWS SDK for C++ API リファレンス」の以下のトピックを参照してください。
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publish
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

AWS STS SDK for C++ を使用した例

次のコード例は、 AWS SDK for C++ で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています AWS STS。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- アクション

アクション

`AssumeRole`

次のコード例は、`AssumeRole` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 の詳細については、AWS SDK for C++ API リファレンスの「AssumeRole」を参照してください。

SDK for C++ を使用した Amazon Transcribe Streaming の例

次のコード例は、Amazon Transcribe Streaming AWS SDK for C++ で を使用してアクションを実行し、一般的なシナリオを実装する方法を示しています。

アクションはより大きなプログラムからのコードの抜粋であり、コンテキスト内で実行する必要があります。アクションは個々のサービス機能を呼び出す方法を示していますが、コンテキスト内のアクションは、関連するシナリオで確認できます。

シナリオは、1 つのサービス内から、または他の AWS のサービスと組み合わせて複数の関数を呼び出し、特定のタスクを実行する方法を示すコード例です。

各例には完全なソースコードへのリンクが含まれており、コードの設定方法と実行方法に関する手順を確認できます。

トピック

- アクション
- シナリオ

アクション

`StartStreamTranscription`

次のコード例は、`StartStreamTranscription` を使用する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 transcripted.
        // 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 の詳細については、「AWS SDK for C++ API リファレンス」の「StartStreamTranscription」を参照してください。

シナリオ

音声ファイルの文字起こし

次のコード例は、Amazon Transcribe ストリーミングを使用してソースオーディオファイルの文字起こしを生成する方法を示しています。

SDK for C++

> **Note**
>
> GitHub には、その他のリソースもあります。用例一覧を検索し、AWS コード例リポジトリでの設定と実行の方法を確認してください。

```
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 transcripted.
        // 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 の詳細については、「AWS SDK for C++ API リファレンス」の「StartStreamTranscription」を参照してください。

AWS SDK for C++ のセキュリティ

クラウドセキュリティは Amazon Web Services (AWS) の最優先事項です。 AWS のお客様は、セキュリティを非常に重視する組織の要件を満たせるように構築されたデータセンターとネットワークアーキテクチャーから利点を得ます。セキュリティは、 AWS とお客様の間の責任共有です。責任共有モデルでは、これをクラウドのセキュリティおよびクラウド内のセキュリティとして説明しています。

クラウドのセキュリティ – AWS クラウドで提供されるすべてのサービスを実行するインフラストラクチャ AWS を保護し、安全に使用できるサービスを提供します。における当社のセキュリティ責任は最優先事項であり AWS、当社のセキュリティの有効性は、AWS コンプライアンスプログラムの一環としてサードパーティーの監査者によって定期的にテストおよび検証されます。

クラウド内のセキュリティ – お客様の責任は、使用している AWS サービス、データの機密性、組織の要件、適用される法律や規制などのその他の要因によって決まります。

この AWS 製品またはサービスは、サポートする特定の Amazon Web Services (AWS) サービスを通じて責任共有モデルに従います。 AWS サービスセキュリティ情報については、AWS 「サービスセキュリティドキュメント」ページとAWS 、コンプライアンスプログラムによる AWS コンプライアンスの取り組みの対象となるサービスを参照してください。

トピック

- AWS SDK for C++ でのデータ保護
- Identity and Access Management
- この AWS 製品またはサービスのコンプライアンス検証
- この AWS 製品またはサービスの耐障害性
- この AWS 製品またはサービスのインフラストラクチャセキュリティ
- AWS SDK for C++ での最小 TLS バージョンの適用
- Amazon S3 暗号化クライアントの移行 (V1 から V2)
- Amazon S3 暗号化クライアントの移行 (V2 から V3)

AWS SDK for C++ でのデータ保護

AWS 責任共有モデルは、AWS SDK for C++ でのデータ保護に適用されます。このモデルで説明されているように、AWS は、AWS クラウドのすべてを実行するグローバルインフラストラクチャを

保護する責任があります。ユーザーは、このインフラストラクチャでホストされるコンテンツに対する管理を維持する責任があります。また、使用する「AWS のサービス」のセキュリティ設定と管理タスクもユーザーの責任となります。データプライバシーの詳細については、データプライバシーに関するよくある質問を参照してください。欧州でのデータ保護の詳細については、AWS セキュリティブログに投稿された AWS 責任共有モデルおよび GDPR のブログ記事を参照してください。

データを保護するため、「AWS アカウント」 認証情報を保護し、「AWS IAM アイデンティティセンター」 または 「AWS Identity and Access Management」 (IAM) を使用して個々のユーザーをセットアップすることをお勧めします。この方法により、それぞれのジョブを遂行するために必要な権限のみが各ユーザーに付与されます。また、次の方法でデータを保護することもお勧めします:

- 各アカウントで多要素認証 (MFA) を使用します。
- SSL/TLS を使用して 「AWS」 リソースと通信します。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- AWS CloudTrail で API とユーザーアクティビティロギングを設定します。CloudTrail 証跡を使用して AWS アクティビティをキャプチャする方法については、「AWS CloudTrail ユーザーガイド」の「CloudTrail 証跡の使用」を参照してください。
- AWS のサービス 内のすべてのデフォルトセキュリティコントロールに加え、AWS 暗号化ソリューションを使用します。
- Amazon Macie などの高度な管理されたセキュリティサービスを使用します。これらは、Amazon S3 に保存されている機密データの検出と保護を支援します。
- コマンドラインインターフェイスまたは API を使用して 「AWS」 にアクセスする際に FIPS 140-3 検証済みの暗号化モジュールが必要な場合は、FIPS エンドポイントを使用します。利用可能な FIPS エンドポイントの詳細については、「連邦情報処理規格 (FIPS) 140-3」を参照してください。

お客様の E メールアドレスなどの極秘または機密情報は、タグ、または名前フィールドなどの自由形式のテキストフィールドに配置しないことを強くお勧めします。これには、コンソール、API、AWS CLI、または AWS SDK で、SDK for C++ やその他の AWS のサービス を使用する場合も含まれます。タグ、または名前に使用される自由記述のテキストフィールドに入力したデータは、請求または診断ログに使用される場合があります。外部サーバーに URL を提供する場合、そのサーバーへのリクエストを検証できるように、認証情報を URL に含めないことを強くお勧めします。

Identity and Access Management

AWS Identity and Access Management (IAM) は、管理者が AWS リソースへのアクセスを安全に制御するために役立つ AWS のサービスです。IAM 管理者は、誰を認証 (サインイン) し、誰に AWS リソースの使用を許可する (権限を持たせる) かを制御します。IAM は、無料で使用できる AWS のサービスです。

トピック

- 対象者
- アイデンティティによる認証
- ポリシーを使用したアクセス権の管理
- AWS のサービスと IAM の連携の仕組み
- AWS ID とアクセスのトラブルシューティング

対象者

AWS Identity and Access Management (IAM) の用途は、AWS で行う作業によって異なります。

サービスユーザー – ジョブを実行するために AWS のサービスを使用する場合は、管理者が必要なアクセス許可と認証情報を用意します。作業を実行するためにさらに多くの AWS 機能を使用するとき、追加の権限が必要になる場合があります。アクセスの管理方法を理解すると、管理者から適切な権限をリクエストするのに役に立ちます。AWS の機能にアクセスできない場合は、「AWS ID とアクセスのトラブルシューティング」を参照するか、使用している AWS のサービスのユーザーガイドを参照してください。

サービス管理者 - 社内の AWS リソースを担当している場合は、通常、AWS への完全なアクセスがあります。サービスのユーザーがどの AWS 機能やリソースにアクセスするかを決めるのは管理者の仕事です。その後、IAM 管理者にリクエストを送信して、サービスユーザーの権限を変更する必要があります。このページの情報を点検して、IAM の基本概念を理解してください。AWS で IAM を利用する方法の詳細については、使用している AWS のサービスのユーザーガイドを参照してください。

IAM 管理者 - 管理者は、AWS へのアクセスを管理するポリシーの書き込み方法の詳細について確認する場合があります。IAM で使用できる AWS ID ベースのポリシーの例を表示するには、使用している AWS のサービスのユーザーガイドを参照してください。

アイデンティティによる認証

認証とは、アイデンティティ認証情報を使用して AWS にサインインする方法です。ユーザーは、AWS アカウントのルートユーザー、IAM ユーザーとして、または IAM ロールを引き受けることによって、認証される必要があります。

AWS IAM アイデンティティセンター (IAM Identity Center)、シングルサインオン認証、Google/Facebook 認証情報などの ID ソースからの認証情報を使用して、フェデレーテッドアイデンティティとしてサインインできます。サインインの詳細については、「AWS サインイン ユーザーガイド」の「How to sign in to your AWS アカウント」を参照してください。

プログラムによるアクセスの場合、AWS はリクエストに暗号で署名するための SDK と CLI を提供します。詳細については、「IAM ユーザーガイド」の「API リクエストに対する AWS Signature Version 4」を参照してください。

AWS アカウント のルートユーザー

AWS アカウントを作成すると、すべての AWS のサービスとリソースに対する完全なアクセス権を持つ AWS アカウントルートユーザーと呼ばれる 1 つのサインイン ID を使用して開始します。日常的なタスクには、ルートユーザーを使用しないことを強くお勧めします。ルートユーザー認証情報を必要とするタスクについては、「IAM ユーザーガイド」の「ルートユーザー認証情報が必要なタスク」を参照してください。

フェデレーテッドアイデンティティ

ベストプラクティスでは、人間のユーザーが一時的な認証情報を使用して AWS のサービス にアクセスする際、アイデンティティプロバイダーとのフェデレーションを使用することが求められます。

フェデレーテッドアイデンティティは、エンタープライズディレクトリ、ウェブ ID プロバイダー、Directory Service のユーザーであり、ID ソースからの認証情報を使用して AWS のサービスにアクセスするユーザーです。フェデレーティッドアイデンティティは、一時的な認証情報を提供するロールを引き受けます。

アクセスを一元管理する場合は、AWS IAM アイデンティティセンター をお勧めします。詳細については、「AWS IAM アイデンティティセンター ユーザーガイド」の「What is IAM Identity Center?」を参照してください。

IAM ユーザーとグループ

IAM ユーザーは、1 人のユーザーまたは 1 つのアプリケーションに対して特定の許可を持つアイデンティティです。長期認証情報を持つ IAM ユーザーの代わりに一時的な認証情報を使用することを

お勧めします。詳細は「IAM ユーザーガイド」の「人間のユーザーが一時的な認証情報を使用して AWS にアクセスするには ID プロバイダーとのフェデレーションの使用が必要です」を参照してください。

IAM グループは IAM ユーザーのコレクションを指定し、大量のユーザーのアクセス許可の管理を容易にします。詳細については、「IAM ユーザーガイド」の「IAM ユーザーに関するユースケース」を参照してください。

IAM ロール

IAM ロールは、一時的な認証情報を提供する特定のアクセス許可を持つ ID です。ユーザーから IAM ロールに切り替える (コンソール)、または AWS CLI や AWS API オペレーションを呼び出すことで、ロールを引き受けることができます。詳細については、「IAM ユーザーガイド」の「ロールを引き受けることができない」を参照してください。

IAM ロールは、フェデレーションユーザーアクセス、一時的な IAM ユーザーアクセス許可、クロスアカウントアクセス、クロスサービスアクセス、および Amazon EC2 で実行されているアプリケーションに役立ちます。詳細については、「IAM ユーザーガイド」の「IAM でのクロスアカウントリソースアクセス」を参照してください。

ポリシーを使用したアクセス権の管理

AWS でアクセスを制御するには、ポリシーを作成して AWS ID またはリソースにアタッチします。ポリシーはアイデンティティやリソースに関連付けて、これらのアクセス許可を定義します。AWS は、プリンシパルがリクエストを行う際に、それらのポリシーを評価します。大半のポリシーは JSON ドキュメントとして AWS に保存されます。JSON ポリシードキュメントの詳細については、「IAM ユーザーガイド」の「JSON ポリシー概要」を参照してください。

ポリシーを使用して、管理者は、どのプリンシパルがどのリソースに対してどの条件でアクションを実行できるかを定義することで、誰が何にアクセスできるかを指定します。

デフォルトでは、ユーザーやロールに権限はありません。IAM 管理者は IAM ポリシーを作成し、ユーザーが引き受けることができるロールに追加します。IAM ポリシーは、オペレーションの実行方法を問わずアクセス許可を定義します。

アイデンティティベースポリシー

アイデンティティベースのポリシーは、アイデンティティ (ユーザー、グループ、ロール) にアタッチする JSON アクセス許可ポリシードキュメントです。これらのポリシーは、アイデンティティが実行できるアクション、リソース、および条件を制御します。アイデンティティベースポリシーの作

成方法については、「IAM ユーザーガイド」の「カスタマー管理ポリシーでカスタム IAM アクセス許可を定義する」を参照してください。

ID ベースのポリシーは、インラインポリシー (単一の ID に直接埋め込む) または管理ポリシー (複数の ID にアタッチされたスタンドアロンポリシー) にすることができます。管理ポリシーおよびインラインポリシーのいずれかを選択する方法については、「IAM ユーザーガイド」の「管理ポリシーとインラインポリシーのいずれかを選択する」を参照してください。

リソースベースのポリシー

リソースベースのポリシーは、リソースにアタッチする JSON ポリシードキュメントです。例としては、IAM ロールの信頼ポリシーや Amazon S3 バケットポリシーなどがあります。リソースベースのポリシーをサポートするサービスでは、サービス管理者はポリシーを使用して特定のリソースへのアクセスをコントロールできます。リソースベースのポリシーで、プリンシパルを指定する必要があります。

リソースベースのポリシーは、そのサービス内にあるインラインポリシーです。リソースベースのポリシーでは IAM の AWSマネージドポリシーは使用できません。

アクセスコントロールリスト (ACL)

アクセスコントロールリスト (ACL) は、どのプリンシパル (アカウントメンバー、ユーザー、またはロール) がリソースにアクセスするための許可を持つかを制御します。ACL はリソースベースのポリシーに似ていますが、JSON ポリシードキュメント形式は使用しません。

Amazon S3、AWS WAF、および Amazon VPC は、ACL をサポートするサービスの例です。ACL の詳細については、「Amazon Simple Storage Service デベロッパーガイド」の「アクセスコントロールリスト (ACL) の概要」を参照してください。

他のポリシータイプ

AWS は、より一般的なポリシータイプで付与された最大の権限を設定できる、追加のポリシータイプをサポートしています。

- アクセス許可の境界 – アイデンティティベースのポリシーで IAM エンティティに付与することのできる許可の上限を設定します。詳細については、「IAM ユーザーガイド」の「IAM エンティティのアクセス許可境界」を参照してください。
- サービスコントロールポリシー (SCP) – AWS Organizations において組織または組織単位のアクセス許可の上限を指定します。詳細については、AWS Organizations ユーザーガイドの「サービスコントロールポリシー」を参照してください。

- リソースコントロールポリシー (RCP) – アカウント内のリソースで利用できるアクセス許可の上限を定義します。詳細については、「AWS Organizations ユーザーガイド」の「リソースコントロールポリシー (RCP)」を参照してください。
- セッションポリシー – ロールまたはフェデレーションユーザーの一時セッションを作成する際にパラメータとして渡す高度なポリシーです。詳細については、「IAM ユーザーガイド」の「セッションポリシー」を参照してください。

複数のポリシータイプ

1 つのリクエストに複数のタイプのポリシーが適用されると、結果として作成される権限を理解するのがさらに難しくなります。複数のポリシータイプが関連するとき、リクエストを許可するかどうかを AWS が決定する方法の詳細については、「IAM ユーザーガイド」の「ポリシーの評価ロジック」を参照してください。

AWS のサービスと IAM の連携の仕組み

AWS のサービスが IAM のほとんどの機能と連携する仕組みの概要については、「IAM ユーザーガイド」の「IAM と連携する AWS のサービス」を参照してください。

特定の AWS のサービスで IAM を使用する方法については、該当するサービスのユーザーガイドでセキュリティに関するセクションを参照してください。

AWS ID とアクセスのトラブルシューティング

以下の情報は、AWS と IAM の使用に伴って発生する可能性がある一般的な問題の診断や修復に役立ちます。

トピック

- AWS でアクションを実行する権限がない
- iam:PassRole を実行する権限がない
- 自分の AWS アカウント 以外のユーザーに AWS リソースへのアクセスを許可したい

AWS でアクションを実行する権限がない

あるアクションを実行する権限がないというエラーが表示された場合、そのアクションを実行できるようにポリシーを更新する必要があります。

次のエラー例は、mateojackson IAM ユーザーがコンソールを使用して、ある *my-example-widget* リソースに関する詳細情報を表示しようとしたことを想定して、その際に必要なawes:*GetWidget* アクセス許可を持っていない場合に発生するものです。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 awes:GetWidget on resource: my-example-widget
```

この場合、awes:*GetWidget* アクションを使用して *my-example-widget*リソースへのアクセスを許可するように、mateojackson ユーザーのポリシーを更新する必要があります。

サポートが必要な場合は、AWS 管理者に問い合わせてください。サインイン認証情報を提供した担当者が管理者です。

iam:PassRole を実行する権限がない

iam:PassRole アクションを実行する権限がないというエラーが表示された場合は、ポリシーを更新して AWS にロールを渡すことができるようにする必要があります。

一部の AWS のサービスでは、新しいサービスロールやサービスリンクロールを作成せずに、既存のロールをサービスに渡すことができます。そのためには、サービスにロールを渡す権限が必要です。

以下の例のエラーは、marymajor という IAM ユーザーがコンソールを使用して AWS でアクションを実行しようする場合に発生します。ただし、このアクションをサービスが実行するには、サービスロールから付与された権限が必要です。メアリーには、ロールをサービスに渡す許可がありません。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

この場合、Mary のポリシーを更新してメアリーに iam:PassRole アクションの実行を許可する必要があります。

サポートが必要な場合は、AWS 管理者に問い合わせてください。サインイン認証情報を提供した担当者が管理者です。

自分の AWS アカウント 以外のユーザーに AWS リソースへのアクセスを許可したい

他のアカウントのユーザーや組織外のユーザーが、リソースへのアクセスに使用できるロールを作成できます。ロールの引き受けを委託するユーザーを指定できます。リソースベースのポリシーまたはアクセスコントロールリスト (ACL) をサポートするサービスの場合、それらのポリシーを使用して、リソースへのアクセスを付与できます。

詳細については、以下を参照してください。

- AWS がこれらの機能をサポートしているかどうかを確認するには、「AWS のサービスと IAM の連携の仕組み」をご参照ください。
- 所有している AWS アカウント 全体のリソースへのアクセス権を提供する方法については、IAM ユーザーガイドの所有している別の AWS アカウント アカウントへのアクセス権を IAM ユーザーに提供を参照してください。
- サードパーティーの AWS アカウント にリソースへのアクセス権を提供する方法については、「IAM ユーザーガイド」の「サードパーティーが所有する AWS アカウント へのアクセス権を付与する」を参照してください。
- ID フェデレーションを介してアクセスを提供する方法については、「IAM ユーザーガイド」の「外部で認証されたユーザー (ID フェデレーション) へのアクセスの許可」を参照してください。
- クロスアカウントアクセスにおけるロールとリソースベースのポリシーの使用方法の違いについては、「IAM ユーザーガイド」の「IAM でのクロスアカウントのリソースへのアクセス」を参照してください。

この AWS 製品またはサービスのコンプライアンス検証

AWS のサービス が特定のコンプライアンスプログラムの対象であるかどうかを確認するには、「コンプライアンスプログラムによる対象範囲内の AWS のサービス」をご覧いただき、関心のあるコンプライアンスプログラムを選択してください。一般的な情報については、「AWSコンプライアンスプログラム」を参照してください。

AWS Artifact を使用して、サードパーティーの監査レポートをダウンロードできます。詳細については、「AWS Artifact でレポートをダウンロードする」を参照してください。

AWS のサービス を使用する際のお客様のコンプライアンス責任は、お客様のデータの機密性や貴社のコンプライアンス目的、適用可能な法律および規制によって決定されます。AWS のサービスを使用する際のコンプライアンス責任の詳細については、「AWS セキュリティドキュメント」を参照してください。

この AWS 製品またはサービスは、サポートしている特定の Amazon Web Services (AWS) のサービスを通じて、責任共有モデルに従います。AWS サービスのセキュリティ情報については、AWS のサービスのセキュリティに関するドキュメントページとAWS コンプライアンスプログラムごとのコンプライアンスの取り組みの対象となる AWS のサービスに関するページを参照してください。

この AWS 製品またはサービスの耐障害性

AWS のグローバルインフラストラクチャは AWS リージョン とアベイラビリティーゾーンを中心として構築されます。

AWS リージョン は、物理的に独立・隔離されたアベイラビリティーゾーンがあり、低レイテンシー、高スループット、そして高冗長性のネットワークで接続されています。

アベイラビリティーゾーンでは、ゾーン間で中断することなく自動的にフェールオーバーするアプリケーションとデータベースを設計および運用することができます。アベイラビリティーゾーンは、従来の単一または複数のデータセンターインフラストラクチャよりも可用性、耐障害性、および拡張性が優れています。

AWS リージョンとアベイラビリティーゾーンの詳細については、「AWS グローバルインフラストラクチャ」を参照してください。

この AWS 製品またはサービスは、サポートしている特定の Amazon Web Services (AWS) のサービスを通じて、責任共有モデルに従います。AWS サービスのセキュリティ情報については、AWS のサービスのセキュリティに関するドキュメントページとAWS コンプライアンスプログラムごとのコンプライアンスの取り組みの対象となる AWS のサービスに関するページを参照してください。

この AWS 製品またはサービスのインフラストラクチャセキュリティ

この AWS 製品またはサービスは、マネージドサービスを使用しているため、AWS グローバルネットワークセキュリティによって保護されています。AWS セキュリティサービスと AWS がインフラストラクチャを保護する方法については、「AWS クラウドセキュリティ」を参照してください。インフラストラクチャセキュリティのベストプラクティスにより AWS 環境を設計するには、「セキュリティの柱 - AWS Well-Architected フレームワーク」の「インフラストラクチャ保護」を参照してください。

AWS の公開された API コールを使用し、ネットワークを介してこの AWS 製品またはサービスにアクセスします。クライアントは以下をサポートする必要があります。

- Transport Layer Security (TLS)。TLS 1.2 が必須で、TLS 1.3 をお勧めします。
- DHE (楕円ディフィー・ヘルマン鍵共有) や ECDHE (楕円曲線ディフィー・ヘルマン鍵共有) などの完全前方秘匿性 (PFS) による暗号スイート。これらのモードは Java 7 以降など、ほとんどの最新システムでサポートされています。

また、リクエストにはアクセスキー ID と、IAM プリンシパルに関連付けられているシークレットアクセスキーを使用して署名する必要があります。または、AWS Security Token Service (AWS STS) を使用して、一時的セキュリティ認証情報を生成し、リクエストに署名することもできます。

この AWS 製品またはサービスは、サポートしている特定の Amazon Web Services (AWS) のサービスを通じて、責任共有モデルに従います。AWS サービスのセキュリティ情報については、AWS のサービスのセキュリティに関するドキュメントページとAWS コンプライアンスプログラムごとのコンプライアンスの取り組みの対象となる AWS のサービスに関するページを参照してください。

AWS SDK for C++ での最小 TLS バージョンの適用

AWS サービスと通信時のセキュリティを強化するには、TLS 1.2 以降を使用するように SDK for C++ を設定する必要があります。TLS 1.3 の使用をお勧めします。

AWS SDK for C++ はクロスプラットフォームのライブラリです。任意のプラットフォームでアプリケーションをビルドして実行できます。プラットフォームによって基盤 HTTP クライアントが異なる場合があります。

デフォルトでは、macOS、Linux、Android、およびその他の非 Windows プラットフォームは libcurl を使用します。libcurl のバージョンが 7.34.0 より後の場合、基盤 HTTP クライアントで使用される TLS の最小バージョンは 1.0 です。

Windows では、デフォルトのライブラリは WinHttp です。Windows は、TLS 1.0、1.1、1.2、1.3 の中から使用する実際のプロトコルを決定します。Windows では、WinINet と IXMLHttpRequest2 という 2 つの選択肢もあります。CMake 中および実行時にデフォルトライブラリを置き換えるようにアプリケーションを設定できます。これら 2 つの HTTP クライアントに対しても、Windows が使用する安全なプロトコルを決定します。

AWS SDK for C++ は、デフォルトの HTTP クライアントをオーバーライドする柔軟性も提供します。例えば、カスタム HTTP クライアントファクトリを使用して、libcurl を適用したり、任意の HTTP クライアントを使用したりできます。そのため、TLS 1.2 を最小バージョンとして使用するには、使用している HTTP クライアントライブラリを把握しておく必要があります。

libcurl を使用してすべてのプラットフォームで特定の TLS バージョンを適用する

このセクションでは、AWS SDK for C++ が HTTP プロトコルのサポートに libcurl を依存関係として使用していることを前提としています。TLS バージョンを明示的に指定するには、libcurl のバージョ

ンが 7.34.0 以降である必要があります。さらに、AWS SDK for C++ のソースコードを変更してから再ビルドする必要がある場合があります。

これらの作業の手順を次に示します。

libcurl で TLS 1.2 を適用するには

1. libcurl のインストールがバージョン 7.34.0 以降であることを確認します。
2. AWS SDK for C++ のソースコードを GitHub からダウンロードします。
3. `aws-cpp-sdk-core/source/http/curl/CurlHttpClient.cpp` を開き、次のコード行を見つけます。

```
#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. 必要に応じて、関数呼び出しの最後のパラメータを次のように変更します。

```
#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. 前述のコード変更を行った場合は、https://github.com/aws/aws-sdk-cpp#building-the-sdk の手順に従って AWS SDK for C++ をビルドしてインストールします。
6. アプリケーションのサービスクライアントで、まだ有効になっていない場合は、クライアント設定で `verifySSL` を有効にします。

libcurl で TLS 1.3 を適用するには

TLS 1.3 を適用するには、前述のセクションの手順に従って、`CURL_SSLVERSION_TLSv1_2` の代わりに `CURL_SSLVERSION_TLSv1_3` オプションを設定します。

Windows で特定の TLS バージョンを適用する

以下の手順では、WinHttp WinINet 、または IXMLHTTPRequest2 を使用して TLS 1.2 または TLS 1.3 を適用する方法を示しています。

前提条件: Windows の TLS サポートを確認する

- https://docs.microsoft.com/en-us/windows/win32/secauthn/protocols-in-tls-ssl–schannel-ssp- の説明に従って、システムで使用可能な TLS プロトコルバージョンのサポートを確認します。
- Windows 7 SP1 または Windows Server 2008 R2 SP1 上で実行している場合は、レジストリで TLS 1.2 のサポートが有効になっていることを確認する必要があります。詳細については、https://docs.microsoft.com/ja-jp/windows-server/security/tls/tls-registry-settings#tls-12 を参照してください。以前のディストリビューションを実行している場合は、オペレーティングシステムをアップグレードする必要があります。

WinHttp で TLS 1.2 または TLS 1.3 を適用するには

WinHttp には、使用可能で安全なプロトコルを明示的に設定する API が用意されています。ただし、これを実行時に設定できるようにするには、AWS SDK for C++ のソースコードを変更してから再ビルドする必要があります。

1. AWS SDK for C++ のソースコードを GitHub から ダウンロードします。
2. `aws-cpp-sdk-core/source/http/windows/WinHttpSyncHttpClient.cpp` を開き、次のコード行を見つけます。

```
#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());
}
```

`WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_3` オプションフラグは、TLS 1.3 が現在のビルドシステムに存在する場合に定義されます。詳細については、Microsoft のウェブサイトで

「WINHTTP_OPTION_SECURE_PROTOCOLS」および「TLS プロトコルバージョンのサポート」を参照してください。

3. 次のいずれかを選択します。
   - TLS 1.2 を適用するには:

     `#else` ディレクティブで、次のように `flags` 変数の値を変更します。

     ```
     DWORD flags = WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_2;
     ```

   - TLS 1.3 を適用するには:

     `#if defined(WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_3)` ディレクティブで、次のように `flags` 変数の値を変更します。

     ```
     DWORD flags = WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_3;
     ```

4. 前述のコード変更を行った場合は、https://github.com/aws/aws-sdk-cpp#building-the-sdk の手順に従って AWS SDK for C++ をビルドしてインストールします。
5. アプリケーションのサービスクライアントで、まだ有効になっていない場合は、クライアント設定で `verifySSL` を有効にします。

WinINet と IXMLHTTPRequest2 で TLS 1.2 を適用するには

WinINet および IXMLHTTPRequest2 ライブラリには、安全なプロトコルを指定する API はありません。そのため、AWS SDK for C++ はオペレーティングシステムのデフォルトを使用します。以下の手順に示すように、Windows レジストリを更新して TLS 1.2 の使用を適用できます。ただし、この結果は Schannel に依存するすべてのアプリケーションに影響するグローバルな変更であることに注意してください。

1. レジストリエディターを開き、`Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols` に移動します。
2. まだ存在しない場合は、`TLS 1.0`,、`TLS 1.1`、`TLS 1.2` のサブキーを作成します。
3. 各サブキーの下に、`Client` サブキーと `Server` サブキーを作成します。
4. 次のキーと値を作成します。

```
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 だけが 1 に設定されているキーであることに注意してください。このキーを 1 に設定することで、TLS 1.2 のみが許容される安全なプロトコルとして適用されます。

WinINet と IXMLHTTPRequest2 で TLS 1.3 を適用するには

WinINet および IXMLHTTPRequest2 ライブラリには、安全なプロトコルを指定する API はありません。そのため、AWS SDK for C++ はオペレーティングシステムのデフォルトを使用します。以下の手順に示すように、Windows レジストリを更新して TLS 1.3 の使用を適用できます。ただし、この結果は Schannel に依存するすべてのアプリケーションに影響するグローバルな変更であることに注意してください。

1. レジストリエディターを開き、Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols に移動します。
2. まだ存在しない場合は、TLS 1.0,、TLS 1.1、TLS 1.2、TLS 1.3 のサブキーを作成します。
3. 各サブキーの下に、Client サブキーと Server サブキーを作成します。
4. 次のキーと値を作成します。

```
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` だけが 1 に設定されているキーであることに注意してください。このキーを 1 に設定することで、TLS 1.3 のみが許容される安全なプロトコルとして適用されます。

Amazon S3 暗号化クライアントの移行 (V1 から V2)

> **Note**
>
> Amazon S3 暗号化クライアントの V2 を使用していて、V3 に移行する場合は、「」を参照してください Amazon S3 暗号化クライアントの移行 (V2 から V3)。

このトピックでは、アプリケーションで使用している Amazon Simple Storage Service (Amazon S3) 暗号化クライアントをバージョン 1 (V1) からバージョン 2 (V2) に移行し、移行プロセス全体でアプリケーションの可用性を確保する方法について説明します。

移行の概要

この移行は 2 つのフェーズから構成されます。

1. 新しいフォーマットを読み取るために既存のクライアントを更新します。まず、更新されたバージョンの AWS SDK for C++ をアプリケーションにデプロイします。これにより、既存の V1 暗号化クライアントが、新しい V2 クライアントによって書き込まれたオブジェクトを復号できるようになります。アプリケーションで AWS SDKs、各 SDK を個別にアップグレードする必要があります。

2. 暗号化および復号クライアントを V2 に移行します。すべての V1 暗号化クライアントが新しいフォーマットを読み取ることができるようになったら、既存の暗号化および復号化クライアントをそれぞれの V2 バージョンに移行できます。

新しいフォーマットを読み取るために既存のクライアントを更新する

まず、既存のクライアントを最新の SDK リリースに更新する必要があります。このステップを完了すると、アプリケーションの V1 クライアントは、アプリケーションのコードベースを更新することなく、V2 暗号化クライアントによって暗号化されたオブジェクトを復号できます。

の最新バージョンをビルドしてインストールする AWS SDK for C++

ソースから SDK を利用するアプリケーション

ソース AWS SDK for C++ から を構築してインストールする場合は、GitHub の から SDK ソースをダウンロードまたはクローンaws/aws-sdk-cppします。その後、通常のビルドおよびインストール手順を繰り返します。

1.8.x より前のバージョン AWS SDK for C++ からアップグレードする場合は、各メジャーバージョンで導入された重大な変更について、この CHANGELOG を参照してください。をビルドおよびインストールする方法の詳細については AWS SDK for C++、「」を参照してください ソースコードからの AWS SDK for C++ の入手。

Vcpkg から SDK を利用するアプリケーション

アプリケーションで Vcpkg を使用して SDK の更新を追跡している場合は、既存の Vcpkg アップグレード方法を使用して SDK を最新バージョンにアップグレードするだけです。バージョンがリリースされてからパッケージマネージャーで使用可能になるまでに遅延があることに注意してください。最新バージョンはソースからインストールすることで常に使用できます。

次のコマンドを実行してパッケージ aws-sdk-cpp をアップグレードできます。

```
vcpkg upgrade aws-sdk-cpp
```

次に、パッケージ aws-sdk-cpp のバージョンを確認します。

```
vcpkg list aws-sdk-cpp
```

バージョンは 1.8.24 以降である必要があります。

で Vcpkg を使用する方法の詳細については AWS SDK for C++、「」を参照してください パッケージマネージャーからの AWS SDK for C++ の取得。

アプリケーションのビルド、インストール、デプロイする

アプリケーションが と静的にリンクされている場合 AWS SDK for C++、アプリケーションにコードの変更は必要ありませんが、最新の SDK の変更を使用するためにアプリケーションを再度構築する必要があります。動的リンクの場合、このステップは不要です。

アプリケーションの依存関係バージョンをアップグレードし、アプリケーションの機能を確認したら、フリートへのアプリケーションのデプロイに進みます。アプリケーションのデプロイが完了したら、V2 の暗号化· 復号クライアントへの移行フェーズに進むことができます。

暗号化および復号クライアントを V2 に移行する

以下の手順は、Amazon S3 の暗号化クライアントのコードを V1 から V2 へ正常に移行する方法を示しています。コードの変更が必要なため、静的または動的にリンクしているかどうかにかかわらず、アプリケーションを再構築する必要があります AWS SDK for C++。

新しい暗号化マテリアルの使用

V2 Amazon S3 暗号化クライアントと V2 暗号設定で、次の暗号化マテリアルは廃止されました。

- `SimpleEncryptionMaterials`
- `KMSEncryptionMaterials`

代わりに、次の安全な暗号マテリアルが使用されます。

- `SimpleEncryptionMaterialsWithGCMAAD`
- `KMSWithContextEncryptionMaterials`

V2 S3 暗号化クライアントをコンストラクトするには、次のコード変更が必要です。

- S3 暗号化クライアント作成時に **`KMSEncryptionMaterials`** を使用している場合:
  - V2 S3 暗号化クライアントを作成するときは、`KMSEncryptionMaterials` を `KMSWithContextEncryptionMaterials` に置き換え、それを V2 暗号設定で指定します。
  - V2 Amazon S3 暗号化クライアントでオブジェクトを配置するときは、CEK の暗号化に使用する KMS コンテキストとして、文字列–文字列のコンテキストマップを明示的に指定する必要があります。このマップは空でも構いません。
- S3 暗号化クライアント作成時に **`SimpleEncryptionMaterials`** を使用している場合:
  - V2 Amazon S3 暗号化クライアントを作成するときは、`SimpleEncryptionMaterials` を `SimpleEncryptionMaterialsWithGCMAAD` に置き換え、それを V2 暗号化設定で指定します。
  - V2 Amazon S3 暗号化クライアントでオブジェクトを配置するときは、空の文字列–文字列コンテキストマップを明示的に指定する必要があります。指定しない場合、SDK はエラーを返します。

例: KMS/KMSWithContext キーラップアルゴリズムの使用

移行前 (KMS キーラップ)

```
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);
```

移行後 (KMSWithContext キーラップ)

```
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
 */);
```

例: AES/AES-GCM キーラップアルゴリズムの使用

移行前 (AES キーラップ)

```
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);
```

移行後 (AES-GCM キーラップ)

```
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 */);
```

その他の例

次の例は、V1 から V2 への移行に関連する特定のユースケースに対処する方法を示しています。

レガシー Amazon S3 暗号化クライアントによって暗号化されたオブジェクトの復号

デフォルトでは、V2 Amazon S3 暗号化クライアントを使用して、非推奨のキーラップアルゴリズムまたは非推奨のコンテンツ暗号化スキーマで暗号化されたオブジェクトを復号することはできません。

次のキーラップアルゴリズムは廃止されました。

- KMS
- AES_KEY_WRAP

また、次のコンテンツ暗号化スキーマも廃止されました。

- CBC
- CTR

でレガシー Amazon S3 AWS SDK for C++ 暗号化クライアントを使用してオブジェクトを暗号化している場合は、次の場合に廃止されたメソッドを使用する可能性があります。

- SimpleEncryptionMaterials または KMSEncryptionMaterials を使用していた。
- 暗号化設定で Crypto Mode として ENCRYPTION_ONLY を使用していた。

廃止されたキーラップアルゴリズムや暗号化スキーマで暗号化されたオブジェクトを V2 Amazon S3 暗号化クライアントで復号するには、V2 暗号設定で SecurityProfile のデフォルト値を V2 から V2_AND_LEGACY にオーバーライドする必要があります。

例

移行前

```
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);
```

移行後

```
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);
```

バイト範囲を指定してオブジェクトを復号する

レガシー Amazon S3 暗号化クライアントでは、S3 オブジェクトを復号するときにバイト範囲を指定できます。V2 の Amazon S3 暗号化クライアントでは、この機能はデフォルトで DISABLED になっています。したがって、V2 暗号化設定で RangeGetMode のデフォルト値を DISABLED から ALL にオーバーライドする必要があります。

例

移行前

```
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);
```

移行後

```
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);
```

任意の CMK でオブジェクトを復号する

KMSWithContextEncryptionMaterials で暗号化されたオブジェクトを復号するとき、V2 Amazon S3 暗号化クライアントは空のマスターキーを提供することで、KMS が適切な CMK を特定できるようになります。この機能はデフォルトで DISABLED になっています。KMS 暗号化マテリアルに対して SetKMSDecryptWithAnyCMK(true) を呼び出して、この機能を明示的に設定する必要があります。

例

移行前

```
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);
```

移行後

```
auto materials = Aws::MakeShared<KMSWithContextEncryptionMaterials>("s3EncryptionV2",
 ""/* provide an empty KMS Master Key*/);
materials.SetKMSDecryptWithAnyCMK(true);
CryptoConfigurationV2 cryptoConfig(materials);
S3EncryptionClientV2 encryptionClient(cryptoConfig);
// Code snippet here to setup the getObjectRequest object.
encryptionClient.GetObject(getObjectRequest);
```

これらすべての移行シナリオに対応する完全なコードについては、GitHub の「Amazon S3 Encryption example」を参照してください。

Amazon S3 暗号化クライアントの移行 (V2 から V3)

> **Note**
>
> Amazon S3 暗号化クライアントの V1 を使用している場合は、まず V2 に移行してから V3 に移行する必要があります。「Amazon S3 暗号化クライアントの移行 (V1 から V2)」を参照してください。

このトピックでは、Amazon Simple Storage Service (Amazon S3) 暗号化クライアントのバージョン 2 (V2) からバージョン 3 (V3) にアプリケーションを移行し、移行プロセス全体でアプリケーションの可用性を確保する方法について説明します。V3 では、命令ファイルのデータキーの改ざんから保護することでセキュリティを強化するためのALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEYアルゴリズムとコミットメントポリシーが導入されています。

移行の概要

この移行は 2 つのフェーズから構成されます。

1. 新しいフォーマットを読み取るために既存のクライアントを更新します。まず、更新されたバージョンの AWS SDK for C++ をアプリケーションにデプロイします。これにより、既存の V2 暗号化クライアントは、新しい V3 クライアントによって書き込まれたオブジェクトを復号できます。アプリケーションで AWS SDKs、各 SDK を個別にアップグレードする必要があります。

2. 暗号化クライアントと復号クライアントを V3 に移行します。すべての V2 暗号化クライアントが新しい形式を読み取れるようになったら、既存の暗号化クライアントと復号クライアントをそれぞれの V3 バージョンに移行できます。

V3 の概念について

Amazon S3 暗号化クライアントのバージョン 3 では、データキーの改ざんに対する保護を強化する新しいセキュリティ機能が導入されています。これらの概念を理解することは、移行を成功させるために不可欠です。

コミットメントポリシー

コミットメントポリシーは、暗号化および復号オペレーション中に暗号化クライアントがキーコミットメントを処理する方法を制御します。V3 には、さまざまな移行シナリオとセキュリティ要件をサポートする 3 つのポリシーオプションが用意されています。

FORBID_ENCRYPT_ALLOW_DECRYPT

暗号化動作: V2 と同じアルゴリズムを使用して、キーコミットメントなしでオブジェクトを暗号化します。

復号動作: キーコミットメントの有無にかかわらず暗号化されたオブジェクトの復号を許可します。

セキュリティへの影響: このポリシーはキーコミットメントを強制せず、命令ファイル内の暗号化されたデータキーの改ざんを許可する場合があります。このポリシーは、V2 クライアントが新しく暗号化されたオブジェクトを読み取る必要がある最初の移行フェーズでのみ使用します。

バージョンの互換性: このポリシーで暗号化されたオブジェクトは、すべての V2 および V3 実装で読み取ることができます。

REQUIRE_ENCRYPT_ALLOW_DECRYPT (デフォルト)

暗号化動作: ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEYアルゴリズムを使用して、キーコミットメントでオブジェクトを暗号化します。

復号動作: キーコミットメントで暗号化されたオブジェクトと、キーコミットメントなしで暗号化されたオブジェクトの両方の復号を許可します。

セキュリティへの影響: このポリシーは、古いオブジェクトを読み取るための下位互換性を維持しながら、新しく暗号化されたオブジェクトに対して強力なセキュリティを提供します。これは、ほとんどの移行シナリオで推奨されるポリシーです。

バージョンの互換性: このポリシーで暗号化されたオブジェクトは、V3 および最新の V2 実装でのみ読み取ることができます。V2 クライアントはこれらのオブジェクトを復号できません。ただし、このポリシーを使用する V3 クライアントは、引き続き V2 クライアントによって暗号化されたオブジェクトを復号できます。

REQUIRE_ENCRYPT_REQUIRE_DECRYPT

暗号化動作: ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEYアルゴリズムを使用して、キーコミットメントでオブジェクトを暗号化します。

復号動作: キーコミットメントで暗号化されたオブジェクトの復号のみを許可します。キーコミットメントなしで暗号化されたオブジェクトを拒否します。

セキュリティへの影響: このポリシーは、すべてのオペレーションに主要なコミットメントを適用することで、最高レベルのセキュリティを提供します。このポリシーは、すべてのオブジェクトがキーコミットメントで再暗号化され、レガシー V1 または V2 で暗号化されたオブジェクトを読み取る必要がなくなった後にのみ使用します。

バージョンの互換性: このポリシーで暗号化されたオブジェクトは、V3 および最新の V2 実装でのみ読み取ることができます。さらに、このポリシーを使用するクライアントは、V1 または V2 クライアントによって暗号化されたオブジェクトを復号できません。

移行に関する考慮事項: 移行中に V2 クライアントFORBID_ENCRYPT_ALLOW_DECRYPTが新しいオブジェクトを読み取る必要がある場合は、 から開始し、すべてのクライアントが V3 にアップグレードREQUIRE_ENCRYPT_ALLOW_DECRYPTされたら に移動します。最後に、すべてのレガシーオブジェクトが再暗号化された後にREQUIRE_ENCRYPT_REQUIRE_DECRYPTのみ検討してください。

ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY アルゴリズム

ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY アルゴリズムは、命令ファイルに保存されている暗号化されたデータキーのセキュリティを強化する V3 で導入された新しい暗号化アルゴリズムです。

命令ファイルの影響: ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEYアルゴリズムは、暗号化されたデータキーを含む暗号化メタデータを保存する個別の S3 オブジェクトである命令ファイルにのみ影響します。暗号化メタデータをオブジェクトメタデータに保存するオブジェクト (デフォルトのストレージ方法) は、このアルゴリズムの変更の影響を受けません。

改ざんに対する保護: ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEYアルゴリズムは、暗号化されたデータキーを暗号化コンテキストに暗号化バインドすることで、データキーの改ざんから保護します。これにより、攻撃者が命令ファイル内の別の暗号化されたデータキーを置き換えるのを防ぎ、意図しないキーによる復号につながる可能性があります。

バージョンの互換性: ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEYアルゴリズムで暗号化されたオブジェクトは、V3 実装と V3 復号サポートを含む SDK の最新の V3 V2 移行バージョンでのみ復号できます。

> **Warning**
>
> 重要: (REQUIRE_ENCRYPT_ALLOW_DECRYPTまたはREQUIRE_ENCRYPT_REQUIRE_DECRYPTコミットメントポリシーを使用して) ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEYアルゴリズムによる暗号化を有効にする前に、これらのオブジェクトを読み取るすべてのクライアントが V3 または V3 復号をサポートするV3 V2 移行バージョンにアップグレードされていることを確認する必要があります。すべてのリーダーを最初にアップグレードしないと、新しく暗号化されたオブジェクトの復号に失敗します。

新しいフォーマットを読み取るために既存のクライアントを更新する

まず、既存のクライアントを最新の SDK リリースに更新する必要があります。このステップを完了すると、アプリケーションの V2 クライアントは、アプリケーションのコードベースを更新することなく、V3 暗号化クライアントによって暗号化されたオブジェクトを復号できます。

の最新バージョンをビルドしてインストールする AWS SDK for C++

ソースから SDK を利用するアプリケーション

ソース AWS SDK for C++ から を構築してインストールする場合は、GitHub の から SDK ソースをダウンロードまたはクローンaws/aws-sdk-cppします。その後、通常のビルドおよびインストール手順を繰り返します。

1.11.x より前のバージョン AWS SDK for C++ からアップグレードする場合は、各メジャーバージョンで導入された重大な変更について、この CHANGELOG を参照してください。をビルドおよびインストールする方法の詳細については AWS SDK for C++、「」を参照してください ソースコードからの AWS SDK for C++ の入手。

Vcpkg から SDK を利用するアプリケーション

アプリケーションで Vcpkg を使用して SDK の更新を追跡している場合は、既存の Vcpkg アップグレード方法を使用して SDK を最新バージョンにアップグレードするだけです。バージョンがリリースされてからパッケージマネージャーで使用可能になるまでに遅延があることに注意してください。最新バージョンはソースからインストールすることで常に使用できます。

次のコマンドを実行してパッケージ aws-sdk-cpp をアップグレードできます。

```
vcpkg upgrade aws-sdk-cpp
```

次に、パッケージ aws-sdk-cpp のバージョンを確認します。

```
vcpkg list aws-sdk-cpp
```

V3-encryptedオブジェクトの復号をサポートするには、バージョンが 1.11.x 以上である必要があります。

での Vcpkg の使用の詳細については AWS SDK for C++、「」を参照してください パッケージマネージャーからの AWS SDK for C++ の取得。

アプリケーションのビルド、インストール、デプロイする

アプリケーションが と静的にリンクされている場合 AWS SDK for C++、アプリケーションにコードの変更は必要ありませんが、最新の SDK の変更を使用するためにアプリケーションを再度構築する必要があります。動的リンクの場合、このステップは不要です。

アプリケーションの依存関係バージョンをアップグレードし、アプリケーションの機能を確認したら、フリートへのアプリケーションのデプロイに進みます。アプリケーションのデプロイが完了したら、次のフェーズに進み、アプリケーションを移行して V3 暗号化クライアントと復号クライアントを使用できます。

暗号化クライアントと復号クライアントを V3 に移行する

次の手順では、Amazon S3 暗号化クライアントの V2 から V3 にコードを正常に移行する方法を示します。 Amazon S3 コードの変更が必要なため、静的または動的にリンクしているかどうかにかかわらず、アプリケーションを再構築する必要があります AWS SDK for C++。

V3 暗号化クライアントの使用

V3 は `S3EncryptionClientV3` クラス と を導入`CryptoConfigurationV3`し、V2 に相当するものを置き換えます。V3 の主な違いは次のとおりです。

- V3 は `KMSWithContextEncryptionMaterials` (V2 と同じ) を使用しますが、 で明示的な設定が必要です`CryptoConfigurationV3`。
- すべての`PutObject`オペレーションには暗号化コンテキストマップが必要です (空にすることもできます) 。
- V3 では、暗号化と復号の動作を制御するコミットメントポリシーが導入されています。
- デフォルトでは、V3 は `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY`アルゴリズムを使用してキーコミットメントで暗号化します。
- レガシーアルゴリズム復号設定 API は から `config.SetSecurityProfile(SecurityProfile::V2_AND_LEGACY);`に変更されます`config.AllowLegacy();`。

例: KMS 暗号化を使用した V2 から V3 への移行

移行前 (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);
```

移行中 (下位互換性がある 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);
```

移行後 (キーコミットメント付き 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);
```

その他の例

このセクションでは、さまざまな移行シナリオと要件をサポートするために V3 暗号化クライアントオプションを設定するための追加の例を示します。

レガシーサポートの有効化

V3 クライアントは、 `REQUIRE_ENCRYPT_ALLOW_DECRYPT`または `FORBID_ENCRYPT_ALLOW_DECRYPT`コミットメントポリシーを使用する場合にのみ、V2 クライアントによって暗号化されたオブジェクトを復号できます。ただし、V1 クライアントによって暗号化されたオブジェクトを復号する必要がある場合は、 `AllowLegacy()`メソッドを使用してレガシーサポートを明示的に有効にする必要があります。

レガシーサポートを使用するタイミング:

- S3 暗号化クライアントの V1 を使用して暗号化されたオブジェクトが S3 にあります。
- これらの V1-encryptedオブジェクトは、移行プロセス中に V3 クライアントで読み取る必要があります。
- `REQUIRE_ENCRYPT_ALLOW_DECRYPT` または `FORBID_ENCRYPT_ALLOW_DECRYPT`コミットメントポリシーを使用している。

> **Warning**
>
> レガシーサポートは、移行中に一時的にのみ有効にする必要があります。すべての V1 オブジェクトが V2 または V3 で再暗号化されたら、レガシーサポートを無効にしてセキュリティを最大化します。

例: レガシーサポートの有効化

```
// 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);
```

ストレージ方法の設定

S3 暗号化クライアントは、暗号化メタデータをオブジェクトメタデータ (デフォルト) として、または別の命令ファイルに 2 つの方法で保存できます。ストレージメソッドは、 の SetStorageMethod()メソッドを使用して設定できますCryptoConfigurationV3。

ストレージメソッドのオプション:

METADATA (デフォルト)

暗号化メタデータは、オブジェクトのメタデータヘッダーに保存されます。これは、すべての暗号化情報がオブジェクト自体に保存されるため、最も一般的で便利な方法です。

使用するタイミング: ほとんどのシナリオでこの方法を使用します。暗号化メタデータはオブジェクトとともに移動するため、オブジェクト管理が簡素化されます。

INSTRUCTION_FILE

暗号化メタデータは、サフィックス が付いた別の S3 オブジェクト (命令ファイル) に保存されます.instruction。

使用時: オブジェクトメタデータのサイズが懸念される場合、または暗号化されたオブジェクトから暗号化メタデータを分離する必要がある場合は、この方法を使用します。命令ファイルを使用するには、2 つの S3 オブジェクト (暗号化されたオブジェクトとその命令ファイル) を 1 つではなく管理する必要があることに注意してください。

例: ストレージ方法の設定

```
// 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` ストレージメソッドを使用する場合、暗号化されたオブジェクトを削除しても、命令ファイルは自動的に削除されないことに注意してください。両方のオブジェクトを個別に管理する必要があります。

AWS SDK for C++ デベロッパーガイドのドキュメント履歴

このトピックでは、 AWS SDK for C++ デベロッパーガイドの重要な変更点を示します。このドキュメントの更新に関する通知については、RSS フィードを購読してください。

変更	説明	日付
クライアント設定オプション	クライアント設定変数のリストを更新しました。	2025 年 9 月 8 日
メモリ管理と目次の更新	メモリ管理パラメータを更新しました。 AWS SDKs。	2025 年 3 月 14 日
Amazon S3 暗号化クライアント V3 の移行	Amazon S3 暗号化クライアントの V2 から V3 への移行に関する情報を追加しました。 V3 Amazon S3	2024 年 12 月 2 日
カスタム libcrypto	カスタム libcrypto の使用に関する内容を追加しました。CMake の最大制限に関する記述を削除しました。使用可能な CMake のパラメータを更新しました。	2024 年 2 月 20 日
目次	目次を更新して、コード例をよりわかりやすくしました。	2023 年 6 月 1 日
IAM ベストプラクティスの更新	IAM ベストプラクティスに沿ってガイドを更新しました。詳細については、「IAM でのセキュリティのベストプラクティス」を参照してください。	2023 年 3 月 1 日
nuget の削除	使用可能な最新バージョンが古すぎるため、パッケージマネージャーの選択肢としての	2022 年 12 月 2 日

	nuget に関する記述を削除しました。	
「開始方法」の更新	vcpkg コンテンツを更新して、 が でサポートされておらず、 AWS が外部オプションであることを明確に伝えました。curl を使用して SDK for Windows をビルドする手順を更新しました。	2022 年 10 月 18 日
`ClientConfiguration` に更新	`ClientConfiguration` の構造を更新して、最新の API を正確に反映しました。	2022 年 9 月 22 日
「開始方法」の改善点	Windows と Linux における SDK のビルド手順の明確さを改善しました。	2022 年 6 月 24 日
Windows curl サポート	curl を使用して SDK for Windows をビルドする方法に関する注意事項を追加しました。	2022 年 6 月 15 日
EC2-Classic の廃止	EC2-Classic の廃止に関する注記を追加しました。	2022 年 4 月 13 日
SDK メトリクスの有効化	廃止されたSDK メトリクスの有効化についての情報を削除しました。	2022 年 1 月 20 日
AWS サービスの使用	コードサンプル リポジトリの GitHub で利用できるコード例のリストを含めました。	2022 年 1 月 11 日
改良点	「開始方法」セクション、「コード例」セクション、全体的な標準化を改善しました。	2021 年 6 月 9 日

バージョン更新	SDK の一般リリースバージョン 1.9 への更新を反映しました。	2021 年 4 月 20 日
の使用開始 AWS SDK for C++	このセクションは新しい構成と詳細を加えて更新されました。	2021 年 3 月 17 日
Amazon S3 暗号化クライアントの移行	Amazon S3 暗号化クライアントを V1 から V2 に移行する方法に関する情報を追加しました。	2020 年 8 月 7 日
セキュリティコンテンツ	セキュリティコンテンツを追加しました。	2020 年 2 月 6 日
バケットの作成、一覧表示、削除	サポートするように Amazon S3 `CreateBucket` の例を更新しました AWS リージョン。	2019 年 6 月 20 日
ビルド手順	SDK のビルド手順を更新しました。	2019 年 4 月 16 日
非同期メソッド	新しいセクションを追加。	2019 年 4 月 16 日
サービスクライアントクラス	各種更新。	2019 年 4 月 5 日
Amazon S3 アクセス許可の管理	各種更新。	2019 年 4 月 3 日
ビルドと設定変数の更新	SDK のビルド手順を更新しました。使用可能な AWS クライアント設定変数を更新しました。	2019 年 3 月 1 日
vcpkg C++ パッケージマネージャー	vcpkg C++ パッケージマネージャーの設定手順を更新しました。	2019 年 1 月 19 日

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。