Entwicklerhandbuch

AWS SDK für C++

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

AWS SDK für C++: Entwicklerhandbuch

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

Die Marken und Handelsmarken von Amazon dürfen nicht in einer Weise in Verbindung mit nicht von Amazon stammenden Produkten oder Services verwendet werden, die geeignet ist, Kunden irrezuführen oder Amazon in irgendeiner Weise herabzusetzen oder zu diskreditieren. Alle anderen Marken, die nicht im Besitz von Amazon sind, gehören den jeweiligen Besitzern, die möglicherweise mit Amazon verbunden sind oder von Amazon gesponsert werden.

Table of Contents

Was ist der AWS SDK für C++ Developer Guide? ... 1
Zusätzliche Dokumentation und Ressourcen ... 1
Wartung und Support für SDK-Hauptversionen ... 2
Erste Schritte ... 3
Authentifizierung mit AWS ... 3
Verwenden von Anmeldeinformationen für die Konsole ... 3
Verwenden von IAM Identity Center ... 4
Weitere Informationen zur Authentifizierung ... 6
Das SDK aus dem Quellcode abrufen ... 6
Auf Windows bauen ... 7
Bauen auf Linux/macOS ... 12
Eine einfache Anwendung erstellen ... 16
Das SDK von einem Paketmanager abrufen ... 22
Voraussetzungen ... 8
Holen Sie sich das SDK mit vcpkg ... 23
Behebung von Build-Problemen ... 24
CMake Fehler: Es konnte keine von AWSSDK "bereitgestellte Paketkonfigurationsdatei gefunden werden ... 25
CMake Fehler: Ladedatei konnte nicht gefunden werden (und Sie verwenden SDK-Version 1.8) ... 25
CMake Fehler: Ladedatei konnte nicht gefunden werden ... 26
Laufzeitfehler: Der Vorgang kann nicht fortgesetzt werden, da `aws-*.dll` er nicht gefunden wurde ... 27
Konfiguration von Service-Clients ... 29
SDK-Konfiguration ... 30
Client-Konfiguration extern ... 31
Client-Konfiguration im Code ... 33
Deklarationen ... 34
Beschreibungen ... 36
AWS-Region ... 42
Anbieter von Anmeldeinformationen ... 43
Die Kette der Anbieter von Anmeldeinformationen ... 43
Anbieter expliziter Anmeldeinformationen ... 47
Zwischenspeichern von Identitäten ... 47

CMake Parameter ... 47
Allgemeine CMake Variablen und Optionen ... 48
CMake Android-Variablen und -Optionen ... 63
Protokollierung ... 66
HTTP ... 70
Steuerung der von und der `HttpClient` verwendeten Iostreams `AWSClient` ... 70
Verwenden eines benutzerdefinierten Libcrypto ... 72
Wie man ein benutzerdefiniertes Libcrypto in das SDK for C++ einbaut ... 72
Alles in einem Docker-Image zusammenführen ... 74
Verwenden der SDK ... 76
Initialisieren und Herunterfahren des SDK ... 76
AWS-Service Anfragen stellen ... 77
Asynchrone Programmierung ... 78
Asynchrone SDK-Methoden ... 78
Asynchrone SDK-Methoden werden aufgerufen ... 78
Benachrichtigung über den Abschluss eines asynchronen Vorgangs ... 80
Utility-Module ... 83
HTTP-Stack ... 83
Zeichenketten-Utils ... 83
Werkzeuge zum Hashing ... 83
JSON-Parser ... 84
XML-Parser ... 84
Speicherverwaltung ... 84
Speicher zuweisen und Zuweisung aufheben ... 85
STL und AWS Zeichenketten und Vektoren ... 86
Verbleibende Probleme ... 87
Native SDK-Entwickler und Speichersteuerungen ... 88
Fehlerbehandlung ... 88
Wird angerufen AWS-Services ... 91
Erste Schritte mit Codebeispielen ... 91
Struktur der Codebeispiele ... 91
Codebeispiele in Visual Studio erstellen und debuggen ... 93
Erste Schritte zur Behebung von Laufzeitfehlern ... 95
Geführte Beispiele ... 98
CloudWatch Amazon-Beispiele ... 99
Beispiele für Amazon DynamoDB ... 116

Beispiele für Amazon EC2 ... 132
Amazon S3 S3-Beispiele ... 157
Beispiele für Amazon SQS ... 192
Codebeispiele ... 209
ACM ... 210
Aktionen ... 210
API Gateway ... 240
Szenarien ... 240
Aurora ... 241
Erste Schritte ... 241
Grundlagen ... 244
Aktionen ... 210
Szenarien ... 240
Auto Scaling ... 285
Erste Schritte ... 241
Grundlagen ... 244
Aktionen ... 210
CloudTrail ... 316
Aktionen ... 210
CloudWatch ... 321
Aktionen ... 210
CloudWatch Protokolle ... 332
Aktionen ... 210
CodeBuild ... 336
Aktionen ... 210
Amazon Cognito Identity Provider ... 341
Erste Schritte ... 241
Aktionen ... 210
Szenarien ... 240
DynamoDB ... 365
Erste Schritte ... 241
Grundlagen ... 244
Aktionen ... 210
Szenarien ... 240
Amazon EC2 ... 443
Erste Schritte ... 241

Aktionen 210
EventBridge 477
Aktionen 210
AWS Glue 481
Erste Schritte 241
Grundlagen 244
Aktionen 210
HealthImaging 520
Erste Schritte 241
Aktionen 210
Szenarien 240
IAM 555
Erste Schritte 241
Grundlagen 244
Aktionen 210
AWS IoT 596
Erste Schritte 241
Grundlagen 244
Aktionen 210
AWS IoT data 638
Aktionen 210
Lambda 641
Erste Schritte 241
Grundlagen 244
Aktionen 210
Szenarien 240
MediaConvert 666
Aktionen 210
Amazon RDS 674
Erste Schritte 241
Grundlagen 244
Aktionen 210
Szenarien 240
Amazon RDS Data Service 711
Szenarien 240
Amazon Rekognition 712

Erste Schritte ... 241
Aktionen ... 210
Szenarien ... 240
Amazon S3 ... 718
Erste Schritte ... 241
Grundlagen ... 244
Aktionen ... 210
Szenarien ... 240
Secrets Manager ... 800
Aktionen ... 210
Amazon SES ... 802
Aktionen ... 210
Szenarien ... 240
Amazon SNS ... 824
Erste Schritte ... 241
Aktionen ... 210
Szenarien ... 240
Amazon SQS ... 866
Erste Schritte ... 241
Aktionen ... 210
Szenarien ... 240
AWS STS ... 902
Aktionen ... 210
Amazon Transcribe Streaming ... 904
Aktionen ... 210
Szenarien ... 240
Sicherheit ... 913
Datenschutz ... 914
Identitäts- und Zugriffsverwaltung ... 915
Zielgruppe ... 915
Authentifizierung mit Identitäten ... 916
Verwalten des Zugriffs mit Richtlinien ... 917
Wie AWS-Services arbeiten Sie mit IAM ... 919
Problembehebung bei AWS Identität und Zugriff ... 920
Compliance-Validierung ... 922
Ausfallsicherheit ... 922

Sicherheit der Infrastruktur 923
Erzwingen einer Mindest-TLS-Version 924
Erzwingen Sie eine bestimmte TLS-Version mit libcurl auf allen Plattformen 924
Erzwingen Sie eine bestimmte TLS-Version unter Windows 925
Migration des Amazon S3 S3-Verschlüsselungsclients (V1 auf V2) 929
Überblick über die Migration 929
Aktualisieren Sie bestehende Clients, um neue Formate zu lesen 929
Migrieren Sie die Verschlüsselungs- und Entschlüsselungsclients zu V2 931
Weitere Beispiele 933
Migration des Amazon S3 S3-Verschlüsselungsclients (V2 auf V3) 936
Überblick über die Migration 936
V3-Konzepte verstehen 937
Aktualisieren Sie bestehende Clients, um neue Formate zu lesen 939
Migrieren Sie die Verschlüsselungs- und Entschlüsselungsclients zu V3 941
Weitere Beispiele 944
Dokumentverlauf 948
..... cmlii

Was ist der AWS SDK für C++ Developer Guide?

Willkommen im AWS SDK für C++ Developer Guide.

Das AWS SDK für C++ bietet eine moderne C++-Schnittstelle (Version C++ 11 oder höher) für Amazon Web Services (AWS). Es bietet sowohl High-Level- als auch AWS Low-Level-Funktionen APIs für fast alle Funktionen, minimiert Abhängigkeiten und bietet Plattformportabilität unter Windows, macOS, Linux und Mobilgeräten.

Erste Schritte mit dem AWS SDK für C++

> **Note**
>
> Das AWS IoT SDKs und das `aws-iot-device-sdk-cpp` sind von diesem SDK getrennt. Das AWS IoT Geräte-SDK SDK for C++ v2 ist `aws-iot-device-sdk-cpp-v2`unter verfügbar GitHub.
> Weitere Informationen zu AWS IoT finden Sie unter Was ist AWS IoT im AWS IoT Entwicklerhandbuch enthalten.

Zusätzliche Dokumentation und Ressourcen

Zusätzlich zu diesem Handbuch stehen AWS SDK für C++ -Entwicklern folgende wertvolle Online-Ressourcen zur Verfügung:

- AWS SDKs Referenzhandbuch für Tools und Tools: Enthält Einstellungen, Funktionen und andere grundlegende Konzepte, die unter den AWS SDKs Benutzern üblich sind.
- GitHub:
  - SDK-Quellcode
  - SDK-Probleme
- AWS SDK für C++ API Reference
- AWS Blog für C++-Entwickler
- Die AWS-Codebeispiel-Katalog
- SDK-Lizenz
- Video: Vorstellung der AWS SDK für C++ von AWS re:invent 2015

Wartung und Support für SDK-Hauptversionen

Informationen zur Wartung und zum Support für SDK-Hauptversionen und die ihnen zugrunde liegenden Abhängigkeiten finden Sie im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch:

- AWS SDKs Richtlinien zur Wartung von Tools
- AWS SDKs und Matrix zur Unterstützung der Tools-Versionen

Erste Schritte mit dem AWS SDK für C++

AWS SDK für C++ ist eine modularisierte, plattformübergreifende Open-Source-Bibliothek, mit der Sie eine Verbindung zu Amazon Web Services herstellen können.

Sie AWS SDK für C++ unterstützt mehrere Plattformen über mehrere Domänen hinweg, darunter Videospiele, Systeme, mobile Geräte und eingebettete Geräte. CMake CMake ist ein Build-Tool, mit dem Sie die Abhängigkeiten Ihrer Anwendung verwalten und Makefiles erstellen können, die für die Plattform geeignet sind, auf der Sie aufbauen. CMake entfernt die Teile des Builds, die nicht für Ihre Plattform oder Anwendung verwendet werden.

Bevor Sie Code für den Zugriff auf AWS Ressourcen ausführen, müssen Sie festlegen, mit AWS welcher Methode sich Ihr Code authentifiziert.

- Authentifizierung mit AWS AWS SDK for C++

Um den AWS SDK für C++ in Ihrem Code zu verwenden, rufen Sie die ausführbaren SDK-Dateien ab, indem Sie die SDK-Quelle direkt oder mithilfe eines Paketmanagers erstellen.

- Den AWS SDK für C++ aus dem Quellcode holen
- AWS SDK für C++ Von einem Paketmanager bekommen

Falls Sie Probleme mit dem Build haben, finden Sie weitere Informationen CMake unter. Behebung von Problemen mit AWS SDK for C++ C++-Build

Authentifizierung mit AWS AWS SDK for C++

Sie müssen festlegen, wie sich Ihr Code AWS bei der Entwicklung mit authentifiziert. AWS-Services Sie können den programmatischen Zugriff auf AWS Ressourcen je nach Umgebung und verfügbarem AWS Zugriff auf unterschiedliche Weise konfigurieren. Auswahlmöglichkeiten zu allen primären Authentifizierungsmethoden sowie Anleitungen zur Konfiguration dieser Methode für das SDK finden Sie unter Authentifizierung und Zugriff im AWS SDKs Referenzhandbuch zu Tools.

Verwenden von Anmeldeinformationen für die Konsole

Für die lokale Entwicklung empfehlen wir neuen Benutzern, ihre vorhandenen Anmeldedaten für die AWS Management Console für den programmgesteuerten Zugriff auf Dienste zu AWS

verwenden. AWS Generiert nach einem browserbasierten Authentifizierungsablauf temporäre Anmeldeinformationen, die in allen lokalen Entwicklungstools wie der AWS CLI AWS -Tools für PowerShell und AWS SDKs funktionieren. Diese Funktion vereinfacht die Konfiguration und Verwaltung von AWS CLI-Anmeldeinformationen, insbesondere wenn Sie die interaktive Authentifizierung der Verwaltung von langfristigen Zugriffsschlüsseln vorziehen.

Wenn Sie diese Methode wählen, folgen Sie den Anweisungen, um sich mit Konsolenanmeldedaten über die AWS CLI anzumelden. Weitere Informationen finden Sie unter Anmeldung für die AWS lokale Entwicklung mit Konsolenanmeldedaten.

Nach der Einrichtung mit AWS CLI verwendet die standardmäßige Anbieterkette für Anmeldeinformationen automatisch das von der AWS CLI zwischengespeicherte Anmeldetoken, um Anfragen zu stellen.

Verwenden von IAM Identity Center

Diese Methode umfasst die Installation von, AWS CLI um die Konfiguration zu vereinfachen und sich regelmäßig beim AWS Zugriffsportal anzumelden.

Wenn Sie sich für diese Methode entscheiden, führen Sie das Verfahren für die IAM Identity Center-Authentifizierung im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch durch. Danach sollte Ihre Umgebung die folgenden Elemente enthalten:

- Die AWS CLI, mit der Sie eine AWS Access-Portal-Sitzung starten, bevor Sie Ihre Anwendung ausführen.
- Eine gemeinsam genutzte AWS`config` Datei mit einem `[default]` Profil mit einer Reihe von Konfigurationswerten, auf die vom SDK aus verwiesen werden kann. Informationen zum Speicherort dieser Datei finden Sie unter Speicherort der gemeinsam genutzten Dateien im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch.
- Die gemeinsam genutzte `config` Datei legt die `region`Einstellung fest. Dies legt die Standardeinstellung AWS-Region fest, die das SDK für AWS Anfragen verwendet. Diese Region wird für SDK-Dienstanforderungen verwendet, für die keine zu verwendende Region angegeben ist.
- Das SDK verwendet die Konfiguration des SSO-Token-Anbieters des Profils, um Anmeldeinformationen abzurufen, bevor Anfragen an gesendet AWS werden. Der `sso_role_name` Wert, bei dem es sich um eine IAM-Rolle handelt, die mit einem IAM Identity Center-Berechtigungssatz verbunden ist, sollte den Zugriff auf die in Ihrer AWS-Services Anwendung verwendeten Rollen ermöglichen.

Die folgende `config` Beispieldatei zeigt ein Standardprofil, das mit der Konfiguration des SSO-Token-Anbieters eingerichtet wurde. Die `sso_session` Einstellung des Profils bezieht sich auf den genannten `sso-session`Abschnitt. Der `sso-session` Abschnitt enthält Einstellungen zum Initiieren einer AWS Access-Portal-Sitzung.

```
[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
```

Für die Verwendung der IAM Identity Center-Authentifizierung müssen Ihrer Anwendung AWS SDK für C++ keine zusätzlichen Pakete (wie `SSO` und`SSOOIDC`) hinzugefügt werden.

Starten Sie eine AWS Access-Portal-Sitzung

Bevor Sie eine Zugriffsanwendung ausführen AWS-Services, benötigen Sie eine aktive AWS Access-Portal-Sitzung, damit das SDK die IAM Identity Center-Authentifizierung zur Auflösung von Anmeldeinformationen verwenden kann. Abhängig von Ihrer konfigurierten Sitzungsdauer läuft Ihr Zugriff irgendwann ab und das SDK wird auf einen Authentifizierungsfehler stoßen. Um sich beim AWS Zugriffsportal anzumelden, führen Sie den folgenden Befehl in der aus AWS CLI.

```
aws sso login
```

Da Sie ein Standardprofil eingerichtet haben, müssen Sie den Befehl nicht mit einer `--profile`-Option aufrufen. Wenn die Konfiguration Ihres SSO-Token-Anbieters ein benanntes Profil verwendet, lautet der Befehl `aws sso login --profile named-profile`.

Führen Sie den folgenden AWS CLI Befehl aus, um zu testen, ob Sie bereits eine aktive Sitzung haben.

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

In der Antwort auf diesen Befehl sollten das in der freigegebenen `config`-Datei konfigurierte IAM-Identity-Center-Konto und der Berechtigungssatz angegeben werden.

> **Note**
>
> Wenn Sie bereits über eine aktive AWS Access-Portal-Sitzung verfügen und diese ausführen`aws sso login`, müssen Sie keine Anmeldeinformationen angeben. Beim Anmeldevorgang werden Sie möglicherweise aufgefordert, den AWS CLI Zugriff auf Ihre Daten zu gewähren. Da AWS CLI das auf dem SDK für Python aufbaut, können Berechtigungsnachrichten Variationen des `botocore` Namens enthalten.

Weitere Informationen zur Authentifizierung

Menschliche Benutzer, auch bekannt als menschliche Identitäten, sind die Personen, Administratoren, Entwickler, Betreiber und Verbraucher Ihrer Anwendungen. Sie benötigen eine Identität, um auf Ihre AWS Umgebungen und Anwendungen zugreifen zu können. Menschliche Benutzer, die Mitglieder Ihres Unternehmens sind, werden auch als Mitarbeiteridentitäten bezeichnet, d. h. Sie, der Entwickler. Verwenden Sie beim Zugriff AWS temporäre Anmeldeinformationen. Sie können einen Identitätsanbieter für Ihre menschlichen Benutzer verwenden, um Verbundzugriff auf AWS Konten zu ermöglichen, indem Sie Rollen übernehmen, die temporäre Anmeldeinformationen bereitstellen. Für eine zentralisierte Zugriffsverwaltung empfehlen wir Ihnen, AWS IAM Identity Center (IAM Identity Center) zu verwenden, um den Zugriff auf Ihre Konten und die Berechtigungen innerhalb dieser Konten zu verwalten. Weitere Alternativen finden Sie im Folgenden:

- Weitere Informationen zu bewährten Methoden finden Sie unter Bewährte Methoden für die Sicherheit in IAM im IAM-Benutzerhandbuch.
- Informationen zum Erstellen kurzfristiger AWS Anmeldeinformationen finden Sie unter Temporäre Sicherheitsanmeldeinformationen im IAM-Benutzerhandbuch.
- Weitere Informationen zu anderen Anbietern von AWS SDK für C++ Anmeldeinformationen finden Sie unter Standardisierte Anmeldeinformationsanbieter im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch.

Den AWS SDK für C++ aus dem Quellcode holen

Sie können das AWS SDK für C++ aus Ihrem Code heraus verwenden, indem Sie das SDK zuerst aus dem Quellcode erstellen und es dann lokal installieren.

Prozessübersicht

Allgemeiner Prozess	Detaillierter Prozess
Erstellen und installieren Sie die SDK-Quelle 1. Wird verwendet CMake , um Build-Dateien für das SDK zu generieren. 2. Erstellen Sie das SDK. 3. Installieren Sie das SDK.	Erstellen Sie zuerst das SDK aus dem Quellcode und installieren Sie es. • Auf Windows bauen • Bauen auf Linux/macOS
Erstellen Sie Ihre Anwendung mit dem SDK 1. Schreiben Sie Ihren eigenen Code, um das SDK zu verwenden, oder verwenden Sie eine Beispielanwendung und fügen Sie das `AWSSDK` Paket zu Ihrer Cmake-Datei hinzu. 2. Verwenden Sie CMake es, um Build-Dat eien für Ihre Anwendung zu generieren. 3. Erstellen Sie Ihre Anwendung. 4. Führen Sie Ihre Anwendung aus.	Entwickeln Sie dann mithilfe des SDK Ihre eigene Anwendung. • Eine einfache Anwendung erstellen

Das unter AWS SDK für C++ Windows erstellen

Um das einzurichten AWS SDK für C++, können Sie das SDK entweder selbst direkt aus der Quelle erstellen oder die Bibliotheken mithilfe eines Paketmanagers herunterladen.

Die SDK-Quelle ist nach Diensten in einzelne Pakete aufgeteilt. Die Installation des gesamten SDK kann bis zu einer Stunde dauern. Wenn Sie nur die spezifische Teilmenge der Dienste installieren, die Ihr Programm verwendet, verringert sich die Installationszeit und auch die Festplattengröße. Um auswählen zu können, welche Dienste installiert werden sollen, müssen Sie den Paketnamen der einzelnen Dienste kennen, die Ihr Programm verwendet. Die Liste der Paketverzeichnisse finden Sie unter `aws/aws-sdk-cpp`on GitHub. Der Paketname ist das Suffix des Verzeichnisnamens für den Dienst.

```
aws-sdk-cpp\aws-cpp-sdk-<packageName>   # Repo directory name and packageName
```

```
aws-sdk-cpp\aws-cpp-sdk-s3              # Example: Package name is s3
```

Voraussetzungen

Sie benötigen mindestens 4 GB RAM, um einige der größeren AWS Clients zu erstellen. Das SDK kann möglicherweise aufgrund unzureichenden Speichers nicht auf den Amazon EC2 EC2-Instance-Typen t2.micro, t2.small und anderen kleinen Instance-Typen aufbauen.

Um das verwenden zu können AWS SDK für C++, benötigen Sie eine der folgenden Optionen:

- Microsoft Visual Studio 2015 oder höher,
- GNU Compiler Collection (GCC) 4.9 oder höher oder
- Clang 3.3 oder höher.

Das SDK für Windows mit curl erstellen

Unter Windows wurde das SDK mit WinHTTP als Standard-HTTP-Client erstellt. WinHTTP 1.0 unterstützt jedoch kein bidirektionales HTTP/2-Streaming, das für einige Anwendungen AWS-Services wie Amazon Transcribe und Amazon Lex erforderlich ist. Daher ist es manchmal notwendig, die Curl-Unterstützung mit dem SDK aufzubauen. Eine Übersicht aller verfügbaren Curl-Download-Optionen finden Sie unter curl-Releases und -Downloads. Eine Methode zum Erstellen des SDK mit Curl-Unterstützung ist die folgende:

Um das SDK mit Unterstützung für die Curl-Bibliothek zu erstellen

1. Navigieren Sie zu curl für Windows und laden Sie das curl-Binärpaket für Microsoft Windows herunter.
2. Entpacken Sie das Paket in einen Ordner auf Ihrem Computer, z. B. `C:\curl`
3. Navigieren Sie zu den CA-Zertifikaten, die aus Mozilla extrahiert wurden, und laden Sie die `cacert.pem` Datei herunter. Diese Privacy Enhanced Mail (PEM) -Datei enthält ein Bündel gültiger digitaler Zertifikate, mit denen die Echtheit sicherer Websites überprüft wird. Die Zertifikate werden von Zertifizierungsstellen (CA) wie Verisign GlobalSign vertrieben.
4. Verschieben Sie die `cacert.pem` Datei in den `bin` Unterordner, den Sie in einem vorherigen Schritt entpackt haben, z. B. `C:\curl\bin` Benennen Sie die Datei um als. `curl-ca-bundle.crt`

Außerdem muss die Microsoft Build Engine (MSBuild) in der Lage sein, die Curl `dll` im folgenden Verfahren zu finden. Daher sollten Sie den Pfad zum `bin` Ordner curl zu Ihrer `PATH` Windows-Umgebungsvariablen hinzufügen, z. B. `set PATH=%PATH%;C:\curl\bin` Sie müssen dies jedes Mal hinzufügen, wenn Sie eine neue Befehlszeile öffnen, um das SDK zu erstellen. Alternativ können Sie die Umgebungsvariable global in Ihren Windows-Systemeinstellungen festlegen, sodass die Einstellung gespeichert wird.

Wenn Sie das SDK mit dem folgenden Verfahren aus dem Quellcode erstellen, finden Sie in Schritt 5 (Generieren von Build-Dateien) die erforderliche Befehlssyntax, um Curl in Ihr SDK einzubauen.

Wenn Sie Ihren Code schreiben, müssen Sie `caFile` AWS SDK for C++ C++-Serviceclients im Code konfigurieren den Speicherort Ihrer Zertifikatsdatei angeben. Ein Beispiel für die Verwendung von Amazon Transcribe finden Sie `transcribe-streaming`im AWS Codebeispiel-Repository unter. GitHub

Das SDK aus dem Quellcode erstellen

Sie können das SDK mithilfe von Befehlszeilentools aus dem Quellcode erstellen. Mit dieser Methode können Sie Ihren SDK-Build anpassen. Informationen zu den verfügbaren Optionen finden Sie unter CMake Parameter. Es gibt drei Hauptschritte. Zunächst erstellen Sie die Dateien mit CMake. Zweitens erstellen Sie MSBuild damit die SDK-Binärdateien, die mit Ihrem Betriebssystem und der Build-Toolchain funktionieren. Drittens installieren oder kopieren Sie die Binärdateien an den richtigen Ort auf dem Entwicklungscomputer.

Um das SDK aus dem Quellcode zu erstellen

1. Installieren Sie CMake(mindestens Version 3.13) und die entsprechenden Build-Tools für Ihre Plattform. Es wird empfohlen, es `cmake` zu Ihrem `PATH` hinzuzufügen. Um Ihre Version von zu überprüfen CMake, öffnen Sie eine Eingabeaufforderung und führen Sie den Befehl aus **`cmake --version`**
2. Navigieren Sie in einer Befehlszeile zu einem Ordner, in dem Sie das SDK speichern möchten.
3. Holen Sie sich den neuesten Quellcode.

   Version 1.11 verwendet Git-Submodule, um externe Abhängigkeiten zu umschließen. Dazu gehören die im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch beschriebenen CRT-Bibliotheken.

   Laden Sie die SDK-Quelle herunter oder klonen Sie sie von `aws/aws-sdk-cpp`: GitHub

- Mit Git klonen: HTTPS

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

- Mit Git klonen: SSH

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

4. Wir empfehlen, die generierten Build-Dateien außerhalb des SDK-Quellverzeichnisses zu speichern. Erstellen Sie ein neues Verzeichnis, in dem die Build-Dateien gespeichert werden, und navigieren Sie zu diesem Ordner.

```
mkdir sdk_build
cd sdk_build
```

5. Generieren Sie die Build-Dateien, indem Sie Folgendes ausführen`cmake`. Geben Sie in der `cmake` Befehlszeile an, ob eine Debug - oder Release-Version erstellt werden soll. Wählen Sie `Debug` in diesem Verfahren aus, ob Sie eine Debug-Konfiguration Ihres Anwendungscodes ausführen möchten. Wählen Sie `Release` in diesem Verfahren aus, ob Sie eine Release-Konfiguration Ihres Anwendungscodes ausführen möchten. Für Windows ist der SDK-Installationsort in der Regel`\Program Files (x86)\aws-cpp-sdk-all\`. Befehlssyntax:

   ```
   {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}
   ```

   Weitere Möglichkeiten, die Build-Ausgabe zu ändern, finden Sie unter CMakeParameter.

   Gehen Sie wie folgt vor, um die Build-Dateien zu generieren:

   - Generieren Sie Build-Dateien (alle AWS-Services): Um das gesamte SDK zu erstellen, führen Sie cmake aus und geben Sie an, ob eine Debug - oder Release-Version erstellt werden soll. Beispiel:

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

   - Generieren Sie Build-Dateien (Teilmenge AWS-Services): Um nur einen bestimmten Dienst oder ein bestimmtes Servicepaket (e) für das SDK zu erstellen, fügen Sie den CMake

BUILD_ONLY Parameter hinzu, wobei die Dienstnamen durch Semikolons getrennt sind. Das folgende Beispiel erstellt nur das Amazon S3 S3-Servicepaket:

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

- Generieren Sie Build-Dateien (mit Curl): Nachdem Sie die Curl-Voraussetzungen erfüllt haben, sind drei zusätzliche cmake-Befehlszeilenoptionen erforderlich, um die Curl-Unterstützung in das SDK aufzunehmen:FORCE_CURL, und. CURL_INCLUDE_DIR CURL_LIBRARY Beispiel:

```
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**
>
> Wenn die Fehlermeldung Fehler beim Erstellen von Bibliotheken von Drittanbietern angezeigt wird, überprüfen Sie Ihre Version von, indem Sie den Befehl ausführen. CMake **`cmake --version`** Sie müssen CMake mindestens Version 3.13 verwenden.

6. Erstellen Sie die SDK-Binärdateien. Wenn Sie das gesamte SDK erstellen, kann dieser Schritt eine Stunde oder länger dauern. Befehlssyntax:

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

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

   > **Note**
   >
   > Wenn Sie auf den Fehler stoßen Die Codeausführung kann nicht fortgesetzt werden... Die DLL wurde nicht gefunden. Durch eine Neuinstallation des Programms kann dieses Problem möglicherweise behoben werden.“ , wiederholen Sie den `cmake` Befehl erneut.

7. Öffnen Sie eine Befehlszeile mit Administratorrechten, um das SDK mithilfe des `CMAKE_PREFIX_PATH` Parameters an dem zuvor angegebenen Speicherort zu installieren. Befehlssyntax:

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

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

Bauen für Android unter Windows

Um für Android zu bauen, fügen Sie `-DTARGET_ARCH=ANDROID` es Ihrer `cmake` Befehlszeile hinzu. Die AWS SDK für C++ enthält eine CMake Toolchain-Datei, die alles enthält, was Sie benötigen, indem sie auf die entsprechenden Umgebungsvariablen () `ANDROID_NDK` verweist.

Um das SDK for Android unter Windows zu erstellen, müssen Sie es über eine Visual Studio-Entwickler-Befehlszeile (2015 oder höher) ausführen`cmake`. Außerdem müssen Sie NMAKE NMAKE installiert haben und die Befehle **`git`**müssen sich **`patch`**in Ihrem Pfad befinden. Wenn du Git auf einem Windows-System installiert hast, findest du es höchstwahrscheinlich **`patch`**in einem Geschwisterverzeichnis (). `.../Git/usr/bin/` Sobald du diese Anforderungen überprüft hast, ändert sich deine `cmake` Befehlszeile geringfügig, sodass du NMAKE verwendest.

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

NMAKE wird seriell erstellt. Um schneller zu bauen, empfehlen wir Ihnen, JOM als Alternative zu NMAKE zu installieren und dann den `cmake` Aufruf wie folgt zu ändern:

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

Eine Beispielanwendung finden Sie unter Eine Android-Anwendung einrichten mit AWS SDK für C++

Aufbau des AWS SDK für C++ auf Linux/macOS

Um das einzurichten AWS SDK für C++, können Sie das SDK entweder selbst direkt aus der Quelle erstellen oder die Bibliotheken mit einem Paketmanager herunterladen.

Die SDK-Quelle ist nach Diensten in einzelne Pakete aufgeteilt. Die Installation des gesamten SDK kann bis zu einer Stunde dauern. Wenn Sie nur die spezifische Teilmenge der Dienste installieren, die Ihr Programm verwendet, verringert sich die Installationszeit und auch die Festplattengröße. Um auswählen zu können, welche Dienste installiert werden sollen, müssen Sie den Paketnamen der einzelnen Dienste kennen, die Ihr Programm verwendet. Die Liste der Paketverzeichnisse finden Sie

unter aws/aws-sdk-cppon GitHub. Der Paketname ist das Suffix des Verzeichnisnamens für den Dienst.

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

Voraussetzungen

Sie benötigen mindestens 4 GB RAM, um einige der größeren AWS Clients zu erstellen. Das SDK kann möglicherweise aufgrund unzureichenden Speichers nicht auf den Amazon EC2 EC2-Instance-Typen t2.micro, t2.small und anderen kleinen Instance-Typen aufbauen.

Um das verwenden zu können AWS SDK für C++, benötigen Sie eine der folgenden Optionen:

- GNU Compiler Collection (GCC) 4.9 oder höher oder
- Clang 3.3 oder höher.

Zusätzliche Anforderungen für Linux-Systeme

Sie benötigen die Header-Dateien (`-dev`Pakete) für`libcurl`,`libopenssl`, `libuuid`zlib, und optional `libpulse` für den Amazon Polly Polly-Support. Sie können die Pakete mithilfe des Paketmanagers Ihres Systems finden.

Um die Pakete auf Debian/Ubuntu-basierten Systemen zu installieren

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

Um die Pakete auf Linux/Redhat/Fedora/CentOSAmazon-basierten Systemen zu installieren

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

Das SDK aus dem Quellcode erstellen

Als Alternative zur Verwendung von vcpkg können Sie das SDK mithilfe von Befehlszeilentools aus dem Quellcode erstellen. Mit dieser Methode können Sie Ihren SDK-Build anpassen. Informationen zu den verfügbaren Optionen finden Sie unter CMake Parameter.

Um das SDK aus dem Quellcode zu erstellen

1. Installieren Sie CMake(mindestens Version 3.13) und die entsprechenden Build-Tools für Ihre Plattform. Es wird empfohlen, es `cmake` zu Ihrem `PATH` hinzuzufügen. Um Ihre Version von zu überprüfen CMake, öffnen Sie eine Eingabeaufforderung und führen Sie den Befehl aus **`cmake --version`**
2. Navigieren Sie in einer Befehlszeile zu einem Ordner, in dem Sie das SDK speichern möchten.
3. Holen Sie sich den neuesten Quellcode.

   Version 1.11 verwendet Git-Submodule, um externe Abhängigkeiten zu umschließen. Dazu gehören die im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch beschriebenen CRT-Bibliotheken.

   Laden Sie die SDK-Quelle herunter oder klonen Sie sie von `aws/aws-sdk-cpp`: GitHub

   - Mit Git klonen: HTTPS

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

   - Mit Git klonen: SSH

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

4. Wir empfehlen, die generierten Build-Dateien außerhalb des SDK-Quellverzeichnisses zu speichern. Erstellen Sie ein neues Verzeichnis, in dem die Build-Dateien gespeichert werden, und navigieren Sie zu diesem Ordner.

   ```
   mkdir sdk_build
   cd sdk_build
   ```

5. Generieren Sie die Build-Dateien, indem Sie Folgendes ausführen`cmake`. Geben Sie in der `cmake` Befehlszeile an, ob eine Debug - oder Release-Version erstellt werden soll. Wählen Sie `Debug` in diesem Verfahren aus, ob Sie eine Debug-Konfiguration Ihres Anwendungscodes ausführen möchten. Wählen Sie `Release` in diesem Verfahren aus, ob Sie eine Release-Konfiguration Ihres Anwendungscodes ausführen möchten. Befehlssyntax:

   ```
   {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}
   ```

Weitere Möglichkeiten, die Build-Ausgabe zu ändern, finden Sie unter CMakeParameter.

> **Note**
>
> Wenn Sie auf einem Mac mit einem Dateisystem ohne Berücksichtigung der Groß- und Kleinschreibung kompilieren, überprüfen Sie die Ausgabe des `pwd` Befehls in dem Verzeichnis, in dem Sie den Build ausführen. Stellen Sie sicher, dass bei der `pwd` Ausgabe Groß- und Kleinschreibung für Verzeichnisnamen wie und verwendet wird. `/Users` `Documents`

Gehen Sie wie folgt vor, um die Build-Dateien zu generieren:

- Generieren Sie Build-Dateien (alle AWS-Services): Um das gesamte SDK zu erstellen, führen Sie cmake aus und geben Sie an, ob eine Debug - oder Release-Version erstellt werden soll. Beispiel:

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

- Generieren Sie Build-Dateien (Teilmenge AWS-Services): Um nur einen bestimmten Dienst oder ein bestimmtes Servicepaket (e) für das SDK zu erstellen, fügen Sie den CMake BUILD_ONLY Parameter hinzu, wobei die Dienstnamen durch Semikolons getrennt sind. Das folgende Beispiel erstellt nur das Amazon S3 S3-Servicepaket:

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

> **Note**
>
> Wenn Sie die Fehlermeldung Fehler beim Erstellen von Bibliotheken von Drittanbietern erhalten, überprüfen Sie Ihre Version von, CMake indem Sie Folgendes ausführen**`cmake --version`**. Sie müssen CMake mindestens Version 3.13 verwenden.

6. Erstellen Sie die SDK-Binärdateien. Wenn Sie das gesamte SDK erstellen, kann der Vorgang eine Stunde oder länger dauern.

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

7. Installieren Sie das SDK. Je nachdem, an welchem Ort Sie die Installation vorgenommen haben, müssen Sie möglicherweise Ihre Rechte erweitern.

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

Bauen für Android unter Linux

Um für Android zu bauen, fügen Sie `-DTARGET_ARCH=ANDROID` es Ihrer `cmake` Befehlszeile hinzu. Die AWS SDK für C++ enthält eine CMake Toolchain-Datei, die alles enthält, was Sie benötigen, indem sie auf die entsprechenden Umgebungsvariablen () `ANDROID_NDK` verweist. Eine Beispielanwendung finden Sie unter Einrichten einer Android-Anwendung mit AWS SDK für C++

Erstellen einer einfachen Anwendung mit dem AWS SDK for C++

CMakeist ein Build-Tool, mit dem Sie die Abhängigkeiten Ihrer Anwendung verwalten und Makefiles erstellen können, die für die Plattform geeignet sind, auf der Sie aufbauen. Sie können es verwenden CMake , um Projekte mit dem zu erstellen und zu erstellen. AWS SDK für C++

In diesem Beispiel werden die Amazon S3 S3-Buckets gemeldet, die Sie besitzen. Für dieses Beispiel ist es nicht erforderlich, einen Amazon S3 S3-Bucket in Ihrem AWS Konto zu haben, aber es ist weitaus interessanter, wenn Sie mindestens einen haben. Falls Sie noch keinen Bucket haben, finden Sie weitere Informationen unter Bucket erstellen im Amazon Simple Storage Service-Benutzerhandbuch.

Schritt 1: Schreiben Sie den Code

Dieses Beispiel besteht aus einem Ordner mit einer Quelldatei (`hello_s3.cpp`) und einer `CMakeLists.txt` Datei. Das Programm verwendet Amazon S3, um Speicher-Bucket-Informationen zu melden. Dieser Code ist auch im AWS Codebeispiel-Repository unter verfügbar GitHub.

Sie können viele Optionen in einer `CMakeLists.txt` Build-Konfigurationsdatei festlegen. Weitere Informationen finden Sie im CMakeTutorial auf der CMake Website.

> **Note**
>
> Deep Dive: Einstellung `CMAKE_PREFIX_PATH`

Standardmäßig ist das AWS SDK für C++ auf MacOS, Linux, Android und anderen Nicht-Windows-Plattformen in `/usr/local` und unter Windows installiert. `\Program Files (x86)\aws-cpp-sdk-all`
Wenn Sie das AWS SDK an diesen Standardspeicherorten installieren, findet es CMake automatisch die erforderlichen Ressourcen. Wenn Sie das AWS SDK jedoch an einem benutzerdefinierten Speicherort installieren, müssen Sie angeben, CMake wo sich die folgenden Ressourcen befinden, die sich aus der Erstellung des SDK ergeben:

- `AWSSDKConfig.cmake`: Eine Konfigurationsdatei, die erklärt, CMake wie Sie die AWS SDK-Bibliotheken in Ihrem Projekt finden und verwenden können. Ohne diese Datei CMake können Sie keine AWS SDK-Header-Dateien finden, keine Links zu AWS SDK-Bibliotheken herstellen oder die richtigen Compiler-Flags einrichten.
- (für Version 1.8 und früher) Der Speicherort der Abhängigkeiten:`aws-c-event-stream`,, `aws-c-common` `aws-checksums`

So legen Sie einen benutzerdefinierten Installationspfad fest:

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

Wenn Sie keine benutzerdefinierte Installation festlegen`CMAKE_PREFIX_PATH`, schlägt Ihr Build fehl und es werden Fehler wie „Konnte nicht gefunden AWSSDK“ angezeigt, wenn CMake versucht wird, `find_package(AWSSDK)` in Ihrem zu verarbeiten`CMakeLists.txt`.

Note

Tiefer Einblick: Windows-Runtime-Bibliotheken
Um Ihr Programm auszuführen, DLLs sind mehrere im ausführbaren Verzeichnis Ihres Programms erforderlich:`aws-c-common.dll`,,`aws-c-event-stream.dll`,`aws-checksums.dll`,`aws-cpp-sdk-core.dll`, sowie alle spezifischen, auf den Komponenten Ihres Programms DLLs basierenden Daten (dieses Beispiel erfordert auch, `aws-cpp-sdk-s3` weil es Amazon S3 verwendet). Die zweite `if` Anweisung in der `CMakeLists.txt` Datei kopiert diese Bibliotheken vom Installationsverzeichnis in das Verzeichnis der ausführbaren Datei, um diese Anforderung zu erfüllen. `AWSSDK_CPY_DYN_LIB`Sist ein durch definiertes Makro AWS SDK für C++ , das die SDKs DLLs vom Installationsverzeichnis in das Verzeichnis der ausführbaren Datei Ihres Programms kopiert. Wenn sich diese Dateien nicht

im Verzeichnis der ausführbaren Datei DLLs befinden, treten Laufzeitausnahmen wie „Datei nicht gefunden“ auf. Überprüfen Sie diesen Teil der `CMakeLists.txt` Datei auf notwendige Änderungen für Ihre spezielle Umgebung, falls Sie auf diese Fehler stoßen.

Um den Ordner und die Quelldateien zu erstellen

1. Erstellen Sie ein `hello_s3` and/or Verzeichnisprojekt für Ihre Quelldateien.

   > **Note**
   >
   > Um dieses Beispiel in Visual Studio zu vervollständigen: Wählen Sie Neues Projekt erstellen und dann CMake Projekt. Benennen Sie das Projekt `hello_s3`. Dieser Projektname wird in der `CMakeLists.txt` Datei verwendet.

2. Fügen Sie in diesem Ordner eine `hello_s3.cpp` Datei hinzu, die den folgenden Code enthält, der die Amazon S3 S3-Buckets meldet, die Sie besitzen.

```
#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. Fügen Sie eine `CMakeLists.txt` Datei hinzu, die den Namen Ihres Projekts, die ausführbaren Dateien, die Quelldateien und die verknüpften Bibliotheken angibt.

```
# 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})
```

Schritt 2: Erstellen mit CMake

CMake verwendet die darin enthaltenen Informationen`CMakeLists.txt`, um ein ausführbares Programm zu erstellen.

Wir empfehlen, die Anwendung gemäß den Standardverfahren für Ihre IDE zu erstellen.

Um die Anwendung über die Befehlszeile zu erstellen

1. Erstellen Sie ein Verzeichnis, in dem Ihre Anwendung erstellt **`cmake`**werden soll.

   ```
   mkdir my_project_build
   ```

2. Wechseln Sie in das Build-Verzeichnis und führen Sie es **`cmake`**mit dem Pfad zum Quellverzeichnis Ihres Projekts aus.

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

3. Nachdem Sie Ihr Build-Verzeichnis **`cmake`**generiert haben, können Sie **`make`**(oder **`nmake`**unter Windows) oder MSBUILD (`msbuild ALL_BUILD.vcxproj`oder`cmake --build . --config=Debug`) verwenden, um Ihre Anwendung zu erstellen.

Schritt 3: Ausführen

Wenn Sie diese Anwendung ausführen, zeigt sie eine Konsolenausgabe an, in der die Gesamtzahl der Amazon S3 S3-Buckets und der Name jedes Buckets aufgeführt sind.

Wir empfehlen, die Anwendung gemäß den Standardmethoden für Ihre IDE auszuführen.

> **Note**
>
> Denken Sie daran, sich anzumelden! Wenn Sie IAM Identity Center zur Authentifizierung verwenden, denken Sie daran, sich mit dem AWS CLI `aws sso login` folgenden Befehl anzumelden.

Um das Programm über die Befehlszeile auszuführen

1. Wechseln Sie in das Debug-Verzeichnis, in dem das Ergebnis des Builds generiert wurde.

2. Führen Sie das Programm mit dem Namen der ausführbaren Datei aus.

```
hello_s3
```

Weitere Beispiele für die AWS SDK für C++ Verwendung von finden Sie unterGeführte Beispiele für Aufrufe AWS-Services mit dem AWS SDK for C++.

AWS SDK für C++ Von einem Paketmanager bekommen

> **Important**
>
> Wenn Sie einen Paketmanager wie Homebrew oder vcpkg verwenden:
> Nachdem Sie das SDK for C++ auf eine neue Version aktualisiert haben, müssen Sie alle Bibliotheken oder ausführbaren Dateien, die vom SDK abhängen, neu kompilieren.

Um das einzurichten AWS SDK für C++, können Sie das SDK entweder selbst direkt aus der Quelle erstellen oder die Bibliotheken mithilfe eines Paketmanagers herunterladen.

Die SDK-Quelle ist nach Diensten in einzelne Pakete aufgeteilt. Die Installation des gesamten SDK kann bis zu einer Stunde dauern. Wenn Sie nur die spezifische Teilmenge der Dienste installieren, die Ihr Programm verwendet, verringert sich die Installationszeit und auch die Festplattengröße. Um auswählen zu können, welche Dienste installiert werden sollen, müssen Sie den Paketnamen der einzelnen Dienste kennen, die Ihr Programm verwendet. Die Liste der Paketverzeichnisse finden Sie unter `aws/aws-sdk-cpp`on GitHub. Der Paketname ist das Suffix des Verzeichnisnamens für den Dienst.

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

Voraussetzungen

Sie benötigen mindestens 4 GB RAM, um einige der größeren AWS Clients zu erstellen. Das SDK kann möglicherweise aufgrund unzureichenden Speichers nicht auf den Amazon EC2 EC2-Instance-Typen t2.micro, t2.small und anderen kleinen Instance-Typen aufbauen.

Linux/macOS

Um das AWS SDK für C++ auf Linux/macOS verwenden zu können, benötigen Sie eines der folgenden:

- GNU Compiler Collection (GCC) 4.9 oder höher, oder
- Clang 3.3 oder höher.

Windows

Um das AWS SDK für C++ unter Windows verwenden zu können, benötigen Sie eines der folgenden Geräte:

- Microsoft Visual Studio 2015 oder höher,
- GNU Compiler Collection (GCC) 4.9 oder höher oder
- Clang 3.3 oder höher.

Holen Sie sich das SDK mit vcpkg

> **Important**
>
> Die verfügbare vcpkg-Distribution wird von externen Mitwirkenden unterstützt und nicht über bereitgestellt. AWS Die neueste Version ist immer über die Installation aus dem Quellcode verfügbar.

vcpkg ist ein Paketmanager, der von externen Mitwirkenden aktualisiert und verwaltet wird. Beachten Sie, dass dieser Paketmanager nicht über die neueste verfügbare Version von bereitgestellt wird AWS und möglicherweise nicht die neueste verfügbare Version von widerspiegelt. AWS SDK für C++ Es gibt eine Verzögerung zwischen dem Zeitpunkt, an dem eine Version veröffentlicht wird, AWS und dem Zeitpunkt, an dem sie über einen externen Paketmanager verfügbar ist. Die neueste Version ist immer verfügbar, wenn sie von der Quelle aus installiert wird.

Sie müssen vcpkg auf Ihrem System installieren.

- Laden Sie vcpkg herunter und booten Sie es, indem Sie den Anweisungen in der GitHub vcpkg-Readme-Datei folgen. Ersetzen Sie dabei die folgenden Optionen, wenn Sie dazu aufgefordert werden:

- Im Rahmen dieser Anweisungen werden Sie aufgefordert, Folgendes einzugeben:

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

  Um das gesamte SDK zu installieren, geben Sie nur bestimmte Dienste des SDK ein, die installiert werden sollen, `.\vcpkg\vcpkg install "aws-sdk-cpp[*]" --recurse` oder geben Sie an, indem Sie einen Paketnamen in Klammern anhängen, z. B. `.\vcpkg\vcpkg install "aws-sdk-cpp[s3, ec2]" --recurse`

  In der Ausgabe werden Meldungen angezeigt, die Folgendes beinhalten:

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

- Kopieren Sie den vollständigen `-DCMAKE_TOOLCHAIN_FILE` Befehl, um ihn CMake später zu verwenden. In der GitHub vcpkg-Readme-Datei finden Sie auch Hinweise, wo Sie dies für Ihr Toolset verwenden können.
- Möglicherweise müssen Sie auch den Build-Konfigurationstyp notieren, den Sie über vcpkg installiert haben. Die Konsolenausgabe zeigt die Build-Konfiguration und die Version des SDK. Die folgende Beispielausgabe gibt an, dass die Build-Konfiguration „x86-Windows“ und die installierte AWS SDK für C++ Version 1.8 ist.

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

Nach der AWS SDK für C++ Installation von können Sie mithilfe des SDK Ihre eigene Anwendung entwickeln. Das Beispiel in Eine einfache Anwendung erstellen berichtet über die Amazon S3 S3-Buckets, die Sie besitzen.

Behebung von Problemen mit AWS SDK for C++ C++-Build

Beim Erstellen der AWS SDK für C++ Datei aus dem Quellcode können einige der folgenden häufigen Build-Probleme auftreten.

Themen

- CMake Fehler: Es konnte keine von AWSSDK "bereitgestellte Paketkonfigurationsdatei gefunden werden
- CMake Fehler: Ladedatei konnte nicht gefunden werden (und Sie verwenden SDK-Version 1.8)
- CMake Fehler: Ladedatei konnte nicht gefunden werden
- Laufzeitfehler: Der Vorgang kann nicht fortgesetzt werden, da aws-*.dll er nicht gefunden wurde

CMake Fehler: Es konnte keine von AWSSDK "bereitgestellte Paketkonfigurationsdatei gefunden werden

CMake löst den folgenden Fehler aus, wenn das installierte SDK nicht gefunden werden kann.

```
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.
```

Um diesen Fehler zu beheben, geben Sie an, CMake wo sich das installierte SDK befindet (z. B. der Ordner, der als Ergebnis der SDK-Installation generiert wurde (Windows, Linux/macOS)). Fügen Sie den folgenden Befehl vor dem ersten Aufruf von `find_package()` in Ihre Datei ein. `CMakeLists.txt` Ein Beispiel finden Sie unter ???.

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

CMake Fehler: Ladedatei konnte nicht gefunden werden (und Sie verwenden SDK-Version 1.8)

CMake löst den folgenden Fehler aus, wenn die installierten Bibliotheken nicht gefunden werden können.

```
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
```

Um diesen Fehler zu beheben, geben Sie an, CMake wo sich das installierte SDK befindet (z. B. der Ordner, der als Ergebnis der SDK-Installation generiert wurde (Windows, Linux/macOS)). Fügen Sie die folgenden Befehle vor dem ersten Aufruf von `find_package()` in Ihre Datei ein. `CMakeLists.txt` Ein Beispiel finden Sie unter ???.

```
#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()
```

Diese Lösung gilt nur für Version 1.8 des SDK, da diese Abhängigkeiten in späteren Versionen unterschiedlich behandelt werden. Version 1.9 behebt diese Probleme, indem eine Zwischenschicht zwischen den `aws-c-*` Bibliotheken `aws-sdk-cpp` und eingeführt wird. Diese neue Ebene heißt `aws-crt-cpp` und ist ein Git-Submodul des SDK for C++. `aws-crt-cpp`hat auch die `aws-c-*` Bibliotheken (einschließlich`aws-c-common`, `aws-checksums aws-c-event-stream`, usw.) als eigene Git-Submodule. Dadurch kann das SDK for C++ alle CRT-Bibliotheken rekursiv abrufen und den Build-Prozess verbessern.

CMake Fehler: Ladedatei konnte nicht gefunden werden

CMake löst den folgenden Fehler aus, wenn die installierten Bibliotheken nicht gefunden werden können.

```
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
```

Um diesen Fehler zu beheben, weisen Sie CMake an, gemeinsam genutzte Bibliotheken zu erstellen. Fügen Sie den folgenden Befehl vor dem ersten Aufruf von `find_package()` in Ihre `CMakeLists.txt` Datei ein. Ein Beispiel finden Sie unter ???.

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

Laufzeitfehler: Der Vorgang kann nicht fortgesetzt werden, da **`aws-*.dll`** er nicht gefunden wurde

CMake löst einen Fehler ähnlich dem folgenden aus, wenn eine erforderliche DLL nicht gefunden werden kann.

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

Dieser Fehler tritt auf, weil die erforderlichen Bibliotheken oder ausführbaren Dateien für das SDK for C++ nicht im selben Ordner wie die ausführbaren Dateien Ihrer Anwendung verfügbar sind. Um diesen Fehler zu beheben, kopieren Sie die SDK-Build-Ausgabe in Ihr ausführbares Verzeichnis. Der spezifische DLL-Dateiname des Fehlers hängt davon ab, welche AWS Dienste Sie verwenden. Führen Sie einen der folgenden Schritte aus:

- Kopieren Sie den Inhalt des `/bin` AWS SDK für C++ Installationsordners in den Build-Ordner Ihrer Anwendung.
- Verwenden Sie in Ihrer `CMakeLists.txt` Datei das Makro AWSSDK_CPY _DYN_LIBS, um diese für Sie zu kopieren.

  Fügen Sie einen Aufruf zu einer Datei `AWSSDK_CPY_DYN_LIBS(SERVICE_LIST "" ${CMAKE_CURRENT_BINARY_DIR})` oder `AWSSDK_CPY_DYN_LIBS(SERVICE_LIST "" ${CMAKE_CURRENT_BINARY_DIR}/${CMAKE_BUILD_TYPE})` zu Ihrer `CMakeLists.txt` Datei hinzu, damit dieses Makro das Kopieren für Sie übernimmt. Ein Beispiel finden Sie unter ???.

  Wählen Sie den richtigen Kopierpfad für Ihre Build-Umgebung. Beim Erstellen über die Befehlszeile wird die Build-Ausgabe häufig in einen Unterordner (`/Debug`) verschoben, Visual Studio und andere jedoch IDEs häufig nicht. Überprüfen Sie, wo sich Ihre ausführbaren

Ausgabedateien befinden, und stellen Sie sicher, dass das Makro an diesen Speicherort kopiert wird. Wenn Sie diese Art von Änderungen vornehmen, empfiehlt es sich, den Inhalt Ihres Build-Ausgabeverzeichnisses zu löschen, damit Sie einen sauberen Ausgangspunkt für den nächsten Build haben.

Konfiguration von Service-Clients im AWS SDK for C++

Für den programmgesteuerten Zugriff AWS-Services verwendet das AWS SDK for C++ jeweils eine Clientklasse. AWS-Service Wenn Ihre Anwendung beispielsweise auf Amazon zugreifen muss EC2, erstellt Ihre Anwendung ein EC2 Amazon-Client-Objekt als Schnittstelle zu diesem Service. Anschließend verwenden Sie den Service-Client, um Anfragen an dieses zu stellen AWS-Service.

Um eine Anfrage an einen zu stellen AWS-Service, müssen Sie zunächst einen Service-Client erstellen. Für jeden Code, den AWS-Service Sie verwenden, gibt es eine eigene Bibliothek und einen eigenen Typ für die Interaktion mit ihm. Der Client stellt für jeden API-Vorgang, der vom Dienst verfügbar gemacht wird, eine Methode zur Verfügung.

Es gibt viele alternative Möglichkeiten, das SDK-Verhalten zu konfigurieren, aber letztendlich hat alles mit dem Verhalten von Service-Clients zu tun. Jede Konfiguration hat keine Wirkung, bis ein aus ihnen erstellter Service-Client verwendet wird.

Sie müssen festlegen, wie sich Ihr Code authentifiziert AWS , wenn Sie mit AWS-Services entwickeln. Sie müssen auch festlegen, was AWS-Region Sie verwenden möchten.

Das AWS SDKs Referenzhandbuch für Tools enthält außerdem Einstellungen, Funktionen und andere grundlegende Konzepte, die in vielen der AWS SDKs Tools üblich sind.

Themen

- Allgemeine Konfiguration unter Verwendung Aws::SDKOptions von AWS SDK für C++
- Externes Konfigurieren des AWS SDK for C++ C++-Serviceclients
- AWS SDK for C++ C++-Serviceclients im Code konfigurieren
- Einstellung der AWS-Region für das AWS SDK for C++
- Verwenden des AWS SDK for C++ C++-Anmeldeinformationsanbieter
- CMake Parameter für den Aufbau des AWS SDK für C++
- Konfiguration und Verwendung der Protokollierung im AWS SDK for C++
- Überschreiben Sie Ihren HTTP-Client in der AWS SDK für C++
- Steuerung von Iostreams, die von HttpClient und AWSClient in der verwendet werden AWS SDK für C++
- Verwenden einer benutzerdefinierten libcrypto-Bibliothek in der AWS SDK für C++

Allgemeine Konfiguration unter Verwendung **`Aws::SDKOptions`** von AWS SDK für C++

Die `Aws::SDKOptions`Struktur enthält SDK-Konfigurationsoptionen. `Aws::SDKOptions`konzentriert sich auf die allgemeine SDK-Konfiguration, wohingegen sich die `ClientConfiguration`Struktur auf die Konfiguration der Kommunikation mit konzentriert AWS-Services.

Eine Instanz von `Aws::SDKOptions`wird an die `Aws::ShutdownAPI`Methoden `Aws::InitAPI` und übergeben. Dieselbe Instanz sollte an beide Methoden gesendet werden.

Die folgenden Beispiele veranschaulichen einige der verfügbaren Optionen.

- Schalten Sie die Protokollierung mit dem Standard-Logger ein

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

- Überschreiben Sie die standardmäßige HTTP-Client-Factory

```
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`verwendet eine Closure (auch als anonyme Funktion oder Lambda-Ausdruck bezeichnet) statt einer`std::shared_ptr`. Jede der SDK-Factory-Funktionen

funktioniert auf diese Weise, da der Speichermanager zu dem Zeitpunkt, zu dem die werkseitige Speicherzuweisung erfolgt, noch nicht installiert war. Wenn der Methode ein Closure übergeben wird, wird der Speichermanager aufgerufen, um die Speicherzuweisung vorzunehmen, wenn dies sicher ist. Eine einfache Methode, um dieses Verfahren durchzuführen, ist die Verwendung eines Lambda-Ausdrucks.

- Verwenden Sie einen globalen Handler `SIGPIPE`

  Wenn Sie das SDK for C++ mit curl und OpenSSL erstellen, müssen Sie einen Signal-Handler angeben. Wenn Sie keinen eigenen benutzerdefinierten Signal-Handler verwenden, setzen Sie `installSigPipeHandler` ihn auf. `true`

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

  Wenn `installSigPipeHandler` jatrue, verwendet das SDK for C++ einen Handler, der `SIGPIPE` Signale ignoriert. Weitere Informationen dazu finden Sie unter Operations Error Signals auf der Website des GNU-Betriebssystems. `SIGPIPE` Weitere Informationen zum curl-Handler finden Sie unter CURLOPT_NOSIGNAL, das auf der curl-Website erklärt wird.

  Die zugrunde liegenden Bibliotheken von curl und OpenSSL können ein `SIGPIPE` Signal senden, um zu benachrichtigen, wenn die Remote-Seite eine Verbindung schließt. Diese Signale müssen von der Anwendung verarbeitet werden. Weitere Informationen zu dieser Curl-Funktionalität finden Sie unter libcurl thread safety auf der curl-Website. Dieses Verhalten ist nicht automatisch in das SDK integriert, da Signalhandler für jede Anwendung global sind und die Bibliothek eine Abhängigkeit vom SDK ist.

Externes Konfigurieren des AWS SDK for C++ C++-Serviceclients

Viele Konfigurationseinstellungen können außerhalb Ihres Codes bearbeitet werden. Wenn die Konfiguration extern vorgenommen wird, wird die Konfiguration auf alle Ihre Anwendungen angewendet. Die meisten Konfigurationseinstellungen können entweder als Umgebungsvariablen oder in einer separaten gemeinsam genutzten AWS `config` Datei festgelegt werden. Die

gemeinsam genutzte `config` Datei kann separate Einstellungssätze, sogenannte Profile, enthalten, um unterschiedliche Konfigurationen für verschiedene Umgebungen oder Tests bereitzustellen.

Umgebungsvariablen und gemeinsam genutzte `config` Dateieinstellungen sind standardisiert und werden von allen AWS SDKs Tools gemeinsam genutzt, um konsistente Funktionen in verschiedenen Programmiersprachen und Anwendungen zu unterstützen.

Weitere Informationen zur Konfiguration Ihrer Anwendung mit diesen Methoden AWS SDKs sowie Einzelheiten zu den einzelnen SDK-übergreifenden Einstellungen finden Sie im Referenzhandbuch und im Tools-Referenzhandbuch. Alle Einstellungen, die das SDK anhand der Umgebungsvariablen oder Konfigurationsdateien auflösen kann, finden Sie in der Referenz zu Einstellungen im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch.

Um eine Anfrage an zu stellen AWS-Service, instanziieren Sie zunächst einen Client für diesen Dienst. Sie können allgemeine Einstellungen für Service-Clients wie Timeouts und den HTTP-Client konfigurieren und die Konfiguration erneut versuchen.

Jeder Dienstclient benötigt einen Anmeldeinformationsanbieter AWS-Region und einen Anmeldeinformationsanbieter. Das SDK verwendet diese Werte, um Anfragen an die richtige Region für Ihre Ressourcen zu senden und Anfragen mit den richtigen Anmeldeinformationen zu signieren. Sie können diese Werte programmgesteuert im Code angeben oder sie automatisch aus der Umgebung laden lassen.

Das SDK verfügt über eine Reihe von Stellen (oder Quellen), die überprüft werden, um einen Wert für Konfigurationseinstellungen zu finden.

1. Jede explizite Einstellung, die im Code oder auf einem Service-Client selbst festgelegt ist, hat Vorrang vor allen anderen Einstellungen.
2. Umgebungsvariablen
   - Einzelheiten zum Setzen von Umgebungsvariablen finden Sie unter Umgebungsvariablen im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch.
   - Beachten Sie, dass Sie Umgebungsvariablen für eine Shell auf verschiedenen Gültigkeitsebenen konfigurieren können: systemweit, benutzerweit und für eine bestimmte Terminalsitzung.
3. Geteilte Dateien und Dateien `config` `credentials`
   - Einzelheiten zum Einrichten dieser Dateien finden Sie im Referenzhandbuch „Gemeinsam genutzte *`configcredentials`* Dateien“AWS SDKs und „Tools“.
4. Jeder Standardwert, der vom SDK-Quellcode selbst bereitgestellt wird, wird zuletzt verwendet.

- Für einige Eigenschaften, z. B. Region, gibt es keine Standardeinstellung. Sie müssen sie entweder explizit im Code, in einer Umgebungseinstellung oder in der gemeinsam genutzten `config` Datei angeben. Wenn das SDK die erforderliche Konfiguration nicht auflösen kann, können API-Anfragen zur Laufzeit fehlschlagen.

> **Note**
>
> Alle Einstellungen, die das SDK aus den Umgebungsvariablen oder Konfigurationsdateien auflösen kann, finden Sie in der Referenz zu Einstellungen im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch.

AWS SDK for C++ C++-Serviceclients im Code konfigurieren

Wenn die Konfiguration direkt im Code abgewickelt wird, ist der Konfigurationsbereich auf die Anwendung beschränkt, die diesen Code verwendet. Innerhalb dieser Anwendung gibt es Optionen für die globale Konfiguration aller Service-Clients, die Konfiguration für alle Clients eines bestimmten AWS-Service Typs oder die Konfiguration für eine bestimmte Service-Client-Instanz.

AWS SDK für C++ Dazu gehören AWS-Service Clientklassen, die Funktionen für die Interaktion mit den Client-Klassen bereitstellen AWS-Services , die Sie in Ihrer Anwendung verwenden. Im SDK for C++ können Sie die Standard-Client-Konfiguration ändern, was hilfreich ist, wenn Sie Dinge tun möchten wie:

- Herstellen einer Internetverbindung über einen Proxy
- Ändern von HTTP-Transport-Einstellungen, z. B. Verbindungstimeout und wiederholte Anforderungsversuche
- Angabe von TCP-Socketpuffer-Größenhinweisen

`ClientConfiguration`ist eine Struktur im SDK for C++, die Sie instanziieren und in Ihrem Code verwenden können. Der folgende Ausschnitt veranschaulicht die Verwendung dieser Klasse für den Zugriff auf Amazon S3 über einen Proxy.

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

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

Deklarationen von Konfigurationsvariablen

Die `ClientConfiguration` Struktur deklariert die folgenden Mitgliedsvariablen:

```
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);
```

Beschreibungen der Konfigurationsvariablen

In der folgenden Liste werden die `ClientConfiguration` Mitgliedsvariablen beschrieben, mit denen Sie das Client-Verhalten anpassen können.

accountId

Gibt die AWS-Konto ID für das kontobasierte Endpunkt-Routing an. Verwenden Sie das Format 111122223333. Das kontobasierte Endpunkt-Routing verbessert die Anforderungsleistung für einige Dienste.

accountIdEndpointModus

Steuert das kontobasierte Endpunkt-Routing-Verhalten. Gültige Werte sind „erforderlich“, „deaktiviert“ oder „bevorzugt“. Der Standardwert ist „bevorzugt“. Stellen Sie ihn auf „deaktiviert“ ein, um das kontobasierte Endpunkt-Routing bei Bedarf zu deaktivieren.

allowSystemProxy

Steuert, ob der HTTP-Client die Proxyeinstellungen des Systems erkennt. Die Standardeinstellung ist falsch. Auf true setzen, um die automatische Proxyerkennung zu aktivieren.

appId

Gibt einen optionalen anwendungsspezifischen Bezeichner an. Wenn dieser Wert gesetzt ist, wird er im Format an den `User-Agent` Header angehängt. `App/{appId}` Sie können diesen Wert mit der `AWS_SDK_UA_APP_ID` Umgebungsvariablen oder dem `sdk_ua_app_id` Shared Config-Profilattribut festlegen.

cPath, CA-Datei

Weist den HTTP-Client an, wo er den Vertrauensspeicher für Ihr SSL-Zertifikat findet. Ein Beispiel für einen Trust Store könnte ein Verzeichnis sein, das mit dem `c_rehash` OpenSSL-Hilfsprogramm erstellt wurde. Diese Variablen sollten nicht gesetzt werden müssen, es sei denn, Ihre Umgebung verwendet Symlinks. Diese Variablen haben keine Auswirkung auf Windows- und MacOS-Systeme.

ChecksumConfig

Enthält Einstellungen für die Berechnung und Validierung von Prüfsummen. Beinhaltet `requestChecksumCalculation` und `responseChecksumValidation` mit dem Standardwert WHEN_SUPPORTED.

ConfigFactories

Spezifiziert Factory-Methoden für die Initialisierung von Client-Hilfsklassen wie und`Executor`. `RetryStrategy` Verwendet Standard-Factorys, sofern sie nicht überschrieben werden.

requestTimeoutMs und connectTimeoutMs

Gibt die Wartezeit in Millisekunden an, bis bei einer HTTP-Anfrage ein Timeout eintritt. Erwägen Sie beispielsweise, diese Zeiten bei der Übertragung großer Dateien zu verlängern.

credentialProviderConfig

Enthält Konfigurationseinstellungen für Anmeldeinformationsanbieter. Verwenden Sie diese Struktur, um anzupassen, wie das SDK Anmeldeinformationen erhält AWS .

disableExpectHeader

Gilt nur für CURL-HTTP-Clients. Standardmäßig fügt CURL einer HTTP-Anfrage den Header „Expect: 100-Continue“ hinzu, um zu verhindern, dass die HTTP-Payload in Situationen gesendet wird, in denen der Server unmittelbar nach Erhalt des Headers mit einem Fehler antwortet. Dieses Verhalten kann eine Hin- und Rückfahrt ersparen und ist in Situationen nützlich, in denen die Nutzlast gering ist und die Netzwerklatenz relevant ist. Die Standardeinstellung der Variablen ist False. Wenn der Wert auf true gesetzt ist, wird CURL angewiesen, sowohl den HTTP-Request-Header als auch den Hauptteil zusammen zu senden.

IMDs deaktivieren

Steuert, ob IMDS-Aufrufe (Instance Metadata Service) deaktiviert sind. Die Standardeinstellung ist falsch. Auf true setzen, um IMDS-Aufrufe zu deaktivieren, wenn sie außerhalb von EC2 Instanzen ausgeführt werden.

IMDSv1 deaktivieren

Steuert, ob IMDSv1 Anrufe beim Zulassen deaktiviert sind. IMDSv2 Die Standardeinstellung ist falsch. Auf true setzen, um aus Sicherheitsgründen nur IMDSv1 Anrufe zu deaktivieren.

enableClockSkewAnpassung

Steuert, ob der Taktversatz nach jedem HTTP-Versuch angepasst wird. Die Standardeinstellung ist falsch.

enableEndpointDiscovery

Steuert, ob Endpoint Discovery verwendet wird. Standardmäßig werden regionale oder überschriebene Endpunkte verwendet. Um die Endpunkterkennung zu aktivieren, setzen Sie die Variable auf true.

enableHostPrefixInjektion

Steuert, ob der HTTP-Host DiscoverInstances Anfragen ein „data-“ -Präfix hinzufügt. Dieses Verhalten ist standardmäßig aktiviert. Um es zu deaktivieren, setzen Sie die Variable auf False.

enableHttpClientTrace

Steuert, ob die HTTP-Client-Ablaufverfolgung für Debugging-Zwecke aktiviert ist. Die Standardeinstellung ist falsch. Auf true setzen, um eine detaillierte Protokollierung von HTTP-Anfragen und -Antworten zu aktivieren.

enableTcpKeepLebendig

Steuert, ob TCP-Keep-Alive-Pakete gesendet werden sollen. Die Standardeinstellung ist true. In Verbindung mit der `tcpKeepAliveIntervalMs` Variablen verwenden. Diese Variable gilt nicht für Win INet und den IXMLHTTPRequest2 Client.

EndpointOverride

Gibt einen übergeordneten HTTP-Endpunkt an, mit dem mit einem Dienst kommuniziert werden soll.

Testamentsvollstrecker

Verweist auf die Implementierung des asynchronen Executor-Handlers. Das Standardverhalten besteht darin, für jeden asynchronen Aufruf einen Thread zu erstellen und zu trennen. Um dieses Verhalten zu ändern, implementieren Sie eine Unterklasse der `Executor` Klasse und weisen Sie dieser Variablen eine Instanz zu.

Folgen Sie Weiterleitungen

Steuert das Verhalten beim Umgang mit HTTP 300-Weiterleitungscodes.

httpLibOverride

Gibt die HTTP-Implementierung an, die von der Standard-HTTP-Factory zurückgegeben wird. Der Standard-HTTP-Client für Windows ist WinHTTP. Der Standard-HTTP-Client für alle anderen Plattformen ist CURL.

httpLibPerfModus

Gibt den Leistungsmodus der HTTP-Bibliothek an. Die Standardeinstellung ist LOW_LATENCY. Sie können diese Einstellung anpassen, um sie für verschiedene Leistungsmerkmale zu optimieren.

httpRequestTimeoutFrau

Gibt das Timeout für HTTP-Anfragen in Millisekunden an. Der Standardwert ist 0 (kein Timeout). Erwägen Sie, diesen Wert zu erhöhen, wenn Sie große Dateien übertragen.

lowSpeedLimit

Gibt die minimal zulässige Übertragungsgeschwindigkeit in Byte pro Sekunde an. Wenn die Übertragungsgeschwindigkeit unter die angegebene Geschwindigkeit fällt, wird der Übertragungsvorgang abgebrochen. Die Standardeinstellung ist 1 Byte/Sekunde. Diese Variable gilt nur für CURL-Clients.

MaxConnections

Gibt die maximale Anzahl von HTTP-Verbindungen zu einem einzelnen Server an. Der Standardwert ist 25. Es gibt keinen anderen zulässigen Höchstwert als den, den Ihre Bandbreite vernünftigerweise unterstützen kann.

nonProxyHosts

Gibt eine Reihe von Hostnamen an, die die Proxyeinstellungen umgehen sollen. Verwenden Sie diese Einstellung, um bestimmte Hosts von der Proxykonfiguration auszuschließen.

Profilname

Gibt den AWS Profilnamen an, der für die Konfiguration verwendet werden soll. Das SDK lädt Einstellungen aus dem angegebenen Profil in die AWS Konfigurationsdatei.

proxyCaFile

Gibt den Pfad zur Zertifizierungsstellendatei für Proxyverbindungen an, falls dieser vom Standard abweicht.

proxyCaPath

Gibt den Pfad zum Vertrauensspeicher der Zertifizierungsstelle für Proxyverbindungen an, falls dieser vom Standard abweicht.

ProxyScheme, ProxyHost, ProxyPort und ProxyPassword proxyUserName

Wird verwendet, um einen Proxy für die gesamte Kommunikation mit einzurichten und zu konfigurieren. AWS Diese Funktionalität könnte beispielsweise beim Debuggen in Verbindung mit der Burp-Suite oder bei der Verwendung eines Proxys für die Verbindung mit dem Internet nützlich sein.

Proxy-Pfad SSLCert

Gibt den Pfad zur SSL-Zertifikatsdatei für Proxyverbindungen an, für die Client-Zertifikate erforderlich sind.

SSLCertProxy-Typ

Gibt den SSL-Zertifikatstyp für Proxyverbindungen an. Zu den gängigen Typen gehören PEM und DER.

Proxy-Passwort SSLKey

Gibt das Passwort für den privaten SSL-Schlüssel an, der bei Proxyverbindungen verwendet wird, wenn der Schlüssel kennwortgeschützt ist.

Proxy-Pfad SSLKey

Gibt den Pfad zur privaten SSL-Schlüsseldatei für Proxyverbindungen an, für die Client-Zertifikate erforderlich sind.

SSLKeyProxy-Typ

Gibt den Typ des privaten SSL-Schlüssels für Proxyverbindungen an. Zu den gängigen Typen gehören PEM und DER.

writeRateLimiter und readRateLimiter

Verweise auf die Implementierungen von Lese- und Schreibratenbegrenzern, die zur Drosselung der von der Transportschicht verwendeten Bandbreite verwendet werden. Standardmäßig werden die Lese- und Schreibraten nicht gedrosselt. Um die Drosselung einzuführen, implementieren Sie

eine Unterklasse von `RateLimiterInterface` und weisen Sie diesen Variablen eine Instanz zu.

Region

Gibt die zu verwendende AWS Region an, z. B. us-east-1. Standardmäßig ist die verwendete Region die Standardregion, die in den entsprechenden AWS Anmeldeinformationen konfiguriert ist.

requestCompressionConfig

Enthält Konfigurationseinstellungen für die Komprimierung von Anfragen. Verwenden Sie diese Struktur, um zu steuern, wann und wie Anfragen vor der Übertragung komprimiert werden.

Strategie erneut versuchen

Verweist auf die Implementierung der Wiederholungsstrategie. Die Standardstrategie implementiert eine exponentielle Backoff-Richtlinie. Um eine andere Strategie auszuführen, implementieren Sie eine Unterklasse der `RetryStrategy` Klasse und weisen Sie dieser Variablen eine Instanz zu.

scheme

Gibt das URI-Adressierungsschema an, entweder HTTP oder HTTPS. Das Standardschema ist HTTPS.

tcpKeepAliveIntervalMs

Gibt das Zeitintervall in Millisekunden an, in dem ein Keep-Alive-Paket über eine TCP-Verbindung gesendet werden soll. Das Standardintervall beträgt 30 Sekunden. Die Mindesteinstellung ist 15 Sekunden. Diese Variable gilt nicht für Win INet und den IXMLHTTPRequest2 Client.

Telemetrieanbieter

Verweist auf die Implementierung des Telemetrieanbieters zum Sammeln von Metriken und Nachverfolgungsdaten. Konfigurieren Sie diese Einstellung, um Observability-Funktionen zu aktivieren.

userAgent

Nur zur internen Verwendung. Ändern Sie die Einstellung dieser Variablen nicht.

useDualStack

Steuert, ob Dual-Stack IPv4 und IPv6 Endpoints verwendet werden sollen. Beachten Sie, dass nicht alle AWS Dienste IPv6 in allen Regionen unterstützt werden.

Verwenden Sie FIPS

Steuert, ob nach den Federal Information Processing Standards (FIPS) 140-2 validierte kryptografische Module verwendet werden sollen. Die Standardeinstellung ist False. Wird auf „true“ gesetzt, wenn FIPS-Konformität erforderlich ist.

SSL verifizieren

Steuert, ob SSL-Zertifikate verifiziert werden sollen. Standardmäßig werden SSL-Zertifikate verifiziert. Um die Überprüfung zu deaktivieren, setzen Sie die Variable auf false.

version

Gibt die HTTP-Version an, die für Anfragen verwendet werden soll. Die Standardeinstellung ist HTTP_VERSION_2TLS (HTTP/2 über TLS).

gewinnen HTTPOptions

Enthält Windows-spezifische HTTP-Konfigurationsoptionen. Beinhaltet useAnonymousAuth mit der Standardeinstellung false.

Einstellung der AWS-Region für das AWS SDK for C++

Sie können auf AWS-Services diese zugreifen, die in einem bestimmten geografischen Gebiet arbeiten, indem Sie AWS-Regionen Dies kann sowohl aus Gründen der Redundanz als auch dafür nützlich sein, dass Ihre Daten und Anwendungen in der Nähe ausgeführt werden, wo Sie und Ihre Benutzer darauf zugreifen.

> **Important**
>
> Die meisten Ressourcen befinden sich in einer bestimmten Region, AWS-Region und Sie müssen die richtige Region für die Ressource angeben, wenn Sie das SDK verwenden.

Beispiele dafür, wie Sie die Standardregion über die gemeinsam genutzte AWS `config` Datei oder Umgebungsvariablen festlegen, finden Sie AWS-Regionim Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch.

Sie müssen einen Standard festlegen, der AWS-Region für AWS Anfragen verwendet werden AWS SDK für C++ soll. Dieser Standard wird für alle Aufrufe von SDK-Dienstmethoden verwendet, die

nicht mit einer Region angegeben sind. Im SDK for C++ können Sie die Standardregion auch mithilfe von festlegenClient-Konfiguration im Code.

Verwenden des AWS SDK for C++ C++-Anmeldeinformationsanbieter

Alle Anfragen an AWS müssen kryptografisch signiert werden, wobei die Anmeldeinformationen verwendet werden müssen, die von ausgestellt wurden. AWS Zur Laufzeit ruft das SDK Konfigurationswerte für Anmeldeinformationen ab, indem es mehrere Speicherorte überprüft.

Die Authentifizierung mit AWS kann außerhalb Ihrer Codebasis erfolgen. Viele Authentifizierungsmethoden können vom SDK mithilfe der Credential Provider-Kette automatisch erkannt, verwendet und aktualisiert werden.

Anleitungen zu den ersten Schritten mit der AWS Authentifizierung für Ihr Projekt finden Sie unter Authentifizierung und Zugriff im AWS SDKs Referenzhandbuch zu Tools.

Die Kette der Anbieter von Anmeldeinformationen

Wenn Sie bei der Erstellung eines Clients nicht explizit einen Anmeldeinformationsanbieter angeben, verwendet das SDK for C++ eine Anmeldeinformationsanbieterkette, die eine Reihe von Stellen überprüft, an denen Sie Anmeldeinformationen angeben können. Sobald das SDK Anmeldeinformationen an einem dieser Orte findet, wird die Suche beendet.

Reihenfolge beim Abrufen der Anmeldeinformationen

Alle SDKs haben eine Reihe von Orten (oder Quellen), die sie überprüfen, um gültige Anmeldeinformationen zu erhalten, mit denen eine Anfrage an eine gestellt werden kann. AWS-Service Nachdem gültige Anmeldeinformationen gefunden wurden, wird die Suche beendet. Diese systematische Suche wird als Credential Provider Chain bezeichnet.

Für jeden Schritt in der Kette gibt es unterschiedliche Möglichkeiten, die Werte festzulegen. Das Setzen von Werten direkt im Code hat immer Vorrang, gefolgt von der Einstellung als Umgebungsvariablen und dann in der gemeinsam genutzten AWS `config` Datei. Weitere Informationen finden Sie unter Priorität der Einstellungen im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch.

Das SDK versucht, Anmeldeinformationen aus dem `[default]` Profil in die gemeinsam genutzten `credentials` Dateien AWS `config` und Dateien zu laden. Sie können die `AWS_PROFILE`

Umgebungsvariable verwenden, um ein benanntes Profil auszuwählen, das das SDK laden und nicht verwenden soll`[default]`. Die `credentials` Dateien `config` und werden von den Tools AWS SDKs und gemeinsam genutzt. Das Referenzhandbuch für AWS SDKs und Tools enthält Informationen zu den SDK-Konfigurationseinstellungen, die von all AWS SDKs und dem verwendet werden AWS CLI. Weitere Informationen zur Konfiguration des SDK mithilfe der gemeinsam genutzten AWS `config` Datei finden Sie unter Gemeinsam genutzte Konfigurations - und Anmeldeinformationsdateien. Weitere Informationen zur Konfiguration des SDK durch das Setzen von Umgebungsvariablen finden Sie unter Unterstützung von Umgebungsvariablen.

Zur Authentifizierung mit AWSüberprüft das SDK for C++ die Anmeldeinformationsanbieter in der folgenden Reihenfolge.

1. AWS Zugriffsschlüssel (temporäre und langfristige Anmeldeinformationen)

   Das SDK versucht, Anmeldeinformationen aus den `AWS_ACCESS_KEY_ID` `AWS_SESSION_TOKEN` Umgebungsvariablen und oder aus der gemeinsam genutzten AWS `credentials` Datei zu laden. `AWS_SECRET_ACCESS_KEY`

   - Anleitungen zur Konfiguration dieses Anbieters finden Sie in den AWS Zugriffstasten im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch.
   - Einzelheiten zu den SDK-Konfigurationseigenschaften für diesen Anbieter finden Sie unter AWS Zugriffstasten im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch.
2. AWS STS Web-Identität

   Beim Erstellen von mobilen Anwendungen oder clientbasierten Webanwendungen, für die Zugriff erforderlich ist AWS, gibt AWS -Security-Token-Service (AWS STS) einen Satz temporärer Sicherheitsanmeldedaten für Verbundbenutzer zurück, die über einen Public Identity Provider (IdP) authentifiziert wurden.

   - Wenn Sie dies in einem Profil angeben, versucht das SDK oder das Tool, temporäre Anmeldeinformationen mithilfe der API-Methode abzurufen. AWS STS `AssumeRoleWithWebIdentity` Einzelheiten zu dieser Methode finden Sie AssumeRoleWithWebIdentityin der AWS -Security-Token-Service API-Referenz.
   - Anleitungen zur Konfiguration dieses Anbieters finden Sie unter Federate with Web Identity oder OpenID Connect im AWS SDKs und Tools Reference Guide.
   - Einzelheiten zu den SDK-Konfigurationseigenschaften für diesen Anbieter finden Sie unter Assume role Credential Provider im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch.

3. IAM Identity Center

   Wenn Sie IAM Identity Center zur Authentifizierung verwenden, verwendet das SDK for C++ in diesem Fall das Single Sign-On-Token, das durch Ausführen AWS des CLI-Befehls eingerichtet wurde. `aws sso login` Das SDK verwendet die temporären Anmeldeinformationen, die das IAM Identity Center gegen ein gültiges Token ausgetauscht hat. Das SDK verwendet dann die temporären Anmeldeinformationen, wenn es aufruft AWS-Services. Ausführliche Informationen zu diesem Prozess finden Sie unter Grundlegendes zur Auflösung von SDK-Anmeldeinformationen im Referenzhandbuch AWS SDKs und AWS-Services im Tools-Referenzhandbuch.

   - Anleitungen zur Konfiguration dieses Anbieters finden Sie unter IAM Identity Center-Authentifizierung im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch.
   - Einzelheiten zu den SDK-Konfigurationseigenschaften für diesen Anbieter finden Sie unter IAM Identity Center-Anmeldeinformationsanbieter im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch.

4. Identity Resolver für Anmeldeinformationen mit Signin AWS

   Wenn Sie AWS-Anmelde- und Konsolenanmeldedaten zur Authentifizierung verwenden, verwendet das SDK for C++ in diesem Fall die Konsolenanmeldedaten, die durch Ausführen `aws login` oder `aws login --profile` in der CLI eingerichtet wurden. Das SDK verwendet diese Anmeldeinformationen, wenn es AWS Dienste aufruft.

   - Ausführliche Informationen zu diesem Vorgang finden Sie unter Anmeldung für die AWS lokale Entwicklung mit Konsolenanmeldedaten im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch.

5. Externer Prozessanbieter

   Dieser Anbieter kann verwendet werden, um benutzerdefinierte Implementierungen bereitzustellen, z. B. zum Abrufen von Anmeldeinformationen aus einem lokalen Anmeldeinformationsspeicher oder zur Integration mit Ihrem lokalen Identitätsanbieter.

   - Eine Anleitung zur Konfiguration dieses Anbieters finden Sie unter IAM Roles Anywhere im Referenzhandbuch zu Tools.AWS SDKs
   - Einzelheiten zu den SDK-Konfigurationseigenschaften für diesen Anbieter finden Sie unter Anbieter für Prozessanmeldedaten im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch.

6. Anmeldeinformationen für Amazon ECS- und Amazon EKS-Container

Ihren Amazon Elastic Container Service-Aufgaben und Kubernetes-Servicekonten kann eine IAM-Rolle zugewiesen werden. Die in der IAM-Rolle gewährten Berechtigungen werden von den Containern übernommen, die in der Aufgabe oder den Containern des Pods ausgeführt werden. Diese Rolle ermöglicht es Ihrem SDK for C++ C++-Anwendungscode (auf dem Container), andere zu verwenden AWS-Services.

Das SDK versucht, Anmeldeinformationen aus den `AWS_CONTAINER_CREDENTIALS_FULL_URI` Umgebungsvariablen `AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` oder abzurufen, die automatisch von Amazon ECS und Amazon EKS festgelegt werden können.

- Einzelheiten zur Einrichtung dieser Rolle für Amazon ECS finden Sie unter Amazon ECS-Aufgaben-IAM-Rolle im Amazon Elastic Container Service Developer Guide.
- Informationen zur Einrichtung von Amazon EKS finden Sie unter Einrichten des Amazon EKS Pod Identity Agent im Amazon EKS-Benutzerhandbuch.
- Einzelheiten zu den SDK-Konfigurationseigenschaften für diesen Anbieter finden Sie unter Container Credential Provider im AWS SDKs und Tools Reference Guide.

7. Metadatenservice für EC2 Amazon-Instances

   Erstellen Sie eine IAM-Rolle und fügen Sie sie Ihrer Instance hinzu. Das SDK for C++ C++-Anwendung auf der Instanz versucht, die von der Rolle bereitgestellten Anmeldeinformationen aus den Instanzmetadaten abzurufen.

   - Einzelheiten zur Einrichtung dieser Rolle und zur Verwendung von Metadaten, IAM-Rollen für Amazon EC2 und Work with Instance-Metadaten finden Sie im EC2 Amazon-Benutzerhandbuch.
   - Einzelheiten zu den SDK-Konfigurationseigenschaften für diesen Anbieter finden Sie unter IMDS-Anmeldeinformationsanbieter im Referenzhandbuch AWS SDKs und im Tools-Referenzhandbuch.

Die Credential Provider-Kette kann `AWSCredentialsProviderChain`im AWS SDK für C++ Quellcode unter eingesehen werden. GitHub

Wenn Sie für den Einstieg den empfohlenen Ansatz für neue Benutzer befolgt haben, richten Sie die Authentifizierung mit den AWS Anmeldeinformationen während Authentifizierung mit AWS AWS SDK for C++ des Themas Erste Schritte ein. Andere Authentifizierungsmethoden sind in verschiedenen Situationen nützlich. Um Sicherheitsrisiken zu vermeiden, empfehlen wir, immer kurzfristige

Anmeldeinformationen zu verwenden. Informationen zu anderen Authentifizierungsmethoden finden Sie unter Authentifizierung und Zugriff im AWS SDKs Referenzhandbuch zu Tools.

Anbieter expliziter Anmeldeinformationen

Anstatt sich bei der Erkennung Ihrer Authentifizierungsmethode auf die Kette der Anmeldeinformationsanbieter zu verlassen, können Sie einen bestimmten Anmeldeinformationsanbieter angeben, den das SDK verwenden soll. Sie können dies tun, indem Sie Anmeldeinformationen im Konstruktor Ihres Service-Clients angeben.

Im folgenden Beispiel wird ein Amazon Simple Storage Service-Client erstellt, indem temporäre Zugangsdaten direkt bereitgestellt werden, anstatt die Kette zu verwenden.

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

Zwischenspeichern von Identitäten

Das SDK speichert Anmeldeinformationen und andere Identitätstypen wie SSO-Token im Cache. Standardmäßig verwendet das SDK eine Lazy-Cache-Implementierung, die Anmeldeinformationen bei der ersten Anfrage lädt, zwischenspeichert und dann versucht, sie bei einer weiteren Anfrage zu aktualisieren, wenn sie fast ablaufen. Clients, die auf derselben Grundlage erstellt wurden, `Aws::Client::ClientConfiguration`teilen sich einen Cache.

CMake Parameter für den Aufbau des AWS SDK für C++

Verwenden Sie die in diesem Abschnitt aufgeführten CMakeParameter, um anzupassen, wie Ihr SDK erstellt wird.

Sie können diese Optionen mit CMake GUI-Tools oder über die Befehlszeile mithilfe von -D festlegen. Zum Beispiel:

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

Allgemeine CMake Variablen und Optionen

Im Folgenden finden Sie allgemeine **`cmake`**Variablen und Optionen, die sich auf den Erstellungsprozess des SDK-Quellcodes auswirken.

> **Note**
>
> Verwenden Sie diese Parameter, wenn Sie den SDK-Quellcode für das SDK for C++ selbst erstellen.

Themen

- ADD_CUSTOM_CLIENTS
- AUTORUN_UNIT_TESTS
- AWS_AUTORUN_LD_LIBRARY_PATH
- AWS_SDK_WARNUNGEN_SIND_FEHLER
- 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
- IMDSV1DEAKTIVIEREN_INTERNAL_ _AUFRUFE
- ENABLE_ADDRESS_SANITIZER

- ENABLE_CURL_LOGGING
- ENABLE_HTTP_CLIENT_TESTING
- ENABLE_RTTI
- ENABLE_TESTING
- ENABLE_UNITY_BUILD
- VIRTUELLE_OPERATIONEN AKTIVIEREN
- 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

Erstellt beliebige Clients auf der Grundlage der API-Definition. Platzieren Sie Ihre Definition in dem `code-generation/api-definitions` Ordner und übergeben Sie dann dieses Argument an **`cmake`**. Der **`cmake`**Konfigurationsschritt generiert Ihren Client und nimmt ihn als Unterverzeichnis in Ihren Build auf. Dies ist besonders nützlich, um einen C++-Client für die Verwendung eines Ihrer API-Gateway-Dienste zu generieren. Beispiel:

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

> **Note**
>
> Um den `ADD_CUSTOM_CLIENTS` Parameter verwenden zu können, müssen Sie Python 2.7, Java (JDK 1.8+) und Maven installiert haben und in Ihrem. `PATH`

AUTORUN_UNIT_TESTS

Falls`ON`, führen Sie die Komponententests nach dem Erstellen automatisch aus.

Werte

EIN | AUS

Standard

AN

AWS_AUTORUN_LD_LIBRARY_PATH

Der Pfad, der an LD_LIBRARY_PATH für Komponententests angehängt werden soll, mit denen Autorun ausgeführt wird. CMake Geben Sie diesen Pfad an, wenn benutzerdefinierte Laufzeitbibliotheken für überschriebene Abhängigkeiten erforderlich sind.

Werte

Zeichenfolge.

Standard

N/A

AWS_SDK_WARNUNGEN_SIND_FEHLER

Falls, behandeln Sie Compiler-Warnungen `ON` als Fehler. Versuchen Sie, dies zu aktivieren`OFF`, wenn Sie Fehler auf einem neuen oder ungewöhnlichen Compiler beobachten.

Werte

EIN | AUS

Standard

AN

AWS_USE_CRYPTO_SHARED_LIBS

Erzwingt FindCrypto , eine gemeinsam genutzte Kryptobibliothek zu verwenden, falls sie gefunden wird. Schalten Sie diese Option `OFF` ein, um stattdessen BUILD_SHARED_LIBS die Einstellung zu verwenden.

Werte

EIN | AUS

Standard

AUS

AWS_TEST_REGION

Das AWS-Region , das für Integrationstests verwendet werden soll.

Werte

Zeichenfolge.

Standard

N/A

BUILD_BENCHMARKS

Wenn`ON`, erstellen Sie die ausführbare Benchmark-Datei.

Werte

EIN | AUS

Standard

AUS

BUILD_DEPS

Wenn0N, erstellen Sie Abhängigkeiten von Drittanbietern.

Werte

EIN | AUS

Standard

AN

BUILD_ONLY

Erstellt nur die Clients, die Sie verwenden möchten. Wenn BUILD_ONLY auf ein High-Level-SDK gesetzt ist*aws-cpp-sdk-transfer*, löst es alle Client-Abhängigkeiten auf niedriger Ebene auf. Es erstellt auch Integrations- und Komponententests, die sich auf die von Ihnen ausgewählten Projekte beziehen, sofern diese existieren. Dies ist ein Listenargument mit Werten, die durch Semikolons () `;` getrennt sind. Beispiel:

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

> **Note**
>
> Das Kern-SDK-Modul,*aws-sdk-cpp-core*, wird immer erstellt, unabhängig vom Wert des Parameters BUILD_ONLY.

BUILD_OPTEL

Wenn0N, erstellt die offene Telemetrie-Implementierung von Tracing.

Werte

EIN | AUS

Standard

AUS

BUILD_SHARED_LIBS

Eine integrierte CMake Option, die hier aus Gründen der Sichtbarkeit erneut verfügbar gemacht wird. Falls`ON`, erstellt sie gemeinsam genutzte Bibliotheken; andernfalls werden nur statische Bibliotheken erstellt.

> **Note**
>
> Um dynamisch auf das SDK zu verlinken, müssen Sie das `USE_IMPORT_EXPORT` Symbol für alle Build-Ziele definieren, die das SDK verwenden.

Werte

AN | AUS

Standard

AN

BYPASS_DEFAULT_PROXY

Falls`ON`, umgehen Sie die Standard-Proxyeinstellungen des Computers, wenn Sie 2 verwenden. IXml HttpRequest

Werte

EIN | AUS

Standard

AN

CPP_STANDARD

Gibt einen benutzerdefinierten C++-Standard für die Verwendung mit den Codebasen C++ 14 und 17 an.

Werte

11 | 14 | 17

Standard

11

CURL_INCLUDE_DIR

Pfad zum Curl-Include-Verzeichnis, das Header enthält. `libcurl`

Werte

Zeichenkettenpfad zum ausgewählten *`include`* Verzeichnis. Zum Beispiel*`D:/path/to/dir/with/curl/include`*.

Standard

N/A

CURL_LIBRARY

Pfad zur Curl-Bibliotheksdatei, gegen die verlinkt werden soll. Diese Bibliothek kann je nach den Anforderungen Ihrer Anwendung eine statische Bibliothek oder eine Importbibliothek sein.

Werte

Zeichenkettenpfad zur CURL-Bibliotheksdatei. Zum Beispiel. *`D:/path/to/static/libcur/file/ie/libcurl.lib.a`*

Standard

N/A

CUSTOM_MEMORY_MANAGEMENT

Um einen benutzerdefinierten Speichermanager zu verwenden, setzen Sie den Wert auf. `1` Sie können einen benutzerdefinierten Allocator installieren, sodass alle STL-Typen die benutzerdefinierte Zuweisungsschnittstelle verwenden. Wenn Sie den Wert festlegen`0`, möchten Sie möglicherweise trotzdem die STL-Vorlagentypen verwenden, um die DLL-Sicherheit unter Windows zu verbessern.

Wenn die statische Verknüpfung aktiviert ist`ON`, ist die benutzerdefinierte Speicherverwaltung standardmäßig auf off () `0` eingestellt. Wenn die dynamische Verknüpfung aktiviert ist`ON`, ist die benutzerdefinierte Speicherverwaltung standardmäßig aktiviert (1), wodurch DLL-übergreifende Zuweisungen und Freigaben vermieden werden.

> **Note**
>
> Um Linker-Fehlanpassungen zu vermeiden, müssen Sie in Ihrem gesamten Build-System denselben Wert (`0`oder`1`) verwenden.

Um Ihren eigenen Speichermanager für die Verwaltung der vom SDK vorgenommenen Zuweisungen zu installieren, müssen Sie `USE_AWS_MEMORY_MANAGEMENT` für alle Build-Ziele, die vom SDK abhängen, festlegen `-DCUSTOM_MEMORY_MANAGEMENT` und definieren.

IMDSV1DEAKTIVIEREN_INTERNAL_ _AUFRUFE

Falls`ON`, werden keine internen Aufrufe an die V1-API des Instanz-Metadatendienstes getätigt. Wenn`OFF`, wird bei IMDSv2 Aufrufen auf die Verwendung zurückgegriffen, IMDSv1 falls der IMDSv2 Aufruf fehlschlägt. Weitere Informationen zu IMDSv1 und IMDSv2 finden Sie unter Verwenden des Instance-Metadaten-Service für den Zugriff auf Instance-Metadaten im Amazon EC2 EC2-Benutzerhandbuch.

Werte

EIN | AUS

Standard

AUS

ENABLE_ADDRESS_SANITIZER

Wenn`ON`, aktiviert Address Sanitizer für GCC oder Clang.

Werte

EIN | AUS

Standard

AUS

ENABLE_CURL_LOGGING

Falls`ON`, leiten Sie das interne Protokoll für curl an den SDK-Logger weiter.

Werte

EIN | AUS

Standard

AUS

ENABLE_HTTP_CLIENT_TESTING

Wenn`ON`, erstellen Sie entsprechende HTTP-Client-Testsuiten und führen Sie sie aus.

Werte

EIN | AUS

Standard

AUS

ENABLE_RTTI

Steuert, ob das SDK so konzipiert ist, dass es Runtime Type Information (RTTI) aktiviert.

Werte

EIN | AUS

Standard

AN

ENABLE_TESTING

Steuert, ob Einheiten- und Integrationstestprojekte während des SDK-Builds erstellt werden.

Werte

EIN | AUS

Standard

AN

ENABLE_UNITY_BUILD

Wenn`ON`, werden die meisten SDK-Bibliotheken als eine einzelne, generierte Datei erstellt. `.cpp` Dies kann die Größe der statischen Bibliothek erheblich reduzieren und die Kompilierungszeit beschleunigen.

Werte

EIN | AUS

Standard

AUS

VIRTUELLE_OPERATIONEN AKTIVIEREN

Dieser Parameter funktioniert normalerweise zusammen mit für die Codegenerierung. `REGENERATE_CLIENTS`

Falls `ENABLE_VIRTUAL_OPERATIONS` ja `ON` und `REGENERATE_CLIENTS` ist`ON`, werden betriebsbezogene Funktionen in Service-Clients als gekennzeichnet. `virtual`

Falls `ENABLE_VIRTUAL_OPERATIONS` ist `OFF` und `REGENERATE_CLIENTS` ist`ON`, werden `virtual` nicht zu den Betriebsfunktionen hinzugefügt, und Serviceclient-Klassen werden als markiert. `final`

Falls ja `ENABLE_VIRTUAL_OPERATIONS``OFF`, fügt das SDK beim Kompilieren `-ffunction-sections` auch `-fdata-sections` Compiler-Flags für GCC und Clang hinzu.

Weitere Informationen finden Sie unter Parameter für. CMake GitHub

Werte

EIN | AUS

Standard

AN

ENABLE_ZLIB_REQUEST_COMPRESSION

Für Dienste, die dies unterstützen, wird der Inhalt der Anfrage komprimiert. Standardmäßig aktiviert, wenn eine Abhängigkeit verfügbar ist.

Werte

AN | AUS

Standard

AN

FORCE_CURL

Nur Windows. Wenn0N, erzwingt die Verwendung des Curl-Clients anstelle des standardmäßigen WinHTTP-Datenübertragungsanbieters.

Werte

EIN | AUS

Standard

AUS

FORCE_SHARED_CRT

Falls0N, verknüpft sich das SDK dynamisch mit der C-Laufzeit; andernfalls verwendet es die Einstellung BUILD_SHARED_LIBS (manchmal notwendig für die Abwärtskompatibilität mit früheren Versionen des SDK).

Werte

AN | AUS

Standard

AN

G

Generiert Build-Artefakte wie Visual Studio-Lösungen und Xcode-Projekte.

Zum Beispiel unter Windows:

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

Weitere Informationen finden Sie in der CMake Dokumentation zu Ihrer Plattform.

MINIMIZE_SIZE

Eine Obermenge von ENABLE_UNITY_BUILD. Falls`ON`, aktiviert diese Option ENABLE_UNITY_BUILD und zusätzliche Einstellungen für die binäre Größenreduzierung.

Werte

EIN | AUS

Standard

AUS

NO_ENCRYPTION

Wenn`ON`, verhindert, dass die standardmäßige plattformspezifische Kryptografie-Implementierung in die Bibliothek integriert wird. Schalten Sie diese Option ein, um Ihre eigene Kryptografie-Implementierung einzufügen.

Werte

EIN | AUS

Standard

AUS

NO_HTTP_CLIENT

Wenn`ON`, verhindert, dass der standardmäßige plattformspezifische HTTP-Client in die Bibliothek integriert wird. Bei der Einstellung ON müssen Sie Ihre eigene plattformspezifische HTTP-Client-Implementierung bereitstellen.

Werte

EIN | AUS

Standard

AUS

REGENERATE_CLIENTS

Falls`ON`, löscht dieser Parameter den gesamten generierten Code und generiert die Client-Verzeichnisse aus dem Ordner. `code-generation/api-definitions` Beispiel:

```
-DREGENERATE_CLIENTS=1
```

> **Note**
>
> Um den `REGENERATE_CLIENTS` Parameter verwenden zu können, müssen Sie Python 2.7, Java (JDK 1.8+) und Maven installiert haben und in Ihrem. `PATH`

REGENERATE_DEFAULTS

Falls`ON`, löscht dieser Parameter den gesamten generierten Standardcode und generiert ihn erneut aus dem Ordner. `code-generation/defaults` Beispiel:

```
-DREGENERATE_DEFAULTS=1
```

> **Note**
>
> Um den `REGENERATE_DEFAULTS` Parameter verwenden zu können, müssen Sie Python 2.7, Java (JDK 1.8+) und Maven installiert haben und in Ihrem. `PATH`

SIMPLE_INSTALL

Falls`ON`, fügt der Installationsvorgang keine plattformspezifischen Zwischenverzeichnisse unter und ein. `bin/` `lib/` Wählen Sie `OFF` diese Option, wenn Sie plattformübergreifende Releases in einem einzigen Installationsverzeichnis erstellen müssen.

Werte

EIN | AUS

Standard

AN

TARGET_ARCH

Um eine Cross-Compilierung oder einen Build für eine mobile Plattform durchzuführen, müssen Sie die Zielplattform angeben. Standardmäßig erkennt der Build das Host-Betriebssystem und erstellt für das erkannte Betriebssystem.

> **Note**
>
> Wenn TARGET_ARCH ANDROID ist, sind zusätzliche Optionen verfügbar. Siehe CMake Android-Variablen und -Optionen.

Werte

WINDOWS | LINUX | APPLE | ANDROID

USE_CRT_HTTP_CLIENT

Falls`ON`, verwenden Sie den gängigen Runtime-HTTP-Client, und die Legacy-Systeme wie libcurl sind weder gebaut WinHttp noch enthalten.

Werte

AN | AUS

Standard

AUS

USE_IXML_HTTP_REQUEST_2

Nur Windows. Falls`ON`, verwenden Sie das COM-Objekt IXml HttpRequest 2 für den HTTP-Stack.

Werte

AN | AUS

Standard

AUS

USE_OPENSSL

Falls ON, erstellt das SDK mit OpenSSL; andernfalls verwendet awslabs/aws-lc es. AWS-LC ist eine kryptografische Bibliothek für allgemeine Zwecke, die vom Cryptography-Team und AWS seinen Kunden verwaltet wird. AWS Wenn OFF Sie den Parameter aktivieren, wird OpenSSL AWS-LC als Ersatz für OpenSSL im Standardverzeichnis des Systems installiert. Verwenden Sie es nicht, wenn Sie bereits eine OpenSSL-Installation in Ihrem System haben.

Werte

EIN | AUS

Standard

AN

USE_TLS_V1_2

Falls ON, erzwingt der HTTP-Client TLS 1.2.

Werte

EIN | AUS

Standard

AN

USE_TLS_V1_3

Falls ON, erzwingt der HTTP-Client TLS 1.3.

Werte

EIN | AUS

Standard

AUS

CMake Android-Variablen und -Optionen

Verwenden Sie die folgenden Variablen, wenn Sie einen Android-Build des SDK erstellen (wenn TARGET_ARCH auf ANDROID gesetzt ist).

Themen

- ANDROID_ABI
- ANDROID_BUILD_CURL
- ANDROID_BUILD_OPENSSL
- ANDROID_BUILD_ZLIB
- ANDROID_NATIVE_ API_LEVEL
- ANDROID_STL
- NAME DER ANDROID_TOOLCHAIN
- DEAKTIVIEREN_ANDROID_STANDALONE_BUILD
- NDK_DIR

ANDROID_ABI

Nur Android. Steuert, für welche Application Binary Interface (ABI) Code ausgegeben werden soll.

> **Note**
>
> Derzeit werden nicht alle gültigen Android-ABI-Werte unterstützt.

Werte

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

Standard

armeabi-v7a

ANDROID_BUILD_CURL

Nur Android. Wenn`ON`, baue auch Curl.

Werte

EIN | AUS

Standard

AN

ANDROID_BUILD_OPENSSL

Nur Android. Wenn`ON`, baue auch Openssl.

Werte

AN | AUS

Standard

AN

ANDROID_BUILD_ZLIB

Nur Android. Wenn`ON`, baue auch Zlib.

Werte

EIN | AUS

Standard

AN

ANDROID_NATIVE_ API_LEVEL

Nur Android. Steuert, auf welcher API-Ebene das SDK baut. Wenn Sie ANDROID_STL auf gnustl setzen, können Sie eine beliebige API-Ebene wählen. Wenn Sie libc++ verwenden, müssen Sie eine API-Stufe von mindestens 21 verwenden.

Standard

Variiert je nach STL-Auswahl.

ANDROID_STL

Nur Android. Steuert, welche Variante der C++-Standardbibliothek das SDK verwendet.

> **Important**
>
> Leistungsprobleme können innerhalb des SDK auftreten, wenn die *gnustl* Optionen verwendet werden. Wir empfehlen dringend, libc++_shared oder libc++_static zu verwenden.

Werte

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

Standard

libc++_shared

NAME DER ANDROID_TOOLCHAIN

Nur Android. Steuert, welcher Compiler zum Erstellen des SDK verwendet wird.

> **Note**
>
> Da GCC vom Android NDK veraltet ist, empfehlen wir, den Standardwert zu verwenden.

Standard

eigenständiger Clang

DEAKTIVIEREN_ANDROID_STANDALONE_BUILD

Nur Android. Standardmäßig verwenden Android-Builds eine eigenständige, auf Clans basierende Toolchain, die über NDK-Skripts erstellt wurde. Um Ihre eigene Toolchain zu verwenden, schalten Sie diese Option ein.

Werte

EIN | AUS

Standard

AUS

NDK_DIR

Nur Android. Gibt einen Override-Pfad an, in dem das Build-System das Android-NDK finden soll. Standardmäßig überprüft das Build-System die Umgebungsvariablen (`ANDROID_NDK`), wenn diese Variable nicht gesetzt ist.

Konfiguration und Verwendung der Protokollierung im AWS SDK for C++

AWS SDK für C++ Dazu gehört eine konfigurierbare Protokollierung, die eine Aufzeichnung der Aktionen generiert, die das SDK während der Ausführung ausführt. Um die Protokollierung zu aktivieren, setzen Sie für `LogLevel` of `SDKOptions` die für Ihre Anwendung passende Ausführlichkeit.

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

Es stehen sieben Ausführlichkeitsstufen zur Auswahl. Der Standardwert ist `Off` und es werden keine Protokolle generiert. `Trace`generiert die meisten Details und generiert die wenigsten Meldungen, in denen nur schwerwiegende Fehler gemeldet `Fatal` werden.

Sobald die Protokollierung in Ihrer Anwendung aktiviert ist, generiert das SDK Protokolldateien in Ihrem ausführbaren Verzeichnis nach dem Standardbenennungsmuster von`aws_sdk_<date>.log`. Die mit der Präfixbenennungsoption generierte Protokolldatei wird einmal pro Stunde aktualisiert, um das Archivieren oder Löschen von Protokolldateien zu ermöglichen.

Die späteren Versionen des SDK hängen zunehmend von den zugrunde liegenden AWS Common Runtime (CRT) -Bibliotheken ab. Diese Bibliotheken bieten gemeinsame Funktionen und grundlegende Operationen unter SDKs. Alle Protokollnachrichten aus den CRT-Bibliotheken werden standardmäßig an das SDK for C++ umgeleitet. Die Protokollebene und das Protokollierungssystem, die Sie für das SDK for C++ angeben, gelten auch für das CRT.

Im vorherigen Beispiel erbt das CRT Nachrichten auf derselben `Info` Ebene `LogLevel::Info` und protokolliert sie auch in derselben Datei.

Sie können die Protokollierung für die CRT-Bibliotheken unabhängig steuern, indem Sie entweder die Ausgabe in eine separate Protokolldatei umleiten oder indem Sie eine andere Protokollebene für Nachrichten aus der CRT festlegen. Oft kann es von Vorteil sein, die Ausführlichkeit der CRT-Bibliotheken zu reduzieren, damit sie die Logs nicht überfordern. Beispielsweise kann die Protokollebene nur für die CRT-Ausgabe wie folgt festgelegt werden: `Warn`

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

Durch die optionale Verwendung der Methode `InitializeAWSLogging` können Sie den Ausführlichkeitsgrad und die Protokollausgabe von steuern. `DefaultLogSystem` Sie können das Präfix für den Protokolldateinamen konfigurieren oder die Ausgabe in einen Stream statt in eine Datei umleiten.

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

Anstatt die zu verwenden, können Sie auch diese Methode verwenden`DefaultLogSystem`, um Ihre eigene Protokollierungsimplementierung bereitzustellen.

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

Wenn Sie die Methode aufrufen`InitializeAWSLogging`, geben Sie am Ende Ihres Programms Ressourcen frei, indem Sie sie aufrufen`ShutdownAWSLogging`.

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

Beispiel für einen Integrationstest mit Protokollierung

```
#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;
}
```

Beispiel für eine Unterklasse von **`Aws::Utils::Logging::DefaultLogSystem`** für benutzerdefinierte Protokollierung

Der folgende Code zeigt, wie Sie eine Unterklasse für die `Aws::Utils::Logging::DefaultLogSystem` Klasse erstellen, die Teil von ist. AWS SDK für C++ In diesem Beispiel wird die `ProcessFormattedStatement` virtuelle Funktion überschrieben, um die Protokollierung anzupassen.

`Aws::Utils::Logging::DefaultLogSystem`ist eine von mehreren Klassen in der Unterklasse That `Aws::Utils::Logging::LogSystemInterface` für AWS SDK für C++ die benutzerdefinierte Protokollierung.

```
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;
}
```

Das vollständige Beispiel finden Sie unter. GitHub

Überschreiben Sie Ihren HTTP-Client in der AWS SDK für C++

Der Standard-HTTP-Client für Windows ist WinHTTP. Der Standard-HTTP-Client für alle anderen Plattformen ist curl.

Optional können Sie die Standardeinstellung des HTTP-Clients überschreiben, indem Sie einen benutzerdefinierten Code erstellen`HttpClientFactory`, der an den Konstruktor eines beliebigen Service-Clients übergeben wird. Um den HTTP-Client zu überschreiben, muss das SDK mit Curl-Unterstützung erstellt werden. Die Curl-Unterstützung ist standardmäßig in Linux und macOS integriert, aber für die Erstellung unter Windows sind zusätzliche Schritte erforderlich. Weitere Informationen zum Erstellen des SDK unter Windows mit Curl-Unterstützung finden Sie unter. Das unter AWS SDK für C++ Windows erstellen

Steuerung von Iostreams, die von **`HttpClient`** und **`AWSClient`** in der verwendet werden AWS SDK für C++

Standardmäßig verwenden alle Antworten einen Eingabestream, der von einem unterstützt wird`stringbuf`. Bei Bedarf können Sie das Standardverhalten überschreiben. Wenn Sie beispielsweise einen Amazon S3 verwenden `GetObject` und nicht die gesamte Datei in den Speicher laden möchten, können Sie `IOStreamFactory` in verwenden, um ein Lambda `AmazonWebServiceRequest` zu übergeben, um einen Dateistream zu erstellen.

Beispiel für eine Datei-Stream-Anfrage

```
 //! 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**
>
> Es gibt noch mehr dazu GitHub. Das vollständige Beispiel finden Sie im AWS Code Examples Repository.

Verwenden einer benutzerdefinierten libcrypto-Bibliothek in der AWS SDK für C++

Standardmäßig AWS SDK für C++ verwendet der die kryptografische Standardbibliothek des Systems für die Sicherheit auf Transportschicht. Das SDK for C++ kann jedoch optional so konfiguriert werden, dass es beim Erstellen des SDK aus dem Quellcode eine andere libcrypto-Bibliothek verwendet. Dies bedeutet funktionell, dass alle kryptografischen Operationen auf eine benutzerdefinierte Implementierung von OpenSSL umgeleitet werden. Möglicherweise möchten Sie die AWS-LCBibliothek beispielsweise im FIPS-Modus verwenden, um einen FIPS-Standard in Ihrer Anwendung zu erreichen.

Wie man ein benutzerdefiniertes Libcrypto in das SDK for C++ einbaut

Schritt 1: Erstellen oder beziehen Sie Ihre libcrypto-Bibliothek

Dies AWS-LCist ein Beispiel für eine alternative libcrypto-Bibliothek, aber jede Distribution von OpenSSL oder einem OpenSSL-Äquivalent würde funktionieren.

Das SDK for C++ und seine Abhängigkeit, die CRT, verwenden beide libcrypto für ihre kryptografischen Funktionen und beide müssen Abhängigkeiten auf die gleiche Weise handhaben. Das SDK for C++ hängt von zwei verschiedenen HTTP-Clients ab, je nachdem, ob die Anfrage die CRT S3 SDK-Funktionalität verwendet. Das CRT ist speziell von s2n abhängig, einer TLS-Implementierung, die beim Start initialisiert wird. Sowohl das SDK als auch das s2n-Team haben einen cmake-Parameter, um die Verwendung einer gemeinsam genutzten libcrypto-Bibliothek unabhängig vom Wert von zu erzwingen. BUILD_SHARED_LIBS Normalerweise möchten Sie, dass der CRT-HTTP-Client und der reguläre HTTP-Client dasselbe libcrypto verwenden. In diesem Fall würde das bedeuten, dass beide im Abhängigkeitsbaum auf OpenSSL verweisen. Das SDK stellt dies über bereit AWS_USE_CRYPTO_SHARED_LIBS und s2n (für CRT-basierte Aufrufe) stellt dies über bereit. S2N_USE_CRYPTO_SHARED_LIBS Die Auflösung der Abhängigkeiten zwischen diesen beiden Bibliotheken ist dieselbe, und in der Regel sind sie so eingestellt, dass sie übereinstimmen, obwohl Sie sie auch explizit als unterschiedlich festlegen können.

Um sie beispielsweise AWS-LC als libcrypto-Bibliothek zu verwenden, würden Sie sie wie folgt erstellen:

```
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 .
```

Schritt 2: Erstellen Sie Curl aus dem Quellcode oder verwenden Sie eine Curl-Distribution mit Ihrer libcrypto-Bibliothek

Das SDK for C++ erfordert, dass auf dem System ein HTTP-Client installiert ist, der für HTTP-Anfragen verwendet wird. Der HTTP-Client muss mit der Datei libcrypto erstellt werden, die Sie verwenden möchten. Der HTTP-Client ist für TLS-Operationen verantwortlich und verwendet daher Ihre libcrypto-Bibliothek.

Im folgenden Beispiel wird die curl-Bibliothek mit einer installierten Version von neu erstellt. `AWS-LC`

```
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
```

Schritt 3: Erstellen Sie das SDK mit den Bibliotheken libcrypto und curl

Das SDK for C++ kann jetzt mit den zuvor erstellten libcrypto- und curl-Artefakten erstellt werden. Dieser Build des SDK verwendet die benutzerdefinierte libcrypto-Bibliothek für alle kryptografischen Funktionen.

```
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 .
```

Alles in einem Docker-Image zusammenführen

Die folgende Docker-Beispieldatei zeigt, wie diese Schritte in der Amazon Linux 2023-Umgebung implementiert werden.

```
# 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 .
```

Verwenden des AWS SDK for C++

Dieser Abschnitt enthält Informationen zur allgemeinen Verwendung von AWS SDK für C++, die über das hinausgehen, was in Erste Schritte mit dem behandelt wird AWS SDK für C++.

Dienstspezifische Programmierbeispiele finden Sie unter AWS SDK für C++ Codebeispiele.

Themen

- Initialisieren und Herunterfahren des AWS SDK für C++
- AWS-Service Anfragen mit dem AWS SDK for C++ stellen
- Asynchrone Programmierung mit dem AWS SDK für C++
- Utility-Module verfügbar in AWS SDK für C++
- Speicherverwaltung in der AWS SDK für C++
- Behandlung von Fehlern im AWS SDK for C++

Initialisieren und Herunterfahren des AWS SDK für C++

Anwendungen, die das verwenden, AWS SDK für C++ müssen es initialisieren. Ebenso muss das SDK heruntergefahren werden, bevor die Anwendung beendet wird. Beide Operationen akzeptieren Konfigurationsoptionen, die sich auf die Initialisierungs- und Shutdown-Prozesse und nachfolgende Aufrufe des SDK auswirken.

Alle Anwendungen, die die verwenden, AWS SDK für C++ müssen die Datei `aws/core/Aws.h` enthalten.

Die AWS SDK für C++ muss durch einen Aufruf `Aws::InitAPI` initialisiert werden. Bevor die Anwendung beendet wird, muss das SDK durch einen Aufruf heruntergefahren werden. `Aws::ShutdownAPI` Jede Methode akzeptiert ein Argument von. `Aws::SDKOptions` Alle anderen Aufrufe des SDK können zwischen diesen beiden Methodenaufrufen ausgeführt werden.

Alle AWS SDK für C++ Aufrufe, die zwischen **`Aws::InitAPI`** und ausgeführt werden, **`Aws::ShutdownAPI`** sollten entweder in einem Paar geschweifter Klammern stehen oder durch Funktionen aufgerufen werden, die zwischen den beiden Methoden aufgerufen werden.

Eine grundlegende Grundapplikation ist unten dargestellt.

```
#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;
}
```

Das SDK for C++ und seine Abhängigkeiten verwenden statische C++-Objekte, und die Reihenfolge der Zerstörung statischer Objekte wird nicht durch den C++-Standard bestimmt. Um Speicherprobleme zu vermeiden, die durch die nichtdeterministische Reihenfolge der Zerstörung statischer Variablen verursacht werden, sollten Sie die Aufrufe von **`Aws::InitAPI`** und **`Aws::ShutdownAPI`** in kein anderes statisches Objekt einbinden.

AWS-Service Anfragen mit dem AWS SDK for C++ stellen

SDKs Verwenden Sie für den AWS-Services programmgesteuerten Zugriff jeweils eine Clientklasse. AWS-Service Wenn Ihre Anwendung beispielsweise auf Amazon zugreifen muss EC2, erstellt Ihre Anwendung ein EC2 Amazon-Client-Objekt als Schnittstelle zu diesem Service. Anschließend verwenden Sie den Service-Client, um Anfragen an dieses zu stellen AWS-Service.

Um eine Anfrage an einen zu stellen AWS-Service, müssen Sie zunächst einen Service-Client erstellen und konfigurieren. Für jeden Code, den AWS-Service Sie verwenden, gibt es eine eigene Bibliothek und einen eigenen Typ für die Interaktion mit ihm. Der Client stellt für jeden API-Vorgang, der vom Dienst verfügbar gemacht wird, eine Methode zur Verfügung.

Der Namespace für eine Clientklasse folgt der Konvention. `Aws::`*`Service`*`::`*`Service`*`Client` Zum Beispiel ist die Clientklasse für AWS Identity and Access Management (IAM) `Aws::IAM::IAMClient` und die Amazon S3 S3-Clientklasse ist`Aws::S3::S3Client`.

Alle Client-Klassen für alle AWS Services sind Thread-sicher.

Bei der Instanziierung einer Clientklasse müssen AWS Anmeldeinformationen angegeben werden. Anmeldeinformationen können aus Ihrem Code, der Umgebung oder der gemeinsam genutzten AWS `config` Datei und der gemeinsam genutzten Datei bereitgestellt werden`credentials`. Weitere Informationen zu Anmeldeinformationen finden Sie in den Anweisungen zur Einrichtung der

empfohlenen IAM Identity Center-Authentifizierung oder verwenden Sie einen anderen verfügbaren Anmeldeinformationsanbieter.

Asynchrone Programmierung mit dem AWS SDK für C++

Asynchrone SDK-Methoden

Für viele Methoden bietet das SDK for C++ sowohl synchrone als auch asynchrone Versionen. Eine Methode ist asynchron, wenn sie das `Async` Suffix in ihrem Namen enthält. Beispielsweise `PutObject` ist die Amazon S3 S3-Methode synchron, während sie asynchron `PutObjectAsync` ist.

Wie alle asynchronen Operationen kehrt eine asynchrone SDK-Methode zurück, bevor ihre Hauptaufgabe abgeschlossen ist. Die `PutObjectAsync` Methode kehrt beispielsweise zurück, bevor das Hochladen der Datei in den Amazon S3 S3-Bucket abgeschlossen ist. Während der Upload-Vorgang fortgesetzt wird, kann die Anwendung andere Operationen ausführen, einschließlich des Aufrufs anderer asynchroner Methoden. Die Anwendung wird benachrichtigt, dass ein asynchroner Vorgang abgeschlossen ist, wenn eine zugehörige Callback-Funktion aufgerufen wird.

In den folgenden Abschnitten wird ein Codebeispiel beschrieben, das den Aufruf der `PutObjectAsync` asynchronen Methode demonstriert. Jeder Abschnitt konzentriert sich auf einzelne Teile der gesamten Quelldatei des Beispiels.

Asynchrone SDK-Methoden werden aufgerufen

Im Allgemeinen akzeptiert die asynchrone Version einer SDK-Methode die folgenden Argumente.

- Ein Verweis auf dasselbe Objekt vom Typ Request wie sein synchrones Gegenstück.
- Ein Verweis auf eine Rückruffunktion für einen Antworthandler. Diese Callback-Funktion wird aufgerufen, wenn der asynchrone Vorgang abgeschlossen ist. Eines der Argumente enthält das Ergebnis der Operation.
- Eine Option `shared_ptr` für ein `AsyncCallerContext` Objekt. Das Objekt wird an den Callback des Antworthandlers übergeben. Es enthält eine UUID-Eigenschaft, die verwendet werden kann, um Textinformationen an den Callback zu übergeben.

Die unten gezeigte `uploadFileAsync` Methode richtet die Amazon S3 `PutObjectAsync` S3-Methode des SDK ein und ruft sie auf, um eine Datei asynchron in einen Amazon S3 S3-Bucket hochzuladen.

Die Funktion empfängt Verweise auf ein `S3Client` Objekt und ein `PutObjectRequest` Objekt. Sie empfängt sie von der Hauptfunktion, weil wir sicherstellen müssen, dass diese Objekte während der gesamten Dauer der asynchronen Aufrufe existieren.

`shared_ptr`Einem `AsyncCallerContext` Objekt wird A zugewiesen. Seine `UUID` Eigenschaft ist auf den Amazon S3 S3-Objektnamen gesetzt. Zu Demonstrationszwecken greift der Response-Handler-Callback auf die Eigenschaft zu und gibt ihren Wert aus.

Der Aufruf von `PutObjectAsync` beinhaltet ein Referenzargument für die Callback-Funktion des Antworthandlers. `uploadFileAsyncFinished` Diese Callback-Funktion wird im nächsten Abschnitt genauer untersucht.

```
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;
```

```
}
```

Die Ressourcen für einen asynchronen Vorgang müssen vorhanden sein, bis der Vorgang abgeschlossen ist. Beispielsweise müssen die Client- und Anforderungsobjekte vorhanden sein, bis die Anwendung eine Benachrichtigung erhält, dass der Vorgang abgeschlossen wurde. Die Anwendung selbst kann erst beendet werden, wenn der asynchrone Vorgang abgeschlossen ist.

Aus diesem Grund akzeptiert die `uploadFileAsync` Methode Verweise auf `PutObjectRequest` Objekte `S3Client` und Objekte, anstatt sie in der `uploadFileAsync` Methode zu erstellen und in einer lokalen Variablen zu speichern.

In diesem Beispiel kehrt die `PutObjectAsync` Methode unmittelbar nach Beginn des asynchronen Vorgangs zum Aufrufer zurück, sodass die aufrufende Kette zusätzliche Aufgaben ausführen kann, während der Upload-Vorgang läuft.

Wenn der Client in einer lokalen Variablen in der `uploadFileAsync` Methode gespeichert würde, würde er den Gültigkeitsbereich verlassen, wenn die Methode zurückgegeben wird. Das Client-Objekt muss jedoch weiterhin existieren, bis der asynchrone Vorgang abgeschlossen ist.

Benachrichtigung über den Abschluss eines asynchronen Vorgangs

Wenn ein asynchroner Vorgang abgeschlossen ist, wird eine Callback-Funktion für den Antworthandler der Anwendung aufgerufen. Diese Benachrichtigung enthält das Ergebnis des Vorgangs. Das Ergebnis ist in derselben Klasse vom Typ Outcome enthalten, die vom synchronen Gegenstück der Methode zurückgegeben wird. Im Codebeispiel befindet sich das Ergebnis in einem Objekt. `PutObjectOutcome`

Die Rückruffunktion des Beispiels für den Antworthandler `uploadFileAsyncFinished` ist unten dargestellt. Sie prüft, ob der asynchrone Vorgang erfolgreich war oder fehlgeschlagen ist. Es verwendet `astd::condition_variable`, um den Anwendungsthread darüber zu informieren, dass der asynchrone Vorgang abgeschlossen ist.

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

Wenn der asynchrone Vorgang abgeschlossen ist, können die damit verbundenen Ressourcen freigegeben werden. Die Anwendung kann auch beendet werden, wenn sie dies wünscht.

Der folgende Code zeigt, wie die `uploadFileAsyncFinished` Methoden `uploadFileAsync` und von einer Anwendung verwendet werden.

Die Anwendung weist die `PutObjectRequest` Objekte `S3Client` und so zu, dass sie so lange bestehen bleiben, bis der asynchrone Vorgang abgeschlossen ist. Nach dem Aufruf `uploadFileAsync` kann die Anwendung alle gewünschten Operationen ausführen. Der Einfachheit halber verwendet das Beispiel ein `std::mutex` und, `std::condition_variable` um zu warten, bis der Antworthandler-Callback das Programm darüber informiert, dass der Upload-Vorgang abgeschlossen ist.

```
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;
}
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Utility-Module verfügbar in AWS SDK für C++

Das AWS SDK für C++ beinhaltet viele Hilfsmodule, um die Komplexität der Entwicklung von AWS Anwendungen in C++ zu reduzieren.

HTTP-Stack

Ein HTTP-Stack, der Verbindungspooling ermöglicht, threadsicher ist und bei Bedarf wiederverwendet werden kann. Weitere Informationen finden Sie unter Client-Konfiguration.AWS

Überschriften	`/aws/core/http/`
API-Dokumentation	`Aws::Http`

Zeichenketten-Utils

Kernfunktionen von Zeichenketten wie `trimlowercase`, und numerische Konvertierungen.

Header	`aws/core/utils/StringUtils.h`
API-Dokumentation	`Aws::Utils::StringUtils`

Werkzeuge zum Hashing

Hashing-Funktionen wie`SHA256`,`MD5`, `Base64` und. `SHA256_HMAC`

Header	`/aws/core/utils/HashingUtils.h`

API-Dokumentation	`Aws::Utils::HashingUtils`

JSON-Parser

Ein voll funktionsfähiger und dennoch leichter JSON-Parser (ein dünner Wrapper). *cJSON*

Header	`/aws/core/utils/json/JsonSerializer.h`
API-Dokumentation	`Aws::Utils::Json::JsonValue`

XML-Parser

Ein leichter XML-Parser (ein dünner Wrapper). *tinyxml2* Das RAII-Muster wurde der Schnittstelle hinzugefügt.

Header	`/aws/core/utils/xml/XmlSerializer.h`
API-Dokumentation	`Aws::Utils::Xml`

Speicherverwaltung in der AWS SDK für C++

Das AWS SDK für C++ bietet eine Möglichkeit, die Speicherzuweisung und -freigabe in einer Bibliothek zu steuern.

> **Note**
>
> Benutzerdefinierte Speicherverwaltung ist nur verfügbar, wenn Sie eine Version der Bibliothek verwenden, die mit der definierten Kompilierzeitkonstante erstellt wurde. `USE_AWS_MEMORY_MANAGEMENT`
> Wenn Sie eine Version der Bibliothek verwenden, die ohne die Kompilierzeitkonstante erstellt wurde, funktionieren globale Speichersystemfunktionen wie z. B.

`InitializeAWSMemorySystem` nicht. Stattdessen werden die globalen `delete` Funktionen `new` und verwendet.

Weitere Hinweise zur Kompilierzeitkonstante finden Sie unter STL und Strings and Vectors. AWS

Speicher zuweisen und Zuweisung aufheben

Um Speicher zuzuweisen oder die Zuweisung aufzuheben

1. Unterklasse`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**
>
> Sie können die Typsignatur für nach `AllocateMemory` Bedarf ändern.

2. Verwenden Sie die `Aws::SDKOptions` Struktur, um die Verwendung des benutzerdefinierten Speichermanagers zu konfigurieren. Übergeben Sie die Instanz der Struktur an`Aws::InitAPI`. Bevor die Anwendung beendet wird, muss das SDK heruntergefahren werden, indem es `Aws::ShutdownAPI` mit derselben Instanz aufgerufen wird.

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

  // ... do stuff
```

```
  Aws::ShutdownAPI(options);

  return 0;
}
```

STL und AWS Zeichenketten und Vektoren

Bei der Initialisierung mit einem Speichermanager verschiebt der die AWS SDK für C++ gesamte Zuweisung und Freigabe an den Speichermanager. Wenn kein Speichermanager vorhanden ist, verwendet das SDK die Optionen Global New und Delete.

Wenn Sie benutzerdefinierte STL-Allokatoren verwenden, müssen Sie die Typsignaturen für alle STL-Objekte so ändern, dass sie der Zuweisungsrichtlinie entsprechen. Da STL in der SDK-Implementierung und -Schnittstelle eine wichtige Rolle spielt, würde ein einziger Ansatz im SDK die direkte Übergabe von Standard-STL-Objekten an das SDK oder die Steuerung der STL-Zuweisung verhindern. Alternativ könnte ein hybrider Ansatz, bei dem intern benutzerdefinierte Zuordnungen verwendet und standardmäßige und benutzerdefinierte STL-Objekte auf der Schnittstelle zugelassen werden, die Untersuchung von Speicherproblemen potenziell erschweren.

Die Lösung besteht darin, die Kompilierzeitkonstante des Speichersystems zu verwenden, um zu steuern, welche STL-Typen das SDK verwendet. `USE_AWS_MEMORY_MANAGEMENT`

Wenn die Kompilierzeitkonstante aktiviert (on) ist, werden die Typen in STL-Typen aufgelöst, wobei ein benutzerdefinierter Allocator mit dem Speichersystem verbunden ist. AWS

Wenn die Kompilierzeitkonstante deaktiviert (off) ist, werden alle `Aws::*` Typen in den entsprechenden Standardtyp aufgelöst. `std::*`

Beispielcode aus der **`AWSAllocator.h`** Datei im SDK

```
#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
```

Im Beispielcode `AwsAllocator` kann es sich je nach Kompilierzeitkonstante um einen benutzerdefinierten Allocator oder einen Standard-Allocator handeln.

Beispielcode aus der Datei im SDK **`AWSVector.h`**

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

Im Beispielcode definieren wir die `Aws::*` Typen.

Wenn die Kompilierzeitkonstante aktiviert (on) ist, wird der Typ mithilfe der benutzerdefinierten Speicherzuweisung und des AWS Speichersystems einem Vektor zugeordnet.

Wenn die Kompilierzeitkonstante deaktiviert (off) ist, wird der Typ einem regulären Typ `std::vector` mit Standard-Typparametern zugeordnet.

Typaliasing wird für alle `std::` Typen im SDK verwendet, die die Speicherzuweisung vornehmen, z. B. für Container, Zeichenfolgenstreams und Zeichenkettenpuffer. The AWS SDK für C++ verwendet diese Typen.

Verbleibende Probleme

Sie können die Speicherzuweisung im SDK steuern. STL-Typen dominieren jedoch weiterhin die öffentliche Schnittstelle über Zeichenkettenparameter für das Modellobjekt `initialize` und die `set` Methoden. Wenn Sie STL nicht verwenden und stattdessen Zeichenketten und Container verwenden, müssen Sie bei jedem Serviceaufruf eine Menge temporärer Dateien erstellen.

Um die meisten temporären Dateien und Zuweisungen zu entfernen, wenn Sie Serviceanfragen mit einem anderen Format als STL tätigen, haben wir Folgendes implementiert:

- Jede Init/Set Funktion, die eine Zeichenfolge akzeptiert, hat eine Überladung, die eine benötigt. `const char*`
- Jede Init/Set Funktion, die einen Container (Karte/Vektor) akzeptiert, hat eine Add-Variante, die einen einzigen Eintrag benötigt.
- Jede Init/Set Funktion, die Binärdaten akzeptiert, hat eine Überladung, die einen Zeiger auf die Daten und einen `length` Wert benötigt.

- (Optional) Jede Init/Set Funktion, die eine Zeichenfolge akzeptiert, hat eine Überladung, bei der ein Wert ungleich Null beendet `const char*` und ein `length` Wert angegeben wird.

Native SDK-Entwickler und Speichersteuerungen

Folgen Sie diesen Regeln im SDK-Code:

- Verwenden Sie nicht `new` und`delete`; verwenden Sie `Aws::Delete<>` stattdessen `Aws::New<>` und.
- Verwenden Sie nicht `new[]` und`delete[]`; verwenden Sie `Aws::NewArray<>` und`Aws::DeleteArray<>`.
- Benutze nicht`std::make_shared`; benutze`Aws::MakeShared`.
- Wird `Aws::UniquePtr` für eindeutige Zeiger auf ein einzelnes Objekt verwendet. Verwenden Sie die `Aws::MakeUnique` Funktion, um den eindeutigen Zeiger zu erstellen.
- Wird `Aws::UniqueArray` für eindeutige Zeiger auf eine Reihe von Objekten verwendet. Verwenden Sie die `Aws::MakeUniqueArray` Funktion, um den eindeutigen Zeiger zu erstellen.
- Verwenden Sie STL-Container nicht direkt. Verwenden Sie eine der `Aws::` Typedefs oder fügen Sie eine Typedef für den gewünschten Container hinzu. Beispiel:

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

- Verwenden Sie diese `shared_ptr` Option für alle externen Zeiger, die an das SDK übergeben und von diesem verwaltet werden. Sie müssen den gemeinsamen Zeiger mit einer Vernichtungsrichtlinie initialisieren, die der Art und Weise entspricht, wie das Objekt zugewiesen wurde. Sie können einen unbearbeiteten Zeiger verwenden, wenn nicht erwartet wird, dass das SDK den Zeiger bereinigt.

Behandlung von Fehlern im AWS SDK for C++

Das verwendet AWS SDK für C++ keine Ausnahmen. Sie können jedoch Ausnahmen in Ihrem Code verwenden. Jeder Service-Client gibt ein Ergebnisobjekt zurück, das das Ergebnis und einen Fehlercode enthält.

Beispiel für den Umgang mit Fehlerbedingungen

```
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;
```

```
}
```

Aufruf AWS-Services aus dem AWS SDK for C++

Die folgenden Abschnitte enthalten Beispiele, Tutorials, Aufgaben und Anleitungen, die Ihnen zeigen, wie Sie mit AWS Diensten arbeiten können. AWS SDK für C++

Wenn Sie mit dem noch nicht vertraut sind AWS SDK für C++, sollten Sie sich zuerst das Erste Schritte Thema durchlesen.

Weitere Codebeispiele finden Sie im C++-Beispielordner unter GitHub.

Weitere Codebeispiele finden Sie im Codebeispiele Kapitel dieses Handbuchs oder im AWS Codebeispiel-Repository unter GitHub.

Themen

- Erste Schritte mit den AWS SDK für C++ Codebeispielen
- Erste Schritte zur Behebung von Laufzeitfehlern in der AWS SDK für C++
- Geführte Beispiele für Aufrufe AWS-Services mit dem AWS SDK for C++

Erste Schritte mit den AWS SDK für C++ Codebeispielen

Struktur der Codebeispiele

Der C++-Beispielordner auf Github enthält Projektordner für jeden AWS Dienst. In der Regel demonstrieren einzelne .cpp-Quelldateien in den Ordnern eine bestimmte Funktionalität oder Aktion für diesen Dienst. Für Amazon DynamoDB sind das Abrufen eines Elements aus der Datenbank und das Hochladen eines Elements in die Datenbank beispielsweise zwei verschiedene Aktionstypen, sodass es für jede Aktion eine separate Datei im DynamoDB-Ordner gibt: und. `get_item.cpp` `put_item.cpp` Jede CPP-Datei enthält eine `main()` Funktion als Einstiegspunkt zu einer eigenständigen ausführbaren Datei. Die ausführbaren Projektdateien werden in einem Ordner generiert, der von Ihrem Build-System festgelegt wurde, und jeder Beispielquelldatei entspricht eine ausführbare Datei. Der Dateiname der ausführbaren Datei folgt den Konventionen der Plattform wie `{name}.exe` oder einfach, `{name}` und es `CMakeLists.txt` gilt jedes benutzerdefinierte Präfix wie. `run_`

Um eine Beispielfunktion auszuführen

1. Laden Sie das gewünschte Codebeispiel aus dem AWS Codebeispiel-Repository unter herunter GitHub.
2. Öffnen Sie eine.cpp-Datei, um ihre `main()` Funktion und alle aufgerufenen Methoden zu untersuchen.
3. Erstellen Sie das Projekt, wie anhand des Starterbeispiels unter Erste Schritte mit dem AWS SDK für C++ gezeigt wird. Beachten Sie, dass beim Erstellen des Projekts jede ausführbare Datei für jede Quelldatei im Projekt generiert wird.
4. Führen Sie die ausführbare Datei für die ausgewählte Funktionalität aus.
   - Führen Sie das Programm in einer Befehlszeile mit der ausführbaren Datei aus, die auf dem Namen der `*.cpp` Datei basiert.
   - Wenn Sie in einer IDE arbeiten, wählen Sie die `.cpp` Datei mit der Funktionalität aus, die Sie demonstrieren möchten, und wählen Sie sie als Startoption (oder Startobjekt) aus.

Komponententests

Tests für Beispiele werden mit dem GoogleTest Framework geschrieben. Weitere Informationen finden Sie unter GoogleTestPrimer auf der GoogleTest Website.

Die Komponententests für jedes Beispiel befinden sich in einem `tests` Unterordner, der eine eigene `CMakeLists.txt` Datei enthält. Für jede Beispielquelldatei gibt es eine entsprechende Testdatei mit dem Namen`test_<source file>`. Die ausführbare Testdatei für den Unterordner ist benannt`<AWS-Service>_gtests`.

CMakeDatei Lists.txt

Der Ordner für jeden Dienst enthält eine Datei mit dem Namen `CMakeLists.txt` file. Viele dieser Dateien enthalten ein Konstrukt, das dem folgenden ähnelt:

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

Für jede .cpp-Datei im Ordner erstellt die `CMakeLists.txt` Datei eine ausführbare Datei (cmake:`add_executable`) mit einem Namen, der auf dem Namen der Quellcodedatei ohne die Dateierweiterung basiert.

Codebeispiele in Visual Studio erstellen und debuggen

Erstellen und Ausführen des Amazon S3 S3-Codebeispiels

1. Besorgen Sie sich den Amazon S3 S3-Beispielquellcode. Dieses Verfahren verwendet das Amazon S3 S3-Codebeispiele mit dem AWS SDK für C++ Codebeispiel, um Visual Studio zum Laufen zu bringen.
2. Navigieren Sie im Windows Explorer zu dem `s3` Ordner (z. B.`\aws-doc-sdk-examples\cpp\example_code\s3`).
3. Klicken Sie mit der rechten Maustaste auf den `s3` Beispielordner und wählen Sie Mit Visual Studio öffnen. Visual Studio für CMake Projekte hat keine Projektdatei, sondern den gesamten Ordner.
4. Stellen Sie in der Dropdownliste zur Konfigurationsauswahl im oberen Menü von Visual Studio sicher, dass die ausgewählte Konfiguration dem Buildtyp entspricht, den Sie beim Erstellen des SDK aus dem Quellcode ausgewählt haben. Beispielsweise sollte eine Debug-Konfiguration ausgewählt werden, wenn Sie mithilfe von Debug (`-DCMAKE_BUILD_TYPE=Debug`in der CMake Befehlszeile der SDK-Installationsanweisungen) aus dem Quellcode erstellt haben.
5. Datei öffnen. `CMakeLists.txt`
6. Klicken Sie auf Speichern. Jedes Mal, wenn Sie in der `CMakeLists.txt` Datei auf Speichern klicken, aktualisiert Visual Studio die CMake generierten Dateien. Wenn Sie die Registerkarte Ausgabe angezeigt haben, können Sie die resultierenden Protokollmeldungen dieser Generation sehen.

   - Auf der Registerkarte Ausgabe befindet sich ein Dropdown-Feld mit der Aufschrift „Ausgabe anzeigen von:“. Diese Option CMakesollte standardmäßig ausgewählt sein.
   - Bei der letzten Nachrichtenausgabe sollte „CMake Generierung abgeschlossen“ stehen. “
   - Wenn die letzte Nachricht nicht das ist, dann hat die CMake Datei Probleme. Fahren Sie nicht mit weiteren Schritten fort, bis das Problem behoben ist. Siehe Behebung von Problemen mit AWS SDK for C++ C++-Build.
   - Beachten Sie, dass der CMake Cache aus CMake Geschwindigkeitsgründen verwendet wird. Wenn Sie CMake Probleme lösen, sollten Sie dafür sorgen, dass alles in Ordnung ist, sodass die Fehlermeldungen, die Sie erhalten, tatsächlich Ihre letzten Änderungen widerspiegeln. Klicken Sie im Projektmappen-Explorer mit der rechten Maustaste auf **`CMakeLists.txt`** und wählen Sie CMakeCache. Wählen Sie dann Cache löschen. Tun Sie dies häufig, wenn Sie CMake Probleme schrittweise lösen.

7. Um Beispiele in Visual Studio zu erstellen und auszuführen, platziert Visual Studio die ausführbaren Dateien in einer anderen Ordnerstruktur als in der Befehlszeile. Um den Code auszuführen, müssen die ausführbaren SDK-Dateien an die richtige Stelle kopiert werden. Suchen Sie die Zeile „TODO“ der CMake Lists-Datei (~Zeile 40) und wählen Sie die Zeile aus, die für die Verwendung in Visual Studio kommentiert wurde. Visual Studio verwendet keinen Unterordner, der dem Buildtyp zugewiesen ist, sodass dieser nicht enthalten ist. Tauschen Sie die auskommentierte Zeile in der `CMakeLists.txt` Datei für die Verwendung in Visual Studio aus.
8. Löschen Sie den CMake Cache (wie oben beschrieben), klicken Sie in der `CMakeLists.txt` Datei auf select/activate die Registerkarte und wählen Sie erneut Speichern für die `CMakeLists.txt` Datei, um die Generierung der CMake Build-Dateien zu starten.
9. Öffnen Sie die Quelldatei des „Programms“, das Sie ausführen möchten.
   - Öffnen Sie zum Beispiel`list_buckets.cpp`.
   - Der Amazon S3-Beispielordner ist so codiert, dass jede vorgestellte „Funktion“ von Amazon S3 in einer speziellen ausführbaren Datei für genau diese Funktion demonstriert wird. `list_buckets.cpp`Wird z. B. zu einer ausführbaren Datei, die nur die Auflistung von Buckets demonstriert.
10. Wählen Sie im oberen Menü Build und dann Build All.
    - Die Option „Ausgabe anzeigen von“ auf der Registerkarte „Ausgabe“ sollte die Auswahl von „Build“ widerspiegeln und alle Nachrichten zum Erstellen und Verknüpfen anzeigen.
    - Die letzte Ausgabe sollte lauten: „Build All successfully“. “
    - Jetzt werden ausführbare Dateien für jede der einzelnen Quelldateien generiert. Sie können dies überprüfen, indem Sie im Build-Ausgabeverzeichnis nachschauen (z. B.`\aws-doc-sdk-examples\cpp\example_code\s3\out\build\x64-Debug`).
    - Beachten Sie, dass den ausführbaren Dateien das Präfix „run_“ vorangestellt wird, da die `CMakeLists.txt` Datei dies vorschreibt.
11. Im oberen Menü befinden sich ein grüner Pfeil und eine Dropdownauswahl für Debug Target. Wählen Sie `run_list_buckets.exe`.
12. Klicken Sie auf die Startschaltfläche mit dem grünen Pfeil, um das Startelement auszuwählen.
13. Ein Fenster der Visual Studio Debug Console wird geöffnet und die Ausgabe des Codes wird angezeigt.

14. Drücken Sie eine Taste, um das Fenster zu schließen, oder schließen Sie das Fenster manuell, um das Programm zu beenden. Sie können im Code auch Haltepunkte setzen. Wenn Sie erneut auf Ausführen klicken, werden die Haltepunkte erreicht.

Erste Schritte zur Behebung von Laufzeitfehlern in der AWS SDK für C++

Wenn Sie lernen, Anwendungen mit dem zu entwickeln AWS SDK für C++, ist es auch nützlich, sich mit dem AWS-Managementkonsole und dem vertraut zu machen AWS CLI. Diese Tools können synonym für verschiedene Problembehebungen und Diagnosen verwendet werden, wenn Laufzeitfehler auftreten.

Das folgende Tutorial zeigt Ihnen ein Beispiel für diese Aufgaben zur Problembehandlung und Diagnose. Es konzentriert sich auf den `Access denied` Fehler, der aus verschiedenen Gründen auftreten kann. Das Tutorial zeigt ein Beispiel dafür, wie Sie die tatsächliche Ursache des Fehlers ermitteln können. Es konzentriert sich auf zwei mögliche Ursachen: falsche Berechtigungen für den aktuellen Benutzer und eine Ressource, die für den aktuellen Benutzer nicht verfügbar ist.

Um die Projektquelle und die ausführbaren Dateien abzurufen

1. Laden Sie den Amazon S3 S3-Codebeispielordner aus dem AWS Code Examples Repository unter herunter GitHub.
2. Öffnen Sie `delete_bucket.cpp` und stellen Sie fest, dass es zwei Methoden gibt: `main()` und`DeleteBucket()`. `DeleteBucket()`verwendet das SDK, um den Bucket zu löschen.
3. Erstellen Sie das Amazon S3 S3-Beispiel mit denselben Build-Schritten, die unter Erste Schritte mit dem beschrieben werden AWS SDK für C++. Der Build-Prozess generiert für jede Quelldatei eine ausführbare Datei.
4. Öffnen Sie eine Befehlszeile für den Ordner, in dem Ihr Build-System Ihre ausführbaren Build-Dateien generiert hat. Führen Sie die ausführbare Datei aus `run_create_bucket` (der tatsächliche Dateiname der ausführbaren Datei hängt von Ihrem Betriebssystem ab). Dadurch wird ein Bucket in Ihrem Konto erstellt (sodass Sie einen zum Löschen haben).
5. Führen Sie in der Befehlszeile die ausführbare Datei aus`run_delete_bucket`. In diesem Beispiel wird ein Parameter mit dem Namen des Buckets erwartet, den Sie löschen möchten. Geben Sie einen falschen Bucket-Namen ein. Geben Sie in diesem Bucket-Namen vorerst absichtlich einen Tippfehler ein, damit wir die Problembehandlung untersuchen können.

6. Vergewissern Sie sich, dass Sie eine `Access Denied` Fehlermeldung erhalten. Wenn Sie eine *`Access Denied`* Fehlermeldung erhalten, fragen Sie sich, ob Sie einen Benutzer mit vollen Berechtigungen für Amazon S3 erstellt haben, was Sie als Nächstes überprüfen werden.

Um den zu installieren AWS CLI und den Benutzernamen zu finden, der Aufrufe tätigt AWS

1. Informationen zur Installation der neuesten Version AWS CLI auf Ihrem Entwicklungscomputer finden Sie unter Installation von AWS CLI im AWS Command Line Interface Benutzerhandbuch.
2. Um zu überprüfen, ob AWS CLI das funktioniert, öffnen Sie eine Befehlszeile und führen Sie den Befehl aus `aws -\-version`

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

3. Führen Sie den AWS CLI Befehl aus, um den Benutzernamen zu erhalten AWS, für den die Aufrufe tatsächlich getätigt `aws sts get-caller-identity` werden. In der folgenden Beispielausgabe lautet dieser Benutzername UserX

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

Es gibt viele Möglichkeiten, Anmeldeinformationen anzugeben. Wenn Sie jedoch den oben beschriebenen Ansatz befolgt haben, stammt dieser Benutzername aus Ihrer AWS gemeinsamen Anmeldeinformationsdatei. Authentifizierung mit AWS AWS SDK for C++ Während dieses Vorgangs haben Sie Ihrem Benutzer AmazonS3-Berechtigungen FullAccess erteilt.

> **Note**
>
> Im Allgemeinen folgen die meisten AWS CLI Befehle der folgenden Syntaxstruktur:
>
> ```
> $ aws <command> <subcommand> [options and parameters]
> ```

wo *command* ist der Dienst und *subcommand* wird die Methode für diesen Dienst aufgerufen. Weitere Informationen finden Sie unter Befehlsstruktur AWS CLI im im AWS Command Line Interface Benutzerhandbuch.

Um zu überprüfen, ob ein Benutzer berechtigt ist, einen Bucket zu löschen

1. Öffnen Sie das AWS-Managementkonsoleund melden Sie sich an. Weitere Informationen finden Sie unter Erste Schritte mit dem AWS-Managementkonsole.
2. In der Hauptnavigationsleiste für Nach Diensten suchen... , geben Sie den IAM-Dienst ein **IAM** und wählen Sie ihn aus den Ergebnissen aus.
3. Wählen Sie in der Dashboard-Seitenleiste oder unter IAM-Ressourcen die Option Benutzer aus.
4. Wählen Sie aus der Tabelle der für Ihr Konto verfügbaren Benutzer den Benutzernamen aus, den Sie im vorherigen Verfahren erhalten haben.
5. Wählen Sie auf der Übersichtsseite die Registerkarte Berechtigungen und wählen Sie in der Tabelle mit dem Richtliniennamen die Option AmazonS3 aus. FullAccess
6. Sehen Sie sich die Zusammenfassung der Richtlinie und die JSON-Daten an. Stellen Sie sicher, dass dieser Benutzer über alle Rechte für den Amazon S3 S3-Service verfügt.

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

Dieser Eliminierungsprozess ist üblich, wenn ausgeschlossen werden soll, wo das Problem liegen könnte. In diesem Fall haben Sie überprüft, ob der Benutzer über die richtigen Berechtigungen verfügt. Das Problem muss also etwas anderes sein. Das heißt, da Sie über die richtigen Berechtigungen für den Zugriff auf Ihre Buckets verfügen, kann der `Access Denied` Fehler bedeuten, dass Sie versuchen, auf einen Bucket zuzugreifen, der nicht Ihnen gehört. Bei der Fehlerbehebung würden Sie als Nächstes den Bucket-Namen überprüfen, der dem Programm zur Verfügung gestellt wurde, und feststellen, dass ein Bucket mit diesem Namen in Ihrem Konto nicht vorhanden ist und Sie daher nicht darauf „zugreifen“ können.

Um das Codebeispiel so zu aktualisieren, dass es erfolgreich ausgeführt wird

1. Wenn Sie wieder in `delete_bucket.cpp` der `main()` Funktion sind, ändern Sie die Region mithilfe der Aufzählung in die Region Ihres Kontos. Um die Region Ihres Kontos zu finden,

melden Sie sich bei der AWS-Managementkonsole an und suchen Sie die Region in der oberen rechten Ecke. Ändern Sie außerdem den Bucket-Namen in **`main()`** einen Bucket, der in Ihrem Konto vorhanden ist. Es gibt mehrere Möglichkeiten, Ihre aktuellen Bucket-Namen zu finden:

- Sie können die `run_list_buckets` ausführbare Datei, die sich auch im Ordner dieses Codebeispiels befindet, verwenden, um die Namen Ihrer Buckets programmatisch abzurufen.
- Alternativ können Sie auch den folgenden AWS CLI Befehl verwenden, um Ihre Amazon S3 S3-Buckets aufzulisten.

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

- Alternativ können Sie auch den AWS-Managementkonsoleverwenden. In der Hauptnavigationsleiste unter Nach Diensten suchen... , geben Sie ein**S3**. Auf der Buckets-Seite werden die Buckets Ihres Kontos aufgeführt.

2. Erstellen Sie den Code neu und führen Sie die aktualisierte ausführbare Datei aus. `run_delete_bucket`
3. Stellen Sie mit dem AWS-Managementkonsole oder dem sicher AWS CLI, dass der Amazon S3 S3-Bucket, den Sie zuvor erstellt haben, gelöscht wurde.

Geführte Beispiele für Aufrufe AWS-Services mit dem AWS SDK for C++

Wenn Sie mit den AWS Codebeispielen AWS noch nicht vertraut sind, empfehlen wir Ihnen, mit zu beginnenErste Schritte mit Codebeispielen.

Quellcode, der zeigt, wie Sie mit AWS Diensten arbeiten, die AWS SDK für C++ das verwenden, finden Sie im Codebeispiele Kapitel dieses Handbuchs oder direkt im AWS Codebeispiel-Repository unter GitHub.

In diesem Abschnitt werden mehrere AWS Dienste ausgewählt und Sie werden durch die Beispiele geführt, in denen sie verwendet werden. Die folgenden Beispiele mit Anleitungen sind eine Teilmenge dessen, was auf Github verfügbar ist.

Servicebeispiele mit zusätzlichen Erläuterungen (eine vollständige Liste finden Sie im AWS Codebeispiel-Repository)

Service	Zusammenfassung dessen, was der Service für Ihr Programm bietet
Amazon CloudWatch	Erfasst und überwacht Messwerte für die AWS Ressourcen, die Sie verwenden
Amazon-DynamoDB	Ein NoSQL-Datenbankdienst
Amazon Elastic Compute Cloud (Amazon EC2)	Sichere, anpassbare Rechenkapazität
Amazon Simple Storage Service (Amazon-S3)	Speichern und Abrufen von Daten (Objekte in Buckets)
Amazon-Simple-Queue-Service (Amazon SQS)	Message Queuing-Dienst zum Senden, Speichern und Empfangen von Nachrichten zwischen Softwarekomponenten

Es gibt auch Beispiele, die zeigen, wie asynchrone Methoden verwendet werden.

Wie Sie dem AWS Dokumentationsteam ein neues Codebeispiel vorschlagen können, finden Sie unter Richtlinien für Beiträge GitHub zum Erstellen einer neuen Anfrage. Das Team zieht es vor, Codebeispiele zu erstellen, die allgemeine Szenarien zeigen, anstatt einzelne API-Aufrufe.

Verwenden der Codebeispiele unter Windows

Wenn Sie die Beispiele unter Windows mit SDK-Version 1.9 erstellen, finden Sie weitere Informationen unterBehebung von Problemen mit AWS SDK for C++ C++-Build.

CloudWatch Amazon-Beispiele mit dem AWS SDK für C++

Amazon CloudWatch (CloudWatch) ist ein Überwachungsdienst für AWS Cloud-Ressourcen und die Anwendungen, auf denen Sie laufen AWS. Sie können die folgenden Beispiele verwenden, um CloudWatchmit dem zu programmieren AWS SDK für C++.

Amazon CloudWatch überwacht Ihre AWS Ressourcen und die Anwendungen, auf denen Sie laufen, AWS in Echtzeit. Sie können CloudWatch damit Metriken sammeln und verfolgen. Dabei handelt es sich um Variablen, die Sie für Ihre Ressourcen und Anwendungen messen können.

CloudWatchAlarme senden Benachrichtigungen oder nehmen auf der Grundlage von von Ihnen festgelegter Regeln automatisch Änderungen an den Ressourcen vor, die Sie überwachen.

Weitere Informationen zu CloudWatch finden Sie im CloudWatch Amazon-Benutzerhandbuch.

> **Note**
>
> In diesem Handbuch ist nur der Code enthalten, der zur Demonstration bestimmter Techniken erforderlich ist. Der vollständige Beispielcode ist jedoch unter verfügbar GitHub. Auf können GitHub Sie eine einzelne Quelldatei herunterladen oder das Repository lokal klonen, um alle Beispiele abzurufen, zu erstellen und auszuführen.

Themen

- Metriken abrufen von CloudWatch
- Veröffentlichen benutzerdefinierter Metrikdaten
- Mit CloudWatch Alarmen arbeiten
- Verwenden von Alarmaktionen in CloudWatch
- Ereignisse senden an CloudWatch

Metriken abrufen von CloudWatch

Voraussetzungen

Bevor Sie beginnen, empfehlen wir Ihnen, Erste Schritte mit dem zu lesen AWS SDK für C++.

Laden Sie den Beispielcode herunter und erstellen Sie die Lösung wie unter beschriebenErste Schritte mit Codebeispielen.

Um die Beispiele ausführen zu können, muss das Benutzerprofil, das Ihr Code für die Anfragen verwendet, über die entsprechenden Berechtigungen verfügen AWS (für den Dienst und die Aktion). Weitere Informationen finden Sie unter Bereitstellen von AWS Anmeldeinformationen.

Auflisten von Metriken

Um CloudWatch Metriken aufzulisten, erstellen Sie eine `ListMetrics` Funktion ListMetricsRequestund rufen sie auf. CloudWatchClient Sie können `ListMetricsRequest` zum Filtern der zurückgegebenen Metriken nach Namespace, Metrikname oder Dimensionen verwenden.

> **Note**
>
> Eine Liste der Metriken und Dimensionen, die von AWS Services veröffentlicht werden, finden Sie in der Amazon CloudWatch Metrics and Dimensions Reference im CloudWatch Amazon-Benutzerhandbuch.

Beinhaltet

```
#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>
```

Code

```
        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();
        }
```

Die Metriken werden in a zurückgegeben, ListMetricsResultindem die zugehörige `GetMetrics` Funktion aufgerufen wird. Eventuell werden die Ergebnisse seitenweise zurückgegeben. Um den nächsten Ergebnisstapel abzurufen, rufen Sie das ursprüngliche Anforderungsobjekt mit dem `ListMetricsResult` Rückgabewert der `GetNextToken` Objektfunktion `SetNextToken` auf und übergeben das geänderte Anforderungsobjekt an einen anderen Aufruf von`ListMetrics`.

Siehe vollständiges Beispiel.

Weitere Informationen

- ListMetricsin der Amazon CloudWatch API-Referenz.

Veröffentlichen benutzerdefinierter Metrikdaten

Eine Reihe von AWS Diensten veröffentlichen ihre eigenen Metriken in Namespaces, die mit beginnen. `AWS/` Sie können benutzerdefinierte Metrikdaten auch in Ihrem eigenen Namespace veröffentlichen (sofern dieser nicht mit beginnt). `AWS/`

Voraussetzungen

Bevor Sie beginnen, empfehlen wir Ihnen, Erste Schritte mit dem zu lesen. AWS SDK für C++

Laden Sie den Beispielcode herunter und erstellen Sie die Lösung wie unter beschriebenErste Schritte mit Codebeispielen.

Um die Beispiele ausführen zu können, muss das Benutzerprofil, das Ihr Code für die Anfragen verwendet, über die entsprechenden Berechtigungen verfügen AWS (für den Dienst und die Aktion). Weitere Informationen finden Sie unter Bereitstellen von AWS Anmeldeinformationen.

Veröffentlichen benutzerdefinierter Metrikdaten

Um Ihre eigenen Metrikdaten zu veröffentlichen, rufen Sie die `PutMetricData` Funktion CloudWatchClient's mit einem auf PutMetricDataRequest. Die `PutMetricDataRequest` muss den benutzerdefinierten Namespace enthalten, der für die Daten verwendet werden soll, und Informationen über den Datenpunkt selbst in einem MetricDatumObjekt.

> **Note**
>
> Sie können keinen Namespace angeben, der mit beginnt. `AWS/` Namespaces, die mit beginnen, `AWS/` sind für die Verwendung durch Amazon Web Services Services-Produkte reserviert.

Beinhaltet

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

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

Code

```
        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;
        }
```

Siehe vollständiges Beispiel.

Weitere Informationen

- Verwenden von Amazon CloudWatch Metrics im CloudWatch Amazon-Benutzerhandbuch.
- AWS Namespaces im CloudWatch Amazon-Benutzerhandbuch.
- PutMetricDatain der Amazon CloudWatch API-Referenz.

Mit CloudWatch Alarmen arbeiten

Voraussetzungen

Bevor Sie beginnen, empfehlen wir Ihnen, Erste Schritte mit dem zu lesen AWS SDK für C++.

Laden Sie den Beispielcode herunter und erstellen Sie die Lösung wie unter beschriebenErste Schritte mit Codebeispielen.

Um die Beispiele ausführen zu können, muss das Benutzerprofil, das Ihr Code für die Anfragen verwendet, über die entsprechenden Berechtigungen verfügen AWS (für den Dienst und die Aktion). Weitere Informationen finden Sie unter Bereitstellen von AWS Anmeldeinformationen.

Einrichten eines Alarms

Um einen Alarm auf der Grundlage einer CloudWatch Metrik zu erstellen, rufen Sie die `PutMetricAlarm` Funktion CloudWatchClient s mit PutMetricAlarmRequesteiner Angabe der Alarmbedingungen auf.

Beinhaltet

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

Code

```
        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;
        }
```

Siehe vollständiges Beispiel.

Auflisten von Alarmen

Um die CloudWatch Alarme aufzulisten, die Sie erstellt haben, rufen Sie die `DescribeAlarms` Funktion CloudWatchClient s mit einer auf DescribeAlarmsRequest, mit der Sie Optionen für das Ergebnis festlegen können.

Beinhaltet

```
#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>
```

Code

```
        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();
        }
```

Die Liste der Alarme kann abgerufen werden, indem Sie `getMetricAlarms` den Befehl aufrufen DescribeAlarmsResult, der von zurückgegeben wird`DescribeAlarms`.

Eventuell werden die Ergebnisse seitenweise zurückgegeben. Um den nächsten Ergebnisstapel abzurufen, rufen Sie das ursprüngliche Anforderungsobjekt mit dem `DescribeAlarmsResult` Rückgabewert der `GetNextToken` Objektfunktion `SetNextToken` auf und übergeben das geänderte Anforderungsobjekt an einen anderen Aufruf von`DescribeAlarms`.

> **Note**
>
> Sie können auch Alarme für eine bestimmte Metrik abrufen, indem Sie die `DescribeAlarmsForMetric` Funktion CloudWatchClient's verwenden. Sie lässt sich ähnlich wie `DescribeAlarms` nutzen.

Siehe vollständiges Beispiel.

Löschen von Alarmen

Um CloudWatch Alarme zu löschen, rufen Sie die `DeleteAlarms` Funktion CloudWatchClient s auf, DeleteAlarmsRequestdie einen oder mehrere Namen von Alarmen enthält, die Sie löschen möchten.

Beinhaltet

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

Code

```
        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;
        }
```

Siehe vollständiges Beispiel.

Weitere Informationen

- CloudWatch Amazon-Alarme im CloudWatch Amazon-Benutzerhandbuch erstellen
- PutMetricAlarmin der Amazon CloudWatch API-Referenz
- DescribeAlarmsin der Amazon CloudWatch API-Referenz
- DeleteAlarmsin der Amazon CloudWatch API-Referenz

Verwenden von Alarmaktionen in CloudWatch

Mithilfe von CloudWatch Alarmaktionen können Sie Alarme erstellen, die Aktionen wie automatisches Stoppen, Beenden, Neustarten oder Wiederherstellen von Amazon EC2 EC2-Instances ausführen.

Alarmaktionen können einem Alarm hinzugefügt werden, indem Sie bei der Erstellung eines Alarms PutMetricAlarmRequestdie `SetAlarmActions` Funktion verwenden.

Voraussetzungen

Bevor Sie beginnen, empfehlen wir Ihnen, Erste Schritte mit dem zu lesen AWS SDK für C++.

Laden Sie den Beispielcode herunter und erstellen Sie die Lösung wie unter beschriebenErste Schritte mit Codebeispielen.

Um die Beispiele ausführen zu können, muss das Benutzerprofil, das Ihr Code für die Anfragen verwendet, über die entsprechenden Berechtigungen verfügen AWS (für den Dienst und die Aktion). Weitere Informationen finden Sie unter Bereitstellen von AWS Anmeldeinformationen.

Aktivieren von Alarmaktionen

Um Alarmaktionen für einen CloudWatch Alarm zu aktivieren, rufen Sie die CloudWatchClient s `EnableAlarmActions` mit einem auf, das einen oder mehrere Namen von Alarmen EnableAlarmActionsRequestenthält, deren Aktionen Sie aktivieren möchten.

Beinhaltet

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

Code

```
    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;
```

Siehe vollständiges Beispiel.

Deaktivieren von Alarmaktionen

Um Alarmaktionen für einen CloudWatch Alarm zu deaktivieren, rufen Sie die CloudWatchClient s `DisableAlarmActions` mit einem auf, der einen oder mehrere Namen von Alarmen DisableAlarmActionsRequestenthält, deren Aktionen Sie deaktivieren möchten.

Beinhaltet

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

Code

```
        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;
```

```
        }
```

Siehe vollständiges Beispiel.

Weitere Informationen

- Alarme zum Stoppen, Beenden, Neustarten oder Wiederherstellen einer Instance im CloudWatch Amazon-Benutzerhandbuch erstellen
- PutMetricAlarmin der Amazon CloudWatch API-Referenz
- EnableAlarmActionsin der Amazon CloudWatch API-Referenz
- DisableAlarmActionsin der Amazon CloudWatch API-Referenz

Ereignisse senden an CloudWatch

CloudWatch Events liefert nahezu in Echtzeit einen Stream von Systemereignissen, die AWS Ressourcenänderungen an Amazon EC2 EC2-Instances, Lambda-Funktionen, Kinesis-Streams, Amazon ECS-Aufgaben, Step Functions Functions-Zustandsmaschinen, Amazon SNS SNS-Themen, Amazon SQS SQS-Warteschlangen oder integrierten Zielen beschreiben. Sie können Ereignisse zuordnen und sie zu einer oder mehreren Zielfunktionen oder Streams umleiten, indem Sie einfache Regeln nutzen.

> **Note**
>
> Bei diesen Codefragmenten wird davon ausgegangen, dass Sie das Material unter Erste Schritte mit dem verstehen AWS SDK für C++ und AWS Standardanmeldedaten anhand der Informationen unter Anmeldeinformationen bereitstellen konfiguriert haben. AWS

Hinzufügen von Ereignissen

Um benutzerdefinierte CloudWatch Ereignisse hinzuzufügen, rufen Sie die `PutEvents` Funktion CloudWatchEventsClient's mit einem PutEventsRequestObjekt auf, das ein oder mehrere PutEventsRequestEntryObjekte enthält, die Details zu jedem Ereignis bereitstellen. Sie können mehrere Parameter für den Eintrag angeben, wie z. B. die Quelle und den Typ des Ereignisses, mit dem Ereignis verknüpfte Ressourcen usw.

> **Note**
>
> Sie können maximal 10 Ereignisse pro Aufruf von `putEvents` angeben.

Beinhaltet

```
#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>
```

Code

```
        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;
        }
```

Hinzufügen von Regeln

Um eine Regel zu erstellen oder zu aktualisieren, rufen Sie die `PutRule` Funktion CloudWatchEventsClient s PutRuleRequestmit einem Namen der Regel und optionalen Parametern

wie dem Ereignismuster, der der Regel zuzuordnenden IAM-Rolle und einem Scheduling-Ausdruck auf, der beschreibt, wie oft die Regel ausgeführt wird.

Beinhaltet

```
#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>
```

Code

```
        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;
        }
```

Hinzufügen von Zielen

Ziele sind die Ressourcen, die beim Auslösen einer Regel aufgerufen werden. Zu den Beispielzielen gehören Amazon EC2 EC2-Instances, Lambda-Funktionen, Kinesis-Streams, Amazon ECS-Aufgaben, Step Functions Functions-Zustandsmaschinen und integrierte Ziele.

Um einer Regel ein Ziel hinzuzufügen, rufen Sie die `PutTargets` 's-Funktion mit einer auf, die CloudWatchEventsClient die zu aktualisierende Regel PutTargetsRequestenthält, und einer Liste von Zielen, die der Regel hinzugefügt werden sollen.

Beinhaltet

```
#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>
```

Code

```
        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;
        }
```

Siehe vollständiges Beispiel.

Weitere Informationen

- Hinzufügen von Ereignissen mit PutEvents im Amazon CloudWatch Events-Benutzerhandbuch
- Ausdrücke für Regeln planen im Amazon CloudWatch Events-Benutzerhandbuch
- Ereignistypen für CloudWatch Ereignisse im Amazon CloudWatch Events-Benutzerhandbuch
- Ereignisse und Ereignismuster im Amazon CloudWatch Events-Benutzerhandbuch
- PutEventsin der Amazon CloudWatch Events API-Referenz
- PutTargetsin der Amazon CloudWatch Events API-Referenz
- PutRulein der Amazon CloudWatch Events API-Referenz

Amazon DynamoDB DynamoDB-Beispiele mit dem AWS SDK für C++

Amazon DynamoDB ist ein vollständig verwalteter NoSQL-Datenbank-Service, der schnelle und planbare Leistung mit nahtloser Skalierbarkeit bereitstellt. Die folgenden Beispiele zeigen, wie Sie Amazon DynamoDB mit dem programmieren können. AWS SDK für C++

> **Note**
>
> In diesem Handbuch ist nur der Code enthalten, der zur Veranschaulichung bestimmter Techniken erforderlich ist. Der vollständige Beispielcode ist jedoch unter verfügbar. GitHub Auf können GitHub Sie eine einzelne Quelldatei herunterladen oder das Repository lokal klonen, um alle Beispiele abzurufen, zu erstellen und auszuführen.

Themen

- Arbeiten mit Tabellen in DynamoDB
- Arbeiten mit Elementen in DynamoDB

Arbeiten mit Tabellen in DynamoDB

Tabellen sind die Container für alle Elemente in einer DynamoDB-Datenbank. Bevor Sie Daten aus DynamoDB hinzufügen oder daraus entfernen können, müssen Sie eine Tabelle erstellen.

Für jede Tabelle definieren Sie:

- Ein Tabellenname, der für Ihr AWS-Konto und eindeutig ist. AWS-Region

- Ein Primärschlüssel, für den jeder Wert eindeutig sein muss. Keine zwei Elemente in Ihrer Tabelle dürfen denselben Primärschlüsselwert haben.

  Ein Primärschlüssel kann einfach sein, also aus einem Schlüssel mit einer einzigen Partition (HASH) bestehen, oder zusammengesetzt, also aus einer Partition und einem Sortierschlüssel (RANGE).

  Jedem Schlüsselwert ist ein Datentyp zugeordnet, der nach der ScalarAttributeTypeKlasse aufgezählt wird. Der Schlüsselwert kann binär (B), numerisch (n) oder eine Zeichenfolge (S) sein. Weitere Informationen finden Sie unter Benennungsregeln und Datentypen im Amazon DynamoDB Developer Guide.
- Bereitgestellte Durchsatzwerte, die die Anzahl der reservierten read/write Kapazitätseinheiten für die Tabelle definieren.

> **Note**
>
> Amazon DynamoDB-Preise basieren auf dem bereitgestellten Durchsatz von Tabellen. Reservieren Sie also nur so viel Kapazität, wie Sie Ihrer Meinung nach je für die Tabelle brauchen werden.
> Der bereitgestellte Durchsatz für eine Tabelle kann jederzeit geändert werden. So können Sie die Kapazität anpassen, wenn sich Ihre Anforderungen ändern.

Erstellen einer Tabelle

Verwenden Sie die `CreateTable`DynamoDB-Clientmethode, um eine neue DynamoDB-Tabelle zu erstellen. Sie müssen Tabellenattribute und ein Tabellenschema erstellen. Beide Komponenten fließen in den Primärschlüssel der Tabelle ein. Sie müssen auch die anfänglichen bereitgestellten Durchsatzwerte und einen Tabellennamen angeben. `CreateTable`ist ein asynchroner Vorgang. `GetTableStatus`gibt CREATING zurück, bis die Tabelle AKTIV und einsatzbereit ist.

Erstellen einer Tabelle mit einem einfachen Primärschlüssel

Dieser Code erstellt eine Tabelle mit einem einfachen Primärschlüssel („Name“).

Beinhaltet

```
#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.
/*!
  \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);
}
```

Siehe vollständiges Beispiel.

Erstellen einer Tabelle mit einem zusammengesetzten Primärschlüssel

Füge ein weiteres hinzu AttributeDefinitionund KeySchemaElementzu CreateTableRequest.

Beinhaltet

```
#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);
}
```

Das vollständige Beispiel finden Sie unter GitHub.

Auflisten von Tabellen

Sie können die Tabellen in einer bestimmten Region auflisten, indem Sie die `ListTables`DynamoDB-Clientmethode aufrufen.

Beinhaltet

```
#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>
```

Code

```
//! 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;
}
```

Standardmäßig werden bis zu 100 Tabellen pro Aufruf zurückgegeben. Wird für das zurückgegebene ListTablesOutcomeObjekt verwendet`GetExclusiveStartTableName`, um die letzte Tabelle abzurufen, die ausgewertet wurde. Mit diesem Wert können Sie die Auflistung nach dem zuletzt zurückgegebenen Wert der vorherigen Auflistung beginnen.

Siehe vollständiges Beispiel.

Informationen zu einer Tabelle abrufen

Sie können mehr über eine Tabelle erfahren, indem Sie die `DescribeTable`DynamoDB-Clientmethode aufrufen.

Beinhaltet

```
#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();
}
```

Das vollständige Beispiel finden Sie unter GitHub.

Ändern Sie eine Tabelle

Sie können die bereitgestellten Durchsatzwerte Ihrer Tabelle jederzeit ändern, indem Sie die DynamoDB-Clientmethode aufrufen. `UpdateTable`

Beinhaltet

```
#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>
```

Code

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

Siehe vollständiges Beispiel.

Löschen einer Tabelle

Rufen Sie die `DeleteTable`DynamoDB-Clientmethode auf und übergeben Sie ihr den Namen der Tabelle.

Beinhaltet

```
#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();
}
```

Das vollständige Beispiel finden Sie unter GitHub.

Weitere Infos

- Richtlinien für die Arbeit mit Tabellen im Amazon DynamoDB DynamoDB-Entwicklerhandbuch
- Arbeiten mit Tabellen in DynamoDB im Amazon DynamoDB Developer Guide

Arbeiten mit Elementen in DynamoDB

In DynamoDB ist ein Element eine Sammlung von Attributen, von denen jedes einen Namen und einen Wert hat. Ein Attributwert kann eine Skalarfunktion, eine Gruppe oder ein Dokumenttyp sein. Weitere Informationen finden Sie unter Benennungsregeln und Datentypen im Amazon DynamoDB Developer Guide.

Rufen Sie ein Element aus einer Tabelle ab

Rufen Sie die `GetItem`DynamoDB-Clientmethode auf. Übergeben Sie ihr ein GetItemRequestObjekt mit dem Tabellennamen und dem Primärschlüsselwert des gewünschten Elements. Es gibt ein GetItemResultObjekt zurück.

Sie können die `GetItem()` Methode des zurückgegebenen `GetItemResult` Objekts verwenden, um eine Anzahl `Aws::Map` von Schlüssel `Aws::String` - und AttributeValueWertepaaren abzurufen, die dem Element zugeordnet sind.

Beinhaltet

```
#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();
}
```

Das vollständige Beispiel finden Sie unter GitHub.

Fügen Sie ein Element zu einer Tabelle hinzu

Erstellen Sie Schlüssel `Aws::String` - und AttributeValueWertepaare, die jedes Element repräsentieren. Diese müssen Werte für die Primärschlüsselfelder der Tabelle enthalten. Wenn das Element mit dem Primärschlüssel bereits vorhanden ist, werden dessen Felder durch die Anforderung aktualisiert. Fügen Sie sie PutItemRequestmithilfe der `AddItem` Methode hinzu.

Beinhaltet

```
#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);
}
```

Das vollständige Beispiel finden Sie unter GitHub.

Aktualisieren eines vorhandenen Elements in einer Tabelle

Sie können ein Attribut für ein Element aktualisieren, das bereits in einer Tabelle vorhanden DBClient ist, indem Sie die `UpdateItem` Dynamo-Methode verwenden und dabei einen Tabellennamen, einen Primärschlüsselwert und zu aktualisierende Felder sowie den entsprechenden Wert angeben.

Importe

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

Code

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

Siehe vollständiges Beispiel.

Weitere Infos

- Richtlinien für die Arbeit mit Elementen im Amazon DynamoDB DynamoDB-Entwicklerhandbuch
- Arbeiten mit Elementen in DynamoDB im Amazon DynamoDB Developer Guide

Amazon EC2 EC2-Beispiele mit dem AWS SDK für C++

Amazon Elastic Compute Cloud (Amazon EC2) ist ein Webservice, der skalierbare Rechenkapazität — buchstäblich Server in den Rechenzentren von Amazon — bereitstellt, die Sie zum Erstellen und Hosten Ihrer Softwaresysteme verwenden. Sie können die folgenden Beispiele verwenden, um Amazon EC2 mit dem AWS SDK für C++ zu programmieren.

> **Note**
>
> In diesem Handbuch ist nur der Code enthalten, der zur Demonstration bestimmter Techniken erforderlich ist. Der vollständige Beispielcode ist jedoch unter GitHub verfügbar. Auf können GitHub Sie eine einzelne Quelldatei herunterladen oder das Repository lokal klonen, um alle Beispiele abzurufen, zu erstellen und auszuführen.

Themen

- Verwalten von Amazon EC2 Instances
- Verwenden von Elastic IP-Adressen in Amazon EC2
- Verwenden von Regionen und Availability Zones für Amazon EC2

- Arbeiten mit Amazon EC2-Schlüsselpaaren
- Arbeiten mit Sicherheitsgruppen in Amazon EC2

Verwalten von Amazon EC2 Instances

Voraussetzungen

Bevor Sie beginnen, empfehlen wir Ihnen, Erste Schritte mit dem AWS SDK für C++ zu lesen.

Laden Sie den Beispielcode herunter und erstellen Sie die Lösung wie unter beschriebenErste Schritte mit Codebeispielen.

Um die Beispiele ausführen zu können, muss das Benutzerprofil, das Ihr Code für die Anfragen verwendet, über die entsprechenden Berechtigungen verfügen AWS (für den Dienst und die Aktion). Weitere Informationen finden Sie unter Bereitstellen von AWS Anmeldeinformationen.

Erstellen einer -Instance

Erstellen Sie eine neue Amazon EC2 EC2-Instance, indem Sie die `RunInstances` Funktion des EC2 Clients aufrufen und ihr ein zu RunInstancesRequestverwendendes Amazon Machine Image (AMI) und einen Instance-Typ zur Verfügung stellen.

Beinhaltet

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

Code

```
    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;
    }
```

Siehe vollständiges Beispiel.

Starten Sie eine Instanz

Um eine Amazon EC2 EC2-Instance zu starten, rufen Sie die `StartInstances` Funktion des EC2 Clients auf und geben ihr eine, die die ID der zu startenden Instance StartInstancesRequestenthält.

Beinhaltet

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

Code

```
    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;
    }
```

Siehe vollständiges Beispiel.

Stoppen Sie eine Instanz

Um eine Amazon EC2 EC2-Instance zu stoppen, rufen Sie die `StopInstances` Funktion des EC2 Clients auf und geben ihr eine, die die ID der zu stoppenden Instance StopInstancesRequestenthält.

Beinhaltet

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

Code

```
    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;
    }
```

Siehe vollständiges Beispiel.

Neustarten einer Instance

Um eine Amazon EC2 EC2-Instance neu zu starten, rufen Sie die `RebootInstances` Funktion des EC2 Clients auf und geben ihr eine, die die ID der neu zu startenden Instance RebootInstancesRequestenthält.

Beinhaltet

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

Code

```
    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;
   }
```

Siehe vollständiges Beispiel.

Instances beschreiben

Um Ihre Instanzen aufzulisten, erstellen Sie eine EC2 `DescribeInstances` Client-Funktion DescribeInstancesRequestund rufen Sie sie auf. Es wird ein DescribeInstancesResponseObjekt zurückgegeben, mit dem Sie die Amazon EC2 EC2-Instances für Ihr AWS-Konto und AWS-Region auflisten können.

Instances werden nach Reservierung gruppiert. Jede Reservierung entspricht dem Aufruf von `StartInstances`, durch den die Instance gestartet wurde. Um Ihre Instances aufzulisten, müssen Sie zuerst die `GetReservations` Funktion der `DescribeInstancesResponse` Klasse aufrufen und dann jedes zurückgegebene `getInstances` Reservierungsobjekt aufrufen.

Beinhaltet

```
#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;
        }
    }
```

Die Ergebnisse werden seitenweise angezeigt. Weitere Ergebnisse erhalten Sie, indem Sie den von der Funktion des Ergebnisobjekts zurückgegebenen Wert an die `GetNextToken` Funktion Ihres ursprünglichen Anforderungsobjekts `SetNextToken` übergeben und dann dasselbe Anforderungsobjekt beim nächsten Aufruf von verwenden. `DescribeInstances`

Siehe vollständiges Beispiel.

Aktivieren Sie die Instanzüberwachung

Sie können verschiedene Aspekte Ihrer Amazon EC2 EC2-Instances überwachen, z. B. die CPU- und Netzwerkauslastung, den verfügbaren Arbeitsspeicher und den verbleibenden Festplattenspeicher. Weitere Informationen zur Instance-Überwachung finden Sie unter Monitoring Amazon EC2 im Amazon EC2 EC2-Benutzerhandbuch.

Um mit der Überwachung einer Instance zu beginnen, müssen Sie eine MonitorInstancesRequestmit der ID der zu überwachenden Instance erstellen und diese an die EC2 Client-Funktion übergeben. `MonitorInstances`

Beinhaltet

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

Code

```
    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;
    }
```

Siehe vollständiges Beispiel.

Deaktivieren Sie die Instanzüberwachung

Um die Überwachung einer Instanz zu beenden, erstellen Sie eine UnmonitorInstancesRequestmit der ID der Instanz, deren Überwachung beendet werden soll, und übergeben Sie sie an die `UnmonitorInstances` Funktion des EC2 Clients.

Beinhaltet

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

Code

```
    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;
    }
```

Siehe vollständiges Beispiel.

Weitere Informationen

- RunInstancesin der Amazon EC2 API-Referenz
- DescribeInstancesin der Amazon EC2 API-Referenz
- StartInstancesin der Amazon EC2 API-Referenz
- StopInstancesin der Amazon EC2 API-Referenz
- RebootInstancesin der Amazon EC2 API-Referenz
- DescribeInstancesin der Amazon EC2 API-Referenz
- MonitorInstancesin der Amazon EC2 API-Referenz
- UnmonitorInstancesin der Amazon EC2 API-Referenz

Verwenden von Elastic IP-Adressen in Amazon EC2

Voraussetzungen

Bevor Sie beginnen, empfehlen wir Ihnen, Getting started using the AWS SDK für C++ zu lesen.

Laden Sie den Beispielcode herunter und erstellen Sie die Lösung wie unter beschriebenErste Schritte mit Codebeispielen.

Um die Beispiele ausführen zu können, muss das Benutzerprofil, das Ihr Code für die Anfragen verwendet, über die entsprechenden Berechtigungen verfügen AWS (für den Dienst und die Aktion). Weitere Informationen finden Sie unter Bereitstellen von AWS Anmeldeinformationen.

Weisen Sie eine Elastic IP-Adresse zu

Um eine Elastic IP-Adresse zu verwenden, verknüpfen Sie sie zuerst mit Ihrem Konto und anschließend mit Ihrer Instance oder Netzwerkschnittstelle.

Um eine Elastic IP-Adresse zuzuweisen, rufen Sie die `AllocateAddress` Funktion des EC2 Clients mit einem AllocateAddressRequestObjekt auf, das den Netzwerktyp (klassisch EC2 oder VPC) enthält.

> **Warning**
>
> EC2-Classic wird am 15. August 2022 eingestellt. Wir empfehlen Ihnen die Migration von EC2-Classic zu einer VPC. Weitere Informationen finden Sie unter Migration von EC2-Classic zu einer VPC im Amazon EC2 EC2-Benutzerhandbuch für Linux-Instances oder im Amazon EC2 EC2-Benutzerhandbuch für Windows-Instances. Lesen Sie außerdem den Blogbeitrag EC2-Classic Networking is Retiring – Here's How to Prepare (EC2-Classic Networking wird außer Betrieb genommen – so bereiten Sie sich vor).

Die AllocateAddressResponseKlasse im Antwortobjekt enthält eine Zuweisungs-ID, mit der Sie die Adresse einer Instance zuordnen können, indem Sie die Zuweisungs-ID und die Instance-ID in a AssociateAddressRequestan die Client-Funktion übergeben. EC2 `AssociateAddress`

Beinhaltet

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

Code

```
    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;
```

Siehe vollständiges Beispiel.

Beschreiben von Elastic IP-Adressen

Rufen Sie die EC2 `DescribeAddresses` Client-Funktion auf, um die Elastic IP-Adressen aufzulisten, die Ihrem Konto zugewiesen sind. Sie gibt ein Ergebnisobjekt zurück, das ein enthält, mit DescribeAddressesResponsedem Sie eine Liste von Address-Objekten abrufen können, die die Elastic IP-Adressen in Ihrem Konto repräsentieren.

Beinhaltet

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

```
#include <iostream>
```

Code

```
   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;
   }
```

Siehe vollständiges Beispiel.

Freigeben einer Elastic IP-Adresse

Um eine Elastic IP-Adresse freizugeben, rufen Sie die `ReleaseAddress` Funktion des EC2 Clients auf und übergeben ihr eine, die die Zuweisungs-ID der Elastic IP-Adresse ReleaseAddressRequestenthält, die Sie freigeben möchten.

Beinhaltet

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

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

Code

```
    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;
    }
```

Nachdem Sie eine Elastic IP-Adresse veröffentlicht haben, wird sie für den AWS IP-Adresspool freigegeben und steht Ihnen danach möglicherweise nicht mehr zur Verfügung. Achten Sie darauf, die DNS-Datensätze sowie alle Server und Geräte zu aktualisieren, die mit der Adresse kommunizieren. Wenn Sie versuchen, eine Elastic IP-Adresse freizugeben, die Sie bereits veröffentlicht haben, erhalten Sie eine AuthFailureFehlermeldung, wenn die Adresse bereits einem anderen AWS Konto zugewiesen ist.

Wenn Sie eine Standard-VPC verwenden, trennt die Freigabe einer Elastic IP-Adresse diese automatisch von allen Instances, mit denen sie verknüpft ist. Verwenden Sie die Funktion des EC2 Clients, um die Zuordnung einer Elastic IP-Adresse zu trennen, ohne sie freizugeben. `DisassociateAddress`

Wenn Sie einen Nicht-Standard-VPC verwenden, müssen Sie die Verknüpfung der Elastic IP-Adresse mit `DisassociateAddress` aufheben, bevor Sie versuchen, sie freizugeben. Andernfalls gibt Amazon EC2 einen Fehler zurück (UngültigIPAddress). InUse).

Siehe vollständiges Beispiel.

Weitere Informationen

- Elastic IP-Adressen im Amazon EC2 EC2-Benutzerhandbuch

- AllocateAddressin der Amazon EC2 API-Referenz
- DescribeAddressesin der Amazon EC2 API-Referenz
- ReleaseAddressin der Amazon EC2 API-Referenz

Verwenden von Regionen und Availability Zones für Amazon EC2

Voraussetzungen

Bevor Sie beginnen, empfehlen wir Ihnen, Erste Schritte mit dem AWS SDK für C++ zu lesen.

Laden Sie den Beispielcode herunter und erstellen Sie die Lösung wie unter beschriebenErste Schritte mit Codebeispielen.

Um die Beispiele ausführen zu können, muss das Benutzerprofil, das Ihr Code für die Anfragen verwendet, über die entsprechenden Berechtigungen verfügen AWS (für den Dienst und die Aktion). Weitere Informationen finden Sie unter Bereitstellen von AWS Anmeldeinformationen.

Beschreiben von Regionen

Um die AWS-Regionen verfügbaren Optionen aufzulisten AWS-Konto, rufen Sie die EC2 `DescribeRegions` Client-Funktion mit einem auf DescribeRegionsRequest.

DescribeRegionsResponseIm Ergebnis erhalten Sie ein Objekt. Rufen Sie seine `GetRegions` Funktion auf, um eine Liste von Region-Objekten zu erhalten, die jede Region repräsentieren.

Beinhaltet

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

Code

```
    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;
    }
```

Siehe vollständiges Beispiel.

Beschreiben von Availability Zones

Um alle Verfügbarkeitszonen aufzulisten, die für Ihr Konto verfügbar sind, rufen Sie die EC2 `DescribeAvailabilityZones` Client-Funktion mit einem auf DescribeAvailabilityZonesRequest.

DescribeAvailabilityZonesResponseIm Ergebnis erhalten Sie ein Objekt. Rufen Sie seine `GetAvailabilityZones` Funktion auf, um eine Liste von AvailabilityZoneObjekten abzurufen, die jede Verfügbarkeitszone repräsentieren.

Beinhaltet

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

Code

```
   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;

    }
```

Siehe vollständiges Beispiel.

Weitere Informationen

- Regionen und Availability Zones im Amazon EC2 EC2-Benutzerhandbuch
- DescribeRegionsin der Amazon EC2 API-Referenz
- DescribeAvailabilityZonesin der Amazon EC2 API-Referenz

Arbeiten mit Amazon EC2-Schlüsselpaaren

Voraussetzungen

Bevor Sie beginnen, empfehlen wir Ihnen, Erste Schritte mit dem AWS SDK für C++ zu lesen.

Laden Sie den Beispielcode herunter und erstellen Sie die Lösung wie unter beschriebenErste Schritte mit Codebeispielen.

Um die Beispiele ausführen zu können, muss das Benutzerprofil, das Ihr Code für die Anfragen verwendet, über die entsprechenden Berechtigungen verfügen AWS (für den Dienst und die Aktion). Weitere Informationen finden Sie unter Bereitstellen von AWS Anmeldeinformationen.

Erstellen eines Schlüsselpaares

Um ein key pair zu erstellen, rufen Sie die `CreateKeyPair` Funktion des EC2 Clients mit einer auf CreateKeyPairRequest, die den Namen des Schlüssels enthält.

Beinhaltet

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

Code

```
    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;
        }

    }
```

Siehe vollständiges Beispiel.

Schlüsselpaare beschreiben

Um Ihre Schlüsselpaare aufzulisten oder Informationen über sie zu erhalten, rufen Sie die EC2 `DescribeKeyPairs` Client-Funktion mit einem auf DescribeKeyPairsRequest.

Sie erhalten eine DescribeKeyPairsResponse, mit der Sie auf die Liste der Schlüsselpaare zugreifen können, indem Sie ihre `GetKeyPairs` Funktion aufrufen, die eine Liste von KeyPairInfoObjekten zurückgibt.

Beinhaltet

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

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

Code

```
    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;
    }
```

Siehe vollständiges Beispiel.

Löschen eines Schlüsselpaars

Um ein key pair zu löschen, rufen Sie die `DeleteKeyPair` Funktion des EC2 Clients auf und übergeben Sie ihr eine DeleteKeyPairRequest, die den Namen des zu löschenden key pair enthält.

Beinhaltet

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

Code

```
    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;
    }
```

Siehe vollständiges Beispiel.

Weitere Informationen

- Amazon EC2 EC2-Schlüsselpaare im Amazon EC2 EC2-Benutzerhandbuch
- CreateKeyPairin der Amazon EC2 API-Referenz
- DescribeKeyPairsin der Amazon EC2 API-Referenz
- DeleteKeyPairin der Amazon EC2 API-Referenz

Arbeiten mit Sicherheitsgruppen in Amazon EC2

Voraussetzungen

Bevor Sie beginnen, empfehlen wir Ihnen, Erste Schritte mit dem AWS SDK für C++ zu lesen.

Laden Sie den Beispielcode herunter und erstellen Sie die Lösung wie unter beschriebenErste Schritte mit Codebeispielen.

Um die Beispiele ausführen zu können, muss das Benutzerprofil, das Ihr Code für die Anfragen verwendet, über die entsprechenden Berechtigungen verfügen AWS (für den Dienst und die Aktion). Weitere Informationen finden Sie unter Bereitstellen von AWS Anmeldeinformationen.

Erstellen einer Sicherheitsgruppe

Um eine Sicherheitsgruppe zu erstellen, rufen Sie die EC2 `CreateSecurityGroup` Client-Funktion mit einer auf CreateSecurityGroupRequest, die den Namen des Schlüssels enthält.

Beinhaltet

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

Code

```
    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;
```

Siehe vollständiges Beispiel.

Konfigurieren einer Sicherheitsgruppe

Eine Sicherheitsgruppe kann sowohl den eingehenden (eingehenden) als auch den ausgehenden (ausgehenden) Datenverkehr zu Ihren Amazon EC2 EC2-Instances kontrollieren.

Um Ihrer Sicherheitsgruppe Eingangsregeln hinzuzufügen, verwenden Sie die EC2 `AuthorizeSecurityGroupIngress` Client-Funktion und geben Sie den Namen der Sicherheitsgruppe und die Zugriffsregeln (IpPermission) an, die Sie ihr innerhalb eines Objekts zuweisen möchten. AuthorizeSecurityGroupIngressRequest Im folgenden Beispiel wird gezeigt, wie Sie einer Sicherheitsgruppe IP-Berechtigungen hinzufügen.

Beinhaltet

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

Code

```
   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;
   }
```

Um der Sicherheitsgruppe eine Ausgangsregel hinzuzufügen, geben Sie ähnliche Daten in einer `AuthorizeSecurityGroupEgress` Funktion AuthorizeSecurityGroupEgressRequestfür den EC2 Client an.

Siehe vollständiges Beispiel.

Beschreiben Sie Sicherheitsgruppen

Um Ihre Sicherheitsgruppen zu beschreiben oder Informationen über sie zu erhalten, rufen Sie die EC2 `DescribeSecurityGroups` Client-Funktion mit einem DescribeSecurityGroupsRequestauf.

DescribeSecurityGroupsResponseIm Ergebnis erhalten Sie ein Objekt, mit dem Sie auf die Liste der Sicherheitsgruppen zugreifen können, indem Sie dessen `GetSecurityGroups` Funktion aufrufen, die eine Liste von SecurityGroupObjekten zurückgibt.

Beinhaltet

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

Code

```
    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());
```

Siehe vollständiges Beispiel.

Löschen einer Sicherheitsgruppe

Um eine Sicherheitsgruppe zu löschen, rufen Sie die EC2 `DeleteSecurityGroup` Client-Funktion auf und übergeben ihr eine DeleteSecurityGroupRequest, die die ID der zu löschenden Sicherheitsgruppe enthält.

Beinhaltet

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

Code

```
    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;
    }
```

Siehe vollständiges Beispiel.

Weitere Informationen

- Amazon EC2-Sicherheitsgruppen im Amazon EC2 EC2-Benutzerhandbuch
- Autorisieren von eingehendem Traffic für Ihre Linux-Instances im Amazon EC2 EC2-Benutzerhandbuch
- CreateSecurityGroupin der Amazon EC2 API-Referenz
- DescribeSecurityGroupsin der Amazon EC2 API-Referenz
- DeleteSecurityGroupin der Amazon EC2 API-Referenz
- AuthorizeSecurityGroupIngressin der Amazon EC2 API-Referenz

Amazon S3 S3-Codebeispiele mit dem AWS SDK für C++

Amazon S3 ist ein Objektspeicher, der zum Speichern und Abrufen beliebiger Datenmengen von überall entwickelt wurde. Es gibt mehrere Klassen, die von der Schnittstelle AWS SDK für C++ zu Amazon S3 bereitgestellt werden.

> **Note**
>
> In diesem Handbuch ist nur der Code enthalten, der zur Demonstration bestimmter Techniken erforderlich ist. Der vollständige Beispielcode ist jedoch unter verfügbar GitHub. Auf können GitHub Sie eine einzelne Quelldatei herunterladen oder das Repository lokal klonen, um alle Beispiele abzurufen, zu erstellen und auszuführen.

- `S3Client`-Klasse

  Die `S3Client` Bibliothek ist eine Amazon S3 S3-Schnittstelle mit vollem Funktionsumfang.

  Das `list_buckets_disabling_dns_cache.cpp` Beispiel in diesem Set ist speziell für die Arbeit mit eingeschaltetem CURL konzipiert Linux/Mac (kann jedoch so geändert werden, dass es unter Windows funktioniert). Wenn Sie Windows verwenden, löschen Sie die Datei, `list_buckets_disabling_dns_cache.cpp` bevor Sie das Projekt erstellen, da sie auf dem CURL HttpClient von Linux basiert.

  Der Beispielcode, der das verwendet, `S3Client` befindet sich im `s3`Ordner auf Github. Eine vollständige Liste der Funktionen, die in diesem Beispielsatz demonstriert wurden, finden Sie in der Readme-Datei auf Github.

  Teile des `s3` Beispielsatzes werden in diesem Handbuch ausführlicher behandelt:

  - Buckets erstellen, auflisten und löschen
  - Operationen für Objekte— Datenobjekte hochladen und herunterladen
  - Amazon S3 S3-Zugriffsberechtigungen verwalten
  - Verwaltung des Zugriffs auf Amazon S3 S3-Buckets mithilfe von Bucket-Richtlinien
  - Konfiguration eines Amazon S3 S3-Buckets als Website

- `S3CrtClient`-Klasse

  Das `S3CrtClient` wurde in Version 1.9 des SDK hinzugefügt. `S3CrtClient`bietet einen hohen Durchsatz für Amazon S3 S3-GET- (Download) - und PUT- (Upload) -Operationen. Das `S3CrtClient` ist auf der Grundlage der AWS Common Runtime (CRT) -Bibliotheken implementiert.

  Der Beispielcode, der das verwendet, `S3CrtClient` befindet sich im `s3-crt`Ordner auf Github. Eine vollständige Liste der Funktionen, die in diesem Beispielsatz demonstriert wurden, finden Sie in der Readme-Datei auf Github.

  - `S3CrtClient`Für Amazon S3 S3-Operationen verwenden

- `TransferManager`-Klasse

  `TransferManager`ist ein vollständig verwalteter Service, der die Übertragung von Dateien über das File Transfer Protocol (FTP), File Transfer Protocol over SSL (FTPS) oder Secure Shell (SSH) File Transfer Protocol (SFTP) direkt in und aus Amazon S3 ermöglicht.

Der Beispielcode, der das verwendet, `TransferManager` befindet sich im `transfer-manager`Ordner auf Github. Eine vollständige Liste der Funktionen, die in diesem Beispielsatz demonstriert wurden, finden Sie in der Readme-Datei auf Github.

- TransferManager Für Amazon S3 S3-Operationen verwenden

Buckets erstellen, auflisten und löschen

Jedes Objekt oder jede Datei in Amazon Simple Storage Service (Amazon S3) ist in einem Bucket enthalten, der einen Ordner mit Objekten darstellt. Jeder Bucket hat einen Namen, der innerhalb eines Buckets weltweit einzigartig ist AWS. Weitere Informationen finden Sie unter Arbeiten mit Amazon S3 S3-Buckets im Amazon Simple Storage Service-Benutzerhandbuch.

Voraussetzungen

Bevor Sie beginnen, empfehlen wir Ihnen, Erste Schritte mit dem AWS SDK für C++ zu lesen.

Laden Sie den Beispielcode herunter und erstellen Sie die Lösung wie unter beschriebenErste Schritte mit Codebeispielen.

Um die Beispiele ausführen zu können, muss das Benutzerprofil, das Ihr Code für die Anfragen verwendet, über die entsprechenden Berechtigungen verfügen AWS (für den Dienst und die Aktion). Weitere Informationen finden Sie unter Bereitstellen von AWS Anmeldeinformationen.

Buckets auflisten

Um das `list_buckets` Beispiel auszuführen, navigieren Sie an einer Befehlszeile zu dem Ordner, in dem Ihr Build-System Ihre ausführbaren Build-Dateien erstellt. Führen Sie die ausführbare Datei wie folgt aus `run_list_buckets` (der vollständige Dateiname der ausführbaren Datei hängt von Ihrem Betriebssystem ab). In der Ausgabe werden die Buckets Ihres Kontos aufgeführt, falls Sie welche haben, oder es wird eine leere Liste angezeigt, wenn Sie keine Buckets haben.

Darin gibt `list_buckets.cpp` es zwei Methoden.

- `main()`Anrufe`ListBuckets()`.
- `ListBuckets()`verwendet das SDK, um Ihre Buckets abzufragen.

Das `S3Client` Objekt ruft die `ListBuckets()` Methode des SDK auf. Bei Erfolg gibt die Methode ein `ListBucketOutcome` Objekt zurück, das ein `ListBucketResult` Objekt enthält. Das

`ListBucketResult` Objekt ruft die `GetBuckets()` Methode auf, um eine Liste von `Bucket` Objekten abzurufen, die Informationen zu jedem Amazon S3 S3-Bucket in Ihrem Konto enthalten.

Code

```
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;
}
```

Das vollständige list_buckets-Beispiel finden Sie auf Github.

Erstellen eines -Buckets

Um das `create_bucket` Beispiel auszuführen, navigieren Sie an einer Befehlszeile zu dem Ordner, in dem Ihr Build-System Ihre ausführbaren Build-Dateien erstellt. Führen Sie die ausführbare Datei wie folgt aus `run_create_bucket` (der vollständige Dateiname der ausführbaren Datei hängt von Ihrem Betriebssystem ab). Der Code erstellt einen leeren Bucket unter Ihrem Konto und zeigt dann an, ob die Anfrage erfolgreich war oder nicht.

`create_bucket.cpp`In gibt es zwei Methoden.

- `main()`Anrufe`CreateBucket()`. `main()`In müssen Sie die AWS-Region Region Ihres Kontos ändern, indem Sie die verwendene`num`. Sie können die Region Ihres Kontos einsehen AWS-Managementkonsole, indem Sie sich bei anmelden und die Region in der oberen rechten Ecke suchen.

- `CreateBucket()`verwendet das SDK, um einen Bucket zu erstellen.

Das `S3Client` Objekt ruft die `CreateBucket()` Methode des SDK auf und übergibt eine `CreateBucketRequest` mit dem Namen des Buckets. Standardmäßig werden Buckets in der Region us-east-1 (Nord-Virginia) erstellt. Wenn Ihre Region nicht us-east-1 ist, richtet der Code eine Bucket-Einschränkung ein, um sicherzustellen, dass der Bucket in Ihrer Region erstellt wird.

Code

```
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();
}
```

Das vollständige create_buckets-Beispiel finden Sie auf Github.

Löschen eines -Buckets

Um das `delete_bucket` Beispiel auszuführen, navigieren Sie an einer Befehlszeile zu dem Ordner, in dem Ihr Build-System Ihre ausführbaren Build-Dateien erstellt. Führen Sie die ausführbare Datei wie folgt aus `run_delete_bucket` (der vollständige Dateiname der ausführbaren Datei hängt von Ihrem Betriebssystem ab). Der Code löscht den angegebenen Bucket in Ihrem Konto und zeigt dann an, ob die Anfrage erfolgreich war oder nicht.

Darin `delete_bucket.cpp` gibt es zwei Methoden.

- `main()`ruft an`DeleteBucket()`. `main()`In müssen Sie die AWS-Region Region Ihres Kontos ändern, indem Sie die verwendene`num`. Sie müssen das auch in `bucket_name` den Namen des Buckets ändern, den Sie löschen möchten.
- `DeleteBucket()`verwendet das SDK, um den Bucket zu löschen.

Das `S3Client` Objekt verwendet die `DeleteBucket()` Methode des SDK und übergibt ein `DeleteBucketRequest` Objekt mit dem Namen des zu löschenden Buckets. Der Bucket muss leer sein, um erfolgreich zu sein.

Code

```
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();
}
```

Das vollständige delete_bucket-Beispiel finden Sie auf Github.

Operationen für Objekte

Ein Amazon S3 S3-Objekt stellt eine Datei dar, bei der es sich um eine Sammlung von Daten handelt. Jedes Objekt muss in einem Bucket enthalten sein.

Voraussetzungen

Bevor Sie beginnen, empfehlen wir Ihnen, Erste Schritte mit dem zu lesen AWS SDK für C++.

Laden Sie den Beispielcode herunter und erstellen Sie die Lösung wie unter beschriebenErste Schritte mit Codebeispielen.

Um die Beispiele ausführen zu können, muss das Benutzerprofil, das Ihr Code für die Anfragen verwendet, über die entsprechenden Berechtigungen verfügen AWS (für den Dienst und die Aktion). Weitere Informationen finden Sie unter Bereitstellen von AWS Anmeldeinformationen.

Laden Sie eine Datei in einen Bucket hoch

Verwenden Sie die `S3Client` `PutObject` Objektfunktion und geben Sie ihr einen Bucket-Namen, einen Schlüsselnamen und eine hochzuladende Datei. `Aws::FStream`wird verwendet, um den Inhalt der lokalen Datei in den Bucket hochzuladen. Der Bucket muss existieren, andernfalls tritt ein Fehler auf.

Ein Beispiel für das asynchrone Hochladen von Objekten finden Sie unter Asynchrone Programmierung mit dem AWS SDK für C++

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();
}
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Laden Sie eine Zeichenfolge in einen Bucket hoch

Verwenden Sie die `S3Client` `PutObject` Objektfunktion und geben Sie ihr einen Bucket-Namen, einen Schlüsselnamen und eine hochzuladende Datei. Der Bucket muss existieren, andernfalls tritt ein Fehler auf. Dieses Beispiel unterscheidet sich vom vorherigen dadurch, dass es verwendet wird`Aws::StringStream`, um ein speicherinternes String-Datenobjekt direkt in einen Bucket hochzuladen.

Ein Beispiel für das asynchrone Hochladen von Objekten finden Sie unter Asynchrone Programmierung mit dem AWS SDK für C++

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();
}
```

Auf GitHub finden Sie ein vollständiges Beispiel.

List objects

Verwenden Sie die Objektfunktion, um eine Liste der Objekte in einem Bucket abzurufen`S3Client`. `ListObjects` Geben Sie ihr ein`ListObjectsRequest`, das Sie mit dem Namen eines Buckets angegeben haben, dessen Inhalt Sie auflisten möchten.

Die `ListObjects` Funktion gibt ein `ListObjectsOutcome` Objekt zurück, mit dem Sie eine Liste von Objekten in Form von `Object` Instanzen abrufen können.

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;
}
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Herunterladen eines Objekts

Verwenden Sie die `S3Client` `GetObject` Objektfunktion und übergeben Sie ihr eine`GetObjectRequest`, die Sie mit dem Namen eines Buckets und dem Objektschlüssel zum Herunterladen festgelegt haben. `GetObject`gibt ein `GetObjectOutcome`Objekt zurück, das aus a

GetObjectResultund a besteht S3Error. GetObjectResultkann verwendet werden, um auf die Daten des S3-Objekts zuzugreifen.

Das folgende Beispiel lädt ein Objekt von Amazon S3 herunter. Der Objektinhalt wird in einer lokalen Variablen gespeichert und die erste Zeile des Inhalts wird an die Konsole ausgegeben.

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();
}
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Ein Objekt löschen

Verwenden Sie die DeleteObject Funktion des S3Client Objekts und übergeben Sie ihm eineDeleteObjectRequest, die Sie mit dem Namen eines Buckets und eines Objekts zum Herunterladen festgelegt haben. Der angegebene Bucket und der Objektschlüssel müssen existieren, andernfalls tritt ein Fehler auf.

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();
}
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Amazon S3 S3-Zugriffsberechtigungen verwalten

Zugriffsberechtigungen für einen Amazon S3 S3-Bucket oder ein Amazon S3-Objekt werden in einer Zugriffskontrollliste (ACL) definiert. Die ACL gibt den Eigentümer der Grants bucket/object und eine Liste der Grants an. Jede Grant gibt einen Benutzer (oder Empfänger) und die Zugriffsberechtigungen des Benutzers auf den Bucket/das Objekt an, z. B. READ- oder WRITE-Zugriff.

Voraussetzungen

Bevor Sie beginnen, empfehlen wir Ihnen, Erste Schritte mit dem zu lesen. AWS SDK für C++

Laden Sie den Beispielcode herunter und erstellen Sie die Lösung wie unter beschriebenErste Schritte mit Codebeispielen.

Um die Beispiele ausführen zu können, muss das Benutzerprofil, das Ihr Code für die Anfragen verwendet, über die entsprechenden Berechtigungen verfügen AWS (für den Dienst und die Aktion). Weitere Informationen finden Sie unter Bereitstellen von AWS Anmeldeinformationen.

Die Zugriffskontrollliste eines Objekts verwalten

Die Zugriffskontrollliste für ein Objekt kann durch Aufrufen der `S3Client` Methode abgerufen werden`GetObjectAcl`. Die Methode akzeptiert die Namen des Objekts und seines Buckets. Der Rückgabewert beinhaltet die ACLs `Owner` und eine Liste von`Grants`.

```
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";
    }
}
```

Die ACL kann geändert werden, indem entweder eine neue ACL erstellt oder die in der aktuellen ACL angegebenen Grants geändert werden. Die aktualisierte ACL wird zur neuen aktuellen ACL, indem sie an die `PutObjectAcl` Methode übergeben wird.

Der folgende Code verwendet die von abgerufene ACL `GetObjectAcl` und fügt ihr einen neuen Grant hinzu. Der Benutzer oder Empfänger erhält die READ-Berechtigung für das Objekt. Die geänderte ACL wird an übergeben `PutObjectAcl` und ist somit die neue aktuelle ACL.

```
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;
}
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Die Zugriffskontrollliste eines Buckets verwalten

In den meisten Fällen besteht die bevorzugte Methode zur Festlegung der Zugriffsberechtigungen für einen Bucket darin, eine Bucket-Richtlinie zu definieren. Buckets unterstützen jedoch auch Zugriffskontrolllisten für Benutzer, die sie verwenden möchten.

Die Verwaltung einer Zugriffskontrollliste für einen Bucket ist identisch mit der für ein Objekt verwendeten. Die `GetBucketAcl` Methode ruft die aktuelle ACL eines Buckets ab und `PutBucketAcl` wendet eine neue ACL auf den Bucket an.

Der folgende Code zeigt, wie eine Bucket-ACL abgerufen und festgelegt wird.

```
//! 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;
}
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Verwaltung des Zugriffs auf Amazon S3 S3-Buckets mithilfe von Bucket-Richtlinien

Sie können eine Bucket-Richtlinie einrichten, abrufen oder löschen, um den Zugriff auf Ihre Amazon S3 S3-Buckets zu verwalten.

Voraussetzungen

Bevor Sie beginnen, empfehlen wir Ihnen, Erste Schritte mit dem AWS SDK für C++ zu lesen.

Laden Sie den Beispielcode herunter und erstellen Sie die Lösung wie unter beschriebenErste Schritte mit Codebeispielen.

Um die Beispiele ausführen zu können, muss das Benutzerprofil, das Ihr Code für die Anfragen verwendet, über die entsprechenden Berechtigungen verfügen AWS (für den Dienst und die Aktion). Weitere Informationen finden Sie unter Bereitstellen von AWS Anmeldeinformationen.

Festlegen einer Bucket-Richtlinie

Sie können die Bucket-Richtlinie für einen bestimmten S3-Bucket festlegen, indem Sie die `PutBucketPolicy` Funktion `S3Client`'s aufrufen und ihr den Bucket-Namen und die JSON-Repräsentation der Richtlinie in a zur Verfügung stellen PutBucketPolicyRequest.

Code

```
//! 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**
>
> Die JsonValue Utility-Klasse Aws: :Utils: :Json:: kann Ihnen dabei helfen, gültige JSON-Objekte zu erstellen, an die Sie übergeben werden können. `PutBucketPolicy`

Auf GitHub finden Sie ein vollständiges Beispiel.

Abrufen einer Bucket-Richtlinie

Um die Richtlinie für einen Amazon S3 S3-Bucket abzurufen, rufen Sie die `GetBucketPolicy` Funktion `S3Client`'s auf und übergeben ihr den Namen des Buckets in GetBucketPolicyRequesta.

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();
}
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Löschen einer Bucket-Richtlinie

Um eine Bucket-Richtlinie zu löschen, rufen Sie die `DeleteBucketPolicy` Funktion `S3Client`'s auf und geben Sie ihr den Bucket-Namen in einer DeleteBucketPolicyRequest.

Code

```
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();
}
```

Diese Funktion ist auch dann erfolgreich, wenn der Bucket noch keine Richtlinie hat. Wenn Sie den Namen eines Buckets angeben, der noch nicht vorhanden ist oder für den Sie keinen Zugriff haben, wird eine `AmazonServiceException` ausgelöst.

Auf GitHub finden Sie ein vollständiges Beispiel.

Weitere Infos

- PutBucketPolicyin der Amazon Simple Storage Service API-Referenz
- GetBucketPolicyin der Amazon Simple Storage Service API-Referenz
- DeleteBucketPolicyin der Amazon Simple Storage Service API-Referenz
- Sprachübersicht der Zugriffsrichtlinien im Amazon Simple Storage Service-Benutzerhandbuch
- Beispiele für Bucket-Richtlinien im Amazon Simple Storage Service-Benutzerhandbuch

Konfiguration eines Amazon S3 S3-Buckets als Website

Sie können einen Amazon S3 S3-Bucket so konfigurieren, dass er sich wie eine Website verhält. Hierzu müssen Sie die Website-Konfiguration festlegen.

Voraussetzungen

Bevor Sie beginnen, empfehlen wir Ihnen, Erste Schritte mit dem zu lesen AWS SDK für C++.

Laden Sie den Beispielcode herunter und erstellen Sie die Lösung wie unter beschriebenErste Schritte mit Codebeispielen.

Um die Beispiele ausführen zu können, muss das Benutzerprofil, das Ihr Code für die Anfragen verwendet, über die entsprechenden Berechtigungen verfügen AWS (für den Dienst und die Aktion). Weitere Informationen finden Sie unter Bereitstellen von AWS Anmeldeinformationen.

Festlegen der Website-Konfiguration eines Buckets

Um die Website-Konfiguration eines Amazon S3 S3-Buckets festzulegen, rufen Sie die `S3Client` `PutBucketWebsite` Funktion mit einem PutBucketWebsiteRequestObjekt auf, das den Bucket-Namen und seine Website-Konfiguration enthält, die in einem WebsiteConfigurationObjekt bereitgestellt werden.

Das Festlegen eines Index-Dokuments ist erforderlich. Alle anderen Parameter sind optional.

Code

```
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**
>
> Beim Festlegen einer Website-Konfiguration werden die Zugriffsberechtigungen für den Bucket nicht geändert. Um die enthaltenen Dateien im Internet sichtbar zu machen, müssen Sie zusätzlich eine Bucket-Richtlinie festlegen, durch die der öffentliche Lesezugriff für die Dateien in dem Bucket ermöglicht wird. Weitere Informationen finden Sie unter Verwalten des Zugriffs auf Amazon S3-Buckets mit Bucket-Richtlinien.

Auf GitHub finden Sie ein vollständiges Beispiel.

Abruf der Website-Konfiguration eines Buckets

Um die Website-Konfiguration eines Amazon S3 S3-Buckets abzurufen, rufen Sie die `S3Client` `GetBucketWebsite` Funktion mit einem auf, das den Namen des Buckets GetBucketWebsiteRequestenthält, für den die Konfiguration abgerufen werden soll.

Die Konfiguration wird als GetBucketWebsiteResultObjekt innerhalb des Ergebnisobjekts zurückgegeben. Wenn keine Website-Konfiguration für den Bucket vorhanden ist, wird `null` zurückgegeben.

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();
```

```
}
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Löschen der Website-Konfiguration eines Buckets

Um die Website-Konfiguration eines Amazon S3 S3-Buckets zu löschen, rufen Sie die `DeleteBucketWebsite` Funktion `S3Client` s mit einem DeleteBucketWebsiteRequest: auf, das den Namen des Buckets enthält, aus dem die Konfiguration gelöscht werden soll.

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();
}
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Weitere Informationen

- PUT Bucket-Website in der Amazon Simple Storage Service API-Referenz
- GET Bucket-Website in der Amazon Simple Storage Service API-Referenz
- DELETE Bucket-Website in der Amazon Simple Storage Service API-Referenz

TransferManager Für Amazon S3 S3-Operationen verwenden

Sie können die AWS SDK für C++ `TransferManager` Klasse verwenden, um Dateien zuverlässig aus der lokalen Umgebung nach Amazon S3 zu übertragen und Objekte von einem Amazon S3 S3-Standort an einen anderen zu kopieren. `TransferManager`kann den Fortschritt einer Übertragung abrufen und Uploads und Downloads pausieren oder fortsetzen.

> **Note**
>
> Um zu vermeiden, dass unvollständige oder teilweise Uploads in Rechnung gestellt werden, empfehlen wir Ihnen, die AbortIncompleteMultipartUploadLebenszyklusregel für Ihre Amazon S3 S3-Buckets zu aktivieren.
> Diese Regel weist Amazon S3 an, mehrteilige Uploads abzubrechen, die nicht innerhalb einer bestimmten Anzahl von Tagen nach der Initiierung abgeschlossen werden. Wenn das festgelegte Zeitlimit überschritten wird, bricht Amazon S3 den Upload ab und löscht dann die unvollständigen Upload-Daten.
> Weitere Informationen finden Sie unter Einstellung der Lebenszykluskonfiguration für einen Bucket im Amazon S3 S3-Benutzerhandbuch.

Voraussetzungen

Bevor Sie beginnen, empfehlen wir Ihnen, Erste Schritte mit dem zu lesen AWS SDK für C++.

Laden Sie den Beispielcode herunter und erstellen Sie die Lösung wie unter beschriebenErste Schritte mit Codebeispielen.

Um die Beispiele ausführen zu können, muss das Benutzerprofil, das Ihr Code für die Anfragen verwendet, über die entsprechenden Berechtigungen verfügen AWS (für den Dienst und die Aktion). Weitere Informationen finden Sie unter Bereitstellen von AWS Anmeldeinformationen.

Objekt hoch- und herunterladen mit **`TransferManager`**

Dieses Beispiel zeigt, wie große Objekte im Speicher `TransferManager`übertragen werden. `UploadFile`und `DownloadFile` Methoden werden beide asynchron aufgerufen und geben a zurück`TransferHandle`, um den Status Ihrer Anfrage zu verwalten. Wenn das hochgeladene Objekt größer als `bufferSize` dann ist, wird ein mehrteiliger Upload durchgeführt. Die `bufferSize` Standardeinstellung ist 5 MB, dies kann jedoch über konfiguriert werden. `TransferManagerConfiguration`

```
        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;
```

Auf GitHub finden Sie ein vollständiges Beispiel.

`S3CrtClient`Für Amazon S3 S3-Operationen verwenden

Die `S3CrtClient` Klasse ist in Version 1.9 von verfügbar AWS SDK für C++ und verbessert den Durchsatz beim Hoch- und Herunterladen großer Datendateien zu und von Amazon S3. Weitere Informationen zu den Verbesserungen dieser Version finden Sie unter Verbessern des Amazon S3 S3-Durchsatzes mit Version AWS SDK für C++ 1.9

Das `S3CrtClient` ist auf der Grundlage der AWS Common Runtime (CRT) -Bibliotheken implementiert.

> **Note**
>
> Um zu vermeiden, dass unvollständige oder teilweise Uploads in Rechnung gestellt werden, empfehlen wir Ihnen, die AbortIncompleteMultipartUploadLebenszyklusregel für Ihre Amazon S3 S3-Buckets zu aktivieren.
> Diese Regel weist Amazon S3 an, mehrteilige Uploads abzubrechen, die nicht innerhalb einer bestimmten Anzahl von Tagen nach der Initiierung abgeschlossen werden. Wenn das festgelegte Zeitlimit überschritten wird, bricht Amazon S3 den Upload ab und löscht dann die unvollständigen Upload-Daten.
> Weitere Informationen finden Sie unter Einstellung der Lebenszykluskonfiguration für einen Bucket im Amazon S3 S3-Benutzerhandbuch.

Voraussetzungen

Bevor Sie beginnen, empfehlen wir Ihnen, Erste Schritte mit dem zu lesen AWS SDK für C++.

Laden Sie den Beispielcode herunter und erstellen Sie die Lösung wie unter beschriebenErste Schritte mit Codebeispielen.

Um die Beispiele ausführen zu können, muss das Benutzerprofil, das Ihr Code für die Anfragen verwendet, über die entsprechenden Berechtigungen verfügen AWS (für den Dienst und die Aktion). Weitere Informationen finden Sie unter Bereitstellen von AWS Anmeldeinformationen.

Objekt hoch- und herunterladen mit **`S3CrtClient`**

Dieses Beispiel zeigt, wie Sie den verwenden `S3CrtClient`. Das Beispiel erstellt einen Bucket, lädt ein Objekt hoch, lädt das Objekt herunter und löscht dann die Datei und den Bucket. Eine PUT-Operation wird zu einem mehrteiligen Upload. Aus einer GET-Operation werden mehrere GET-Anfragen mit „Bereichsbereich“. Weitere Informationen zu mehrteiligen Uploads finden Sie unter Hochladen und Kopieren von Objekten mithilfe des mehrteiligen Uploads im Amazon S3 S3-Benutzerhandbuch.

Die bereitgestellte Datendatei,ny`.json`, wird in diesem Beispiel als mehrteiliger Upload hochgeladen. Dies kann nach einer erfolgreichen Ausführung des Programms anhand der Debug-Protokolle bestätigt werden.

Schlägt der Upload fehl, `AbortMultipartUpload` wird eine in der zugrunde liegenden CRT-Bibliothek ausgegeben, um alle bereits hochgeladenen Teile zu bereinigen. Allerdings können nicht alle Fehler intern behoben werden (z. B. wenn ein Netzwerkkabel abgezogen wird). Es wird empfohlen, eine Lebenszyklusregel für Ihren Amazon S3 S3-Bucket zu erstellen, um sicherzustellen, dass teilweise hochgeladene Daten nicht in Ihrem Konto verbleiben (teilweise hochgeladene Daten sind weiterhin fakturierbar). Informationen zum Einrichten einer Lebenszyklusregel finden Sie unter Erkennen und Löschen unvollständiger mehrteiliger Uploads zur Senkung der Amazon S3 S3-Kosten.

Verwenden Sie das Debug-Protokoll, um Details zu mehrteiligen Uploads zu untersuchen

1. Beachten Sie`main()`, dass es "TODO" Kommentare mit Anweisungen zur Aktualisierung des Codes gibt.

   a. Für`file_name`: Laden Sie über den Link im Codekommentar eine Beispieldatendatei herunter`.json`, oder verwenden Sie eine eigene große Datendatei.

   b. Für`region`: Aktualisieren Sie die `region` Variable mithilfe der Enumeration auf die AWS-Region Ihres Kontos. Um die Region Ihres Kontos zu finden, melden Sie sich bei an und suchen Sie die Region in der oberen rechten Ecke. AWS-Managementkonsole

2. Erstellen Sie das Beispiel.
3. Kopieren Sie die durch die Variable angegebene Datei `file_name` in Ihren ausführbaren Ordner und führen Sie die `s3-crt-demo` ausführbare Datei aus.
4. Suchen Sie in Ihrem ausführbaren Ordner nach der neuesten `.log` Datei.
5. Öffnen Sie die Protokolldatei, wählen Sie Suchen und geben Sie die Eingabetaste ein**`partNumber`**.

6. Das Protokoll enthält Einträge, die den folgenden ähneln, wobei die `partNumber` und für jeden Teil der hochgeladenen Datei angegeben `uploadId` sind:

```
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
```

und

```
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
```

Auf GitHub finden Sie ein vollständiges Beispiel.

Amazon SQS SQS-Codebeispiele mit dem AWS SDK für C++

Amazon Simple Queue Service (Amazon SQS) ist ein vollständig verwalteter Message Queuing-Service, der es einfach macht, Microservices, verteilte Systeme und serverlose Anwendungen zu entkoppeln und zu skalieren. Sie können die folgenden Beispiele verwenden, um Amazon SQS mit dem AWS SDK für C++ zu programmieren.

> **Note**
>
> In diesem Handbuch ist nur der Code enthalten, der zur Veranschaulichung bestimmter Techniken erforderlich ist. Der vollständige Beispielcode ist jedoch unter GitHub verfügbar. Auf können GitHub Sie eine einzelne Quelldatei herunterladen oder das Repository lokal klonen, um alle Beispiele abzurufen, zu erstellen und auszuführen.

Themen

- Mit Amazon SQS Message Queues arbeiten
- Senden, Empfangen und Löschen von Amazon SQS SQS-Nachrichten
- Long Polling für Amazon SQS SQS-Nachrichtenwarteschlangen aktivieren
- Einstellen des Sichtbarkeits-Timeouts in Amazon SQS

- Verwenden von Warteschlangen für unzustellbare Nachrichten in Amazon SQS

Mit Amazon SQS Message Queues arbeiten

Eine Nachrichtenwarteschlange ist der logische Container, den Sie verwenden, um Nachrichten zuverlässig in Amazon SQS zu versenden. Es gibt zwei Arten von Warteschlangen: Standard und First-in-First-out-Verfahren (FIFO). Weitere Informationen zu Warteschlangen und den Unterschieden zwischen diesen Typen finden Sie im Amazon Simple Queue Service Developer Guide.

Diese C++-Beispiele zeigen Ihnen, wie Sie die verwenden, AWS SDK für C++ um eine Amazon SQS SQS-Warteschlange zu erstellen, aufzulisten, zu löschen und deren URL abzurufen.

Voraussetzungen

Bevor Sie beginnen, empfehlen wir Ihnen, Erste Schritte mit dem AWS SDK für C++ zu lesen.

Laden Sie den Beispielcode herunter und erstellen Sie die Lösung wie unter beschriebenErste Schritte mit Codebeispielen.

Um die Beispiele ausführen zu können, muss das Benutzerprofil, das Ihr Code für die Anfragen verwendet, über die entsprechenden Berechtigungen verfügen AWS (für den Dienst und die Aktion). Weitere Informationen finden Sie unter Bereitstellen von AWS Anmeldeinformationen.

Erstellen einer Warteschlange

Verwenden Sie die SQSClient `CreateQueue` Klassenmitgliedsfunktion und stellen Sie ihr ein CreateQueueRequestObjekt zur Verfügung, das die Warteschlangenparameter beschreibt.

Beinhaltet

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

Code

```
    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;
    }
```

Siehe vollständiges Beispiel.

Auflisten von Warteschlangen

Um Amazon SQS SQS-Warteschlangen für Ihr Konto aufzulisten, rufen Sie die SQSClient `ListQueues` Klassenmitgliedsfunktion auf und übergeben Sie ihr ein ListQueuesRequestObjekt.

Beinhaltet

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

Code

```
    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;
    }
```

Siehe vollständiges Beispiel.

Ruft die URL der Warteschlange ab

Rufen Sie die SQSClient `GetQueueUrl` Klassenmitgliedsfunktion auf, um die URL für eine bestehende Amazon SQS SQS-Warteschlange abzurufen.

Beinhaltet

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

Code

```
    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;
    }
```

Siehe vollständiges Beispiel.

Löschen einer Warteschlange

Geben Sie die URL zur SQSClient `DeleteQueue` Klassenmitgliedsfunktion an.

Beinhaltet

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

Code

```
    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;
    }
```

Siehe vollständiges Beispiel.

Weitere Infos

- So funktionieren Amazon SQS SQS-Warteschlangen im Amazon Simple Queue Service Developer Guide
- CreateQueuein der Amazon Simple Queue Service API-Referenz

- GetQueueUrlin der Amazon Simple Queue Service API-Referenz
- ListQueuesin der Amazon Simple Queue Service API-Referenz
- DeleteQueuesin der Amazon Simple Queue Service API-Referenz

Senden, Empfangen und Löschen von Amazon SQS SQS-Nachrichten

Nachrichten werden immer über eine SQS-Warteschlange zugestellt. Diese C++-Beispiele zeigen Ihnen, wie Sie Amazon SQS-Nachrichten aus SQS-Warteschlangen senden, empfangen und löschen können. AWS SDK für C++

Voraussetzungen

Bevor Sie beginnen, empfehlen wir Ihnen, Erste Schritte mit dem zu lesen. AWS SDK für C++

Laden Sie den Beispielcode herunter und erstellen Sie die Lösung wie unter beschriebenErste Schritte mit Codebeispielen.

Um die Beispiele ausführen zu können, muss das Benutzerprofil, das Ihr Code für die Anfragen verwendet, über die entsprechenden Berechtigungen verfügen AWS (für den Dienst und die Aktion). Weitere Informationen finden Sie unter Bereitstellen von AWS Anmeldeinformationen.

Senden einer Nachricht

Sie können einer Amazon SQS SQS-Warteschlange eine einzelne Nachricht hinzufügen, indem Sie die SQSClient `SendMessage` Klassenmitgliedsfunktion aufrufen. Sie stellen `SendMessage` ein SendMessageRequestObjekt bereit, das die URL der Warteschlange, den Nachrichtentext und einen optionalen Verzögerungswert (in Sekunden) enthält.

Beinhaltet

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

Code

```
    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;
    }
```

Siehe vollständiges Beispiel.

Empfangen von Nachrichten

Ruft alle Nachrichten ab, die sich derzeit in der Warteschlange befinden, indem Sie die SQSClient `ReceiveMessage` Klassenmitgliedsfunktion aufrufen und ihr die URL der Warteschlange übergeben. Nachrichten werden als Liste von Message-Objekten zurückgegeben.

Beinhaltet

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

Code

```
    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;
    }
```

Siehe vollständiges Beispiel.

Löschen von Nachrichten nach dem Empfangen

Nachdem Sie eine Nachricht empfangen und ihren Inhalt verarbeitet haben, löschen Sie die Nachricht aus der Warteschlange, indem Sie das Empfangs-Handle der Nachricht und die Warteschlangen-URL an die SQSClient `DeleteMessage` Klassenmitgliedsfunktion senden.

Beinhaltet

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

Code

```
    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;
    }
```

Siehe vollständiges Beispiel.

Weitere Infos

- So funktionieren Amazon SQS SQS-Warteschlangen im Amazon Simple Queue Service Developer Guide
- SendMessagein der Amazon Simple Queue Service API-Referenz
- SendMessageBatchin der Amazon Simple Queue Service API-Referenz
- ReceiveMessagein der Amazon Simple Queue Service API-Referenz
- DeleteMessagein der Amazon Simple Queue Service API-Referenz

Long Polling für Amazon SQS SQS-Nachrichtenwarteschlangen aktivieren

Amazon SQS verwendet standardmäßig kurze Abfragen, bei denen nur eine Teilmenge der Server abgefragt wird — basierend auf einer gewichteten Zufallsverteilung —, um festzustellen, ob Nachrichten für die Antwort verfügbar sind.

Lange Abfragen tragen dazu bei, Ihre Kosten für die Nutzung von Amazon SQS zu senken, indem die Anzahl der leeren Antworten reduziert wird, wenn keine Nachrichten als Antwort auf eine an eine Amazon SQS SQS-Warteschlange gesendete ReceiveMessage Anfrage zurückgesendet werden können, und falsche leere Antworten vermieden werden. Sie können eine Langabfragen-Häufigkeit von 1–20 Sekunden festlegen.

Voraussetzungen

Bevor Sie beginnen, empfehlen wir Ihnen, Erste Schritte mit dem zu lesen. AWS SDK für C++

Laden Sie den Beispielcode herunter und erstellen Sie die Lösung wie unter beschriebenErste Schritte mit Codebeispielen.

Um die Beispiele ausführen zu können, muss das Benutzerprofil, das Ihr Code für die Anfragen verwendet, über die entsprechenden Berechtigungen verfügen AWS (für den Dienst und die Aktion). Weitere Informationen finden Sie unter Bereitstellen von AWS Anmeldeinformationen.

Aktivieren Sie Long Polling beim Erstellen einer Warteschlange

Um lange Abfragen beim Erstellen einer Amazon SQS SQS-Warteschlange zu aktivieren, legen Sie das `ReceiveMessageWaitTimeSeconds` Attribut für das CreateQueueRequestObjekt fest, bevor Sie die Mitgliedsfunktion der SQSClient Klasse aufrufen. `CreateQueue`

Beinhaltet

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

Code

```
    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;
    }
```

Siehe vollständiges Beispiel.

Aktivieren der Langabfrage für eine vorhandene Warteschlange

Zusätzlich zur Aktivierung der langen Abfrage beim Erstellen einer Warteschlange können Sie diese Funktion auch in einer vorhandenen Warteschlange aktivieren, indem `ReceiveMessageWaitTimeSeconds` Sie die Funktion SetQueueAttributesRequestvor dem Aufrufen der Member-Funktion der SQSClient Klasse aktivieren`SetQueueAttributes`.

Beinhaltet

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

Code

```
    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;
    }
```

Siehe vollständiges Beispiel.

Aktivieren von Langabfragen beim Nachrichteneingang

Sie können lange Abfragen beim Empfang einer Nachricht aktivieren, indem Sie die Wartezeit in Sekunden für den Wert festlegen ReceiveMessageRequest, den Sie an die ReceiveMessage Mitgliedsfunktion der SQSClient Klasse übergeben.

> **Note**
>
> Sie sollten sicherstellen, dass das Anfrage-Timeout des AWS Clients größer ist als die maximale lange Abfragezeit (20 s), damit Ihre `ReceiveMessage` Anfragen beim Warten auf das nächste Umfrageereignis nicht zu einem Timeout kommen!

Beinhaltet

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

Code

```
    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;
```

```
    }
```

Siehe vollständiges Beispiel.

Weitere Infos

- Amazon SQS Long Polling im Amazon Simple Queue Service Developer Guide
- CreateQueuein der Amazon Simple Queue Service API-Referenz
- ReceiveMessagein der Amazon Simple Queue Service API-Referenz
- SetQueueAttributesin der Amazon Simple Queue Service API-Referenz

Einstellen des Sichtbarkeits-Timeouts in Amazon SQS

Wenn eine Nachricht in Amazon SQS empfangen wird, bleibt sie in der Warteschlange, bis sie gelöscht wird, um den Empfang sicherzustellen. Eine empfangene, aber nicht gelöschte Nachricht erscheint erst nach Ablauf einer bestimmten Zeitbeschränkung für die Sichtbarkeit in nachfolgenden Anforderungen. Dadurch wird gewährleistet, dass die Nachricht nicht mehrmals empfangen wird, bevor sie verarbeitet und gelöscht werden kann.

Bei Nutzung von Standard-Warteschlangen kann durch die Zeitbeschränkung für die Sichtbarkeit nicht garantiert werden, dass eine Nachricht mehrmals empfangen wird. Wenn Sie eine Standard-Warteschlange verwenden, achten Sie darauf, dass Ihr Code mit dem Fall umgehen kann, dass dieselbe Nachricht mehrmals eingeht.

Voraussetzungen

Bevor Sie beginnen, empfehlen wir Ihnen, Erste Schritte mit dem AWS SDK für C++ zu lesen.

Laden Sie den Beispielcode herunter und erstellen Sie die Lösung wie unter beschriebenErste Schritte mit Codebeispielen.

Um die Beispiele ausführen zu können, muss das Benutzerprofil, das Ihr Code für die Anfragen verwendet, über die entsprechenden Berechtigungen verfügen AWS (für den Dienst und die Aktion). Weitere Informationen finden Sie unter Bereitstellen von AWS Anmeldeinformationen.

Legen Sie das Timeout für die Nachrichtensichtbarkeit beim Empfang der Nachricht fest

Wenn Sie eine Nachricht erhalten haben, können Sie das Zeitlimit für die Sichtbarkeit ändern, indem Sie die Empfangsnummer in einer Nachricht übergeben ChangeMessageVisibilityRequest, die Sie an die Mitgliederfunktion der SQSClient Klasse übergeben. `ChangeMessageVisibility`

Beinhaltet

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

Code

```
    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;
    }
```

Siehe vollständiges Beispiel.

Weitere Infos

- Visibility Timeout im Amazon Simple Queue Service Developer Guide
- SetQueueAttributesin der Amazon Simple Queue Service API-Referenz
- GetQueueAttributesin der Amazon Simple Queue Service API-Referenz
- ReceiveMessagein der Amazon Simple Queue Service API-Referenz
- ChangeMessageVisibilityin der Amazon Simple Queue Service API-Referenz
- ChangeMessageVisibilityBatchin der Amazon Simple Queue Service API-Referenz

Verwenden von Warteschlangen für unzustellbare Nachrichten in Amazon SQS

Amazon SQS bietet Unterstützung für Warteschlangen mit unbestätigten Briefen. Eine Warteschlange für unzustellbare Nachrichten ist eine Warteschlange, die von anderen Warteschlangen für Nachrichten ausgewählt werden kann, die nicht erfolgreich verarbeitet werden können. Sie können diese Nachrichten in der Warteschlange für unzustellbare Nachrichten sammeln und isolieren, um zu bestimmen, warum die Verarbeitung fehlgeschlagen ist.

Um eine Warteschlange für unzustellbare Nachrichten zu erstellen, müssen Sie zuerst eine Redrive-Richtlinie erstellen und diese dann in den Attributen der Warteschlange festlegen.

> **Important**
>
> Bei einer Warteschlange für unzustellbare Nachrichten muss es sich um dieselbe Art von Warteschlange (FIFO oder Standard) handeln wie bei der Quellwarteschlange. Sie muss außerdem mit demselben AWS-Konto und AWS-Region wie die Quellwarteschlange erstellt werden.

Voraussetzungen

Bevor Sie beginnen, empfehlen wir Ihnen, Erste Schritte mit dem zu lesen AWS SDK für C++.

Laden Sie den Beispielcode herunter und erstellen Sie die Lösung wie unter beschriebenErste Schritte mit Codebeispielen.

Um die Beispiele ausführen zu können, muss das Benutzerprofil, das Ihr Code für die Anfragen verwendet, über die entsprechenden Berechtigungen verfügen AWS (für den Dienst und die Aktion). Weitere Informationen finden Sie unter Bereitstellen von AWS Anmeldeinformationen.

Erstellen Sie eine Redrive-Richtlinie

Eine Redrive-Richtlinie ist in JSON angegeben. Um sie zu erstellen, können Sie die JSON-Utility-Klasse verwenden, die im AWS SDK für C++ Lieferumfang von enthalten ist.

Hier ist eine Beispielfunktion, die eine Redrive-Richtlinie erstellt, indem sie ihr den ARN Ihrer Warteschlange für unzustellbare Briefe und die maximale Häufigkeit, mit der die Nachricht empfangen und nicht verarbeitet werden kann, zur Verfügung stellt, bevor sie an die Warteschlange für unzustellbare Briefe gesendet wird.

Beinhaltet

```
#include <aws/core/Aws.h>
#include <aws/core/utils/json/JsonSerializer.h>
```

Code

```
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();
}
```

Siehe vollständiges Beispiel.

Legen Sie die Redrive-Richtlinie für Ihre Quellwarteschlange fest

Um die Einrichtung Ihrer Warteschlange für unzustellbare Briefe abzuschließen, rufen Sie die SQSClient `SetQueueAttributes` Mitgliederfunktion der Klasse mit einem SetQueueAttributesRequestObjekt auf, für das Sie das `RedrivePolicy` Attribut mit Ihrer JSON-Redrive-Richtlinie festgelegt haben.

Beinhaltet

```
#include <aws/sqs/SQSClient.h>
#include <aws/sqs/model/SetQueueAttributesRequest.h>
#include <iostream>
```

Code

```
    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;
    }
```

Siehe vollständiges Beispiel.

Weitere Infos

- Verwenden von Amazon SQS Dead Letter Queues im Amazon Simple Queue Service Developer Guide
- SetQueueAttributesin der Amazon Simple Queue Service API-Referenz

SDK for C++ C++-Codebeispiele

Die Codebeispiele in diesem Thema zeigen Ihnen, wie Sie AWS SDK für C++ with verwenden AWS.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Services aufrufen oder mit anderen AWS-Services kombinieren.

Einige Services enthalten zusätzliche Beispielkategorien, die zeigen, wie Bibliotheken oder dienstspezifische Funktionen genutzt werden können.

Dienstleistungen

- ACM-Beispiele unter Verwendung von SDK für C++
- API-Gateway-Beispiele unter Verwendung von SDK für C++
- Aurora-Beispiele unter Verwendung von SDK für C++
- Auto-Scaling-Beispiele unter Verwendung von SDK für C++
- CloudTrail Beispiele mit SDK for C++
- CloudWatch Beispiele mit SDK for C++
- CloudWatch Log-Beispiele mit SDK for C++
- CodeBuild Beispiele mit SDK for C++
- Beispiele für Amazon Cognito Identity Provider unter Verwendung von SDK für C++
- DynamoDB-Beispiele unter Verwendung von SDK für C++
- EC2 Amazon-Beispiele mit SDK for C++
- EventBridge Beispiele mit SDK for C++
- AWS Glue Beispiele mit SDK for C++
- HealthImaging Beispiele mit SDK for C++
- IAM-Beispiele unter Verwendung von SDK für C++
- AWS IoT Beispiele mit SDK for C++

- AWS IoT data Beispiele mit SDK for C++
- Lambda-Beispiele unter Verwendung von SDK für C++
- MediaConvert Beispiele mit SDK for C++
- Amazon-RDS-Beispiele unter Verwendung von SDK für C++
- Beispiele für Amazon RDS Data Service unter Verwendung von SDK für C++
- Amazon-Rekognition-Beispiele unter Verwendung von SDK für C++
- Amazon-S3-Beispiele unter Verwendung von SDK für C++
- Secrets-Manager-Beispiele unter Verwendung von SDK für C++
- Amazon-SES-Beispiele unter Verwendung von SDK für C++
- Amazon-SNS-Beispiele unter Verwendung von SDK für C++
- Amazon-SQS-Beispiele unter Verwendung von SDK für C++
- AWS STS Beispiele mit SDK for C++
- Beispiele für Amazon Transcribe Streaming unter Verwendung von SDK für C++

ACM-Beispiele unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für C++ mit ACM Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`AddTagsToCertificate`

Das folgende Codebeispiel zeigt, wie Sie`AddTagsToCertificate`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie AddTagsToCertificatein der AWS SDK für C++ API-Referenz.

`DeleteCertificate`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCertificate`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie DeleteCertificatein der AWS SDK für C++ API-Referenz.

`DescribeCertificate`

Das folgende Codebeispiel zeigt die Verwendung`DescribeCertificate`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie DescribeCertificatein der AWS SDK für C++ API-Referenz.

`ExportCertificate`

Das folgende Codebeispiel zeigt die Verwendung`ExportCertificate`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie ExportCertificatein der AWS SDK für C++ API-Referenz.

`GetCertificate`

Das folgende Codebeispiel zeigt die VerwendungGetCertificate.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie GetCertificatein der AWS SDK für C++ API-Referenz.

`ImportCertificate`

Das folgende Codebeispiel zeigt die Verwendung`ImportCertificate`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie ImportCertificatein der AWS SDK für C++ API-Referenz.

`ListCertificates`

Das folgende Codebeispiel zeigt die Verwendung`ListCertificates`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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;
}
```

- Einzelheiten zur API finden Sie ListCertificatesin der AWS SDK für C++ API-Referenz.

`ListTagsForCertificate`

Das folgende Codebeispiel zeigt die Verwendung`ListTagsForCertificate`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie ListTagsForCertificatein der AWS SDK für C++ API-Referenz.

`RemoveTagsFromCertificate`

Das folgende Codebeispiel zeigt die Verwendung`RemoveTagsFromCertificate`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie RemoveTagsFromCertificatein der AWS SDK für C++ API-Referenz.

`RenewCertificate`

Das folgende Codebeispiel zeigt die Verwendung`RenewCertificate`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie RenewCertificatein der AWS SDK für C++ API-Referenz.

`RequestCertificate`

Das folgende Codebeispiel zeigt die Verwendung`RequestCertificate`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie RequestCertificatein der AWS SDK für C++ API-Referenz.

`ResendValidationEmail`

Das folgende Codebeispiel zeigt die Verwendung`ResendValidationEmail`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie ResendValidationEmailin der AWS SDK für C++ API-Referenz.

`UpdateCertificateOptions`

Das folgende Codebeispiel zeigt die VerwendungUpdateCertificateOptions.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie UpdateCertificateOptionsin der AWS SDK für C++ API-Referenz.

API-Gateway-Beispiele unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für C++ with API Gateway Aktionen ausführen und allgemeine Szenarien implementieren.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für C++

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB

- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Aurora-Beispiele unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für C++ mit Aurora Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien

Erste Schritte

Hello Aurora

Das folgenden Codebeispiel veranschaulicht die ersten Schritte mit Aurora.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMake Datei CMake Lists.txt.

```
# 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})
```

Code für die Quelldatei „hello_aurora.cpp“.

```
#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;
}
```

- Einzelheiten zur API finden Sie unter Describe DBClusters in der AWS SDK für C++ API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine benutzerdefinierte Cluster-Parametergruppe von Aurora DB und legen Sie Parameterwerte fest.
- Erstellen Sie einen DB-Cluster, der die Parametergruppe verwendet.
- Erstellen Sie eine DB-Instance, die eine Datenbank enthält.
- Erstellen Sie einen Snapshot des DB-Clusters und bereinigen Sie dann die Ressourcen.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - CreateDBCluster
  - CreateDBClusterParameterGroup
  - DBClusterSchnappschuss erstellen
  - CreateDBInstance
  - LöschenDBCluster
  - LöschenDBClusterParameterGroup
  - LöschenDBInstance
  - Beschreiben DBCluster ParameterGroups
  - Beschreiben Sie die DBCluster Parameter
  - Beschreiben Sie DBCluster Schnappschüsse
  - Beschreiben DBClusters

- DBEngineVersionen beschreiben
- Beschreiben DBInstances
- DescribeOrderableDBInstanceOptionen
- Modifizieren SieDBClusterParameterGroup

Aktionen

`CreateDBCluster`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBCluster`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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;
        }
```

- Einzelheiten zur API finden Sie unter DBCluster In der AWS SDK für C++ API-Referenz erstellen.

`CreateDBClusterParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBClusterParameterGroup`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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;
        }
```

- Einzelheiten zur API finden Sie unter DBCluster ParameterGroup In der AWS SDK für C++ API-Referenz erstellen.

`CreateDBClusterSnapshot`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBClusterSnapshot`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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;
            }
```

- Einzelheiten zur API finden Sie unter DBClusterSnapshot erstellen in der AWS SDK für C++ API-Referenz.

`CreateDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBInstance`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
        }
```

- Einzelheiten zur API finden Sie unter DBInstance In der AWS SDK für C++ API-Referenz erstellen.

`DeleteDBCluster`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBCluster`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
        }
```

- Einzelheiten zur API finden Sie unter Löschen DBCluster in der AWS SDK für C++ API-Referenz.

`DeleteDBClusterParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBClusterParameterGroup`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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;
        }
```

- Einzelheiten zur API finden Sie unter Löschen DBCluster ParameterGroup in der AWS SDK für C++ API-Referenz.

`DeleteDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBInstance`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
            }
```

- Einzelheiten zur API finden Sie unter Löschen DBInstance in der AWS SDK für C++ API-Referenz.

`DescribeDBClusterParameterGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusterParameterGroups`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
        }
```

- Einzelheiten zur API finden Sie unter Describe DBCluster ParameterGroups in der AWS SDK für C++ API-Referenz.

`DescribeDBClusterParameters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusterParameters`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
}
```

- Einzelheiten zur API finden Sie unter Describe DBCluster Parameters in der AWS SDK für C++ API-Referenz.

`DescribeDBClusterSnapshots`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusterSnapshots`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
            }
```

- Einzelheiten zur API finden Sie unter Beschreiben von DBCluster Snapshots in der AWS SDK für C++ API-Referenz.

`DescribeDBClusters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBClusters`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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;

}
```

- Einzelheiten zur API finden Sie unter Describe DBClusters in der AWS SDK für C++ API-Referenz.

`DescribeDBEngineVersions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBEngineVersions`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
}
```

- Einzelheiten zur API finden Sie unter DBEngineVersionen beschreiben in der AWS SDK für C++ API-Referenz.

`DescribeDBInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBInstances`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
}
```

- Einzelheiten zur API finden Sie unter Describe DBInstances in der AWS SDK für C++ API-Referenz.

`DescribeOrderableDBInstanceOptions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeOrderableDBInstanceOptions`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
}
```

- Einzelheiten zur API finden Sie unter DescribeOrderableDBInstanceOptionen in der AWS SDK für C++ API-Referenz.

`ModifyDBClusterParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`ModifyDBClusterParameterGroup`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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;
    }
```

- Einzelheiten zur API finden Sie unter Ändern DBCluster ParameterGroup in der AWS SDK für C ++ API-Referenz.

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für C++

Zeigt, wie eine Webanwendung erstellt wird, die in einer Datenbank von Amazon Aurora Serverless gespeicherte Arbeitselemente verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer C++-REST-API, die Amazon Aurora Aurora-Serverless-Daten abfragt und von einer React-Anwendung verwendet werden kann, finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Auto-Scaling-Beispiele unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für C++ mit Auto Scaling Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello Auto Scaling

Das folgende Codebeispiel zeigt die ersten Schritte mit Auto Scaling.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMake Datei CMake Lists.txt.

```
# 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})
```

Code für die Quelldatei „hello_autoscaling.cpp“.

```
#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;
}
```

- Einzelheiten zur API finden Sie DescribeAutoScalingGroupsunter AWS SDK für C++ API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine Amazon EC2 Auto Scaling Scaling-Gruppe mit einer Startvorlage und Availability Zones und erhalten Sie Informationen über laufende Instances.
- Aktivieren Sie die Erfassung von CloudWatch Amazon-Metriken.
- Aktualisieren der gewünschten Kapazität der Gruppe und Warten auf den Start einer Instance
- Beenden Sie eine Instance in der Gruppe.
- Listen Sie Skalierungsaktivitäten auf, die als Reaktion auf Benutzeranfragen und Kapazitätsänderungen erfolgen.
- Holen Sie sich Statistiken für CloudWatch Metriken und bereinigen Sie dann Ressourcen.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - CreateAutoScalingGroup
  - DeleteAutoScalingGroup
  - DescribeAutoScalingGroups
  - DescribeAutoScalingInstances
  - DescribeScalingActivities
  - DisableMetricsCollection
  - EnableMetricsCollection
  - SetDesiredCapacity
  - TerminateInstanceInAutoScalingGroup
  - UpdateAutoScalingGroup

Aktionen

`CreateAutoScalingGroup`

Das folgende Codebeispiel zeigt, wie man es benutzt`CreateAutoScalingGroup`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;

        }
```

- Einzelheiten zur API finden Sie CreateAutoScalingGroupin der AWS SDK für C++ API-Referenz.

`DeleteAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAutoScalingGroup`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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;
        }
    }
```

- Einzelheiten zur API finden Sie DeleteAutoScalingGroupin der AWS SDK für C++ API-Referenz.

`DescribeAutoScalingGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAutoScalingGroups`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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;
    }
```

- Einzelheiten zur API finden Sie DescribeAutoScalingGroupsin der AWS SDK für C++ API-Referenz.

`DescribeAutoScalingInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAutoScalingInstances`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
        }
```

- Einzelheiten zur API finden Sie DescribeAutoScalingInstancesin der AWS SDK für C++ API-Referenz.

`DescribeScalingActivities`

Das folgende Codebeispiel zeigt die Verwendung`DescribeScalingActivities`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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;
        }
```

- Einzelheiten zur API finden Sie DescribeScalingActivitiesin der AWS SDK für C++ API-Referenz.

`DisableMetricsCollection`

Das folgende Codebeispiel zeigt die Verwendung`DisableMetricsCollection`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;

        }
```

- Einzelheiten zur API finden Sie DisableMetricsCollectionin der AWS SDK für C++ API-Referenz.

`EnableMetricsCollection`

Das folgende Codebeispiel zeigt die Verwendung`EnableMetricsCollection`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;

        }
```

- Einzelheiten zur API finden Sie EnableMetricsCollectionin der AWS SDK für C++ API-Referenz.

`SetDesiredCapacity`

Das folgende Codebeispiel zeigt die Verwendung`SetDesiredCapacity`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;

        }
```

- Einzelheiten zur API finden Sie SetDesiredCapacityin der AWS SDK für C++ API-Referenz.

`TerminateInstanceInAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`TerminateInstanceInAutoScalingGroup`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;

        }
```

- Einzelheiten zur API finden Sie TerminateInstanceInAutoScalingGroupin der AWS SDK für C++ API-Referenz.

`UpdateAutoScalingGroup`

Das folgende Codebeispiel zeigt die Verwendung`UpdateAutoScalingGroup`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;

        }
```

- Einzelheiten zur API finden Sie UpdateAutoScalingGroupin der AWS SDK für C++ API-Referenz.

CloudTrail Beispiele mit SDK for C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für C++ with Aktionen ausführen und allgemeine Szenarien implementieren CloudTrail.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateTrail`

Das folgende Codebeispiel zeigt, wie Sie`CreateTrail`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie CreateTrailin der AWS SDK für C++ API-Referenz.

`DeleteTrail`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTrail`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie DeleteTrailin der AWS SDK für C++ API-Referenz.

`DescribeTrail`

Das folgende Codebeispiel zeigt die Verwendung`DescribeTrail`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie DescribeTrailin der AWS SDK für C++ API-Referenz.

`LookupEvents`

Das folgende Codebeispiel zeigt die Verwendung`LookupEvents`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// 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;
}
```

- Einzelheiten zur API finden Sie LookupEventsin der AWS SDK für C++ API-Referenz.

CloudWatch Beispiele mit SDK for C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für C++ with Aktionen ausführen und allgemeine Szenarien implementieren CloudWatch.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`DeleteAlarms`

Das folgende Codebeispiel zeigt, wie Sie`DeleteAlarms`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

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

Löschen Sie den Alarm.

```
        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;
        }
```

- Einzelheiten zur API finden Sie DeleteAlarmsin der AWS SDK für C++ API-Referenz.

`DescribeAlarmsForMetric`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAlarmsForMetric`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#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>
```

Beschreiben Sie die Alarme.

```
        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();
        }
```

- Einzelheiten zur API finden Sie DescribeAlarmsForMetricin der AWS SDK für C++ API-Referenz.

`DisableAlarmActions`

Das folgende Codebeispiel zeigt die Verwendung`DisableAlarmActions`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

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

Deaktivieren Sie die Alarmaktionen.

```
        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;
        }
```

- Einzelheiten zur API finden Sie DisableAlarmActionsin der AWS SDK für C++ API-Referenz.

`EnableAlarmActions`

Das folgende Codebeispiel zeigt die Verwendung`EnableAlarmActions`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

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

Aktivieren Sie die Alarmaktionen.

```
    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;
```

- Einzelheiten zur API finden Sie EnableAlarmActionsin der AWS SDK für C++ API-Referenz.

`ListMetrics`

Das folgende Codebeispiel zeigt die Verwendung`ListMetrics`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#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>
```

Listen Sie die Metriken auf.

```
        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();
    }
```

- Einzelheiten zur API finden Sie ListMetricsin der AWS SDK für C++ API-Referenz.

`PutMetricAlarm`

Das folgende Codebeispiel zeigt die Verwendung`PutMetricAlarm`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

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

Erstellen Sie den Alarm, um die Metrik zu beobachten.

```
        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;
        }
```

- Einzelheiten zur API finden Sie PutMetricAlarmin der AWS SDK für C++ API-Referenz.

`PutMetricData`

Das folgende Codebeispiel zeigt die Verwendung`PutMetricData`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

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

Fügen Sie Daten in eine Metrik ein.

```
        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;
        }
```

- Einzelheiten zur API finden Sie PutMetricDatain der AWS SDK für C++ API-Referenz.

CloudWatch Log-Beispiele mit SDK for C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für C++ with CloudWatch Logs Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`DeleteSubscriptionFilter`

Das folgende Codebeispiel zeigt, wie Sie`DeleteSubscriptionFilter`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>
#include <aws/core/utils/Outcome.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/DeleteSubscriptionFilterRequest.h>
#include <iostream>
```

Löschen Sie den Abonnementfilter.

```
        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;
        }
```

- Einzelheiten zur API finden Sie DeleteSubscriptionFilterin der AWS SDK für C++ API-Referenz.

`DescribeSubscriptionFilters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeSubscriptionFilters`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#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>
```

Listen Sie die Abonnementfilter auf.

```
        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();
        }
```

- Einzelheiten zur API finden Sie DescribeSubscriptionFiltersin der AWS SDK für C++ API-Referenz.

`PutSubscriptionFilter`

Das folgende Codebeispiel zeigt die Verwendung`PutSubscriptionFilter`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#include <aws/core/Aws.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/PutSubscriptionFilterRequest.h>
#include <aws/core/utils/Outcome.h>
#include <iostream>
```

Erstellen Sie den Abonnementfilter.

```
        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;
        }
```

- Einzelheiten zur API finden Sie PutSubscriptionFilterin der AWS SDK für C++ API-Referenz.

CodeBuild Beispiele mit SDK for C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für C++ with Aktionen ausführen und allgemeine Szenarien implementieren CodeBuild.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`ListBuilds`

Das folgende Codebeispiel zeigt, wie Sie`ListBuilds`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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;
}
```

- Einzelheiten zur API finden Sie ListBuildsin der AWS SDK für C++ API-Referenz.

`ListProjects`

Das folgende Codebeispiel zeigt die Verwendung`ListProjects`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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;
}
```

- Einzelheiten zur API finden Sie ListProjectsin der AWS SDK für C++ API-Referenz.

`StartBuild`

Das folgende Codebeispiel zeigt die Verwendung`StartBuild`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie StartBuildin der AWS SDK für C++ API-Referenz.

Beispiele für Amazon Cognito Identity Provider unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Cognito Identity Provider Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für C++

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hello Amazon Cognito

Die folgenden Codebeispiele veranschaulichen die ersten Schritte mit Amazon Cognito.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMake Datei CMake Lists.txt.

```
# 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})
```

Code für die Quelldatei hello_cognito.cpp.

```
#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;
}
```

- Einzelheiten zur API finden Sie ListUserPoolsunter AWS SDK für C++ API-Referenz.

Aktionen

`AdminGetUser`

Das folgende Codebeispiel zeigt die Verwendung`AdminGetUser`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
    }
```

- Einzelheiten zur API finden Sie AdminGetUserin der AWS SDK für C++ API-Referenz.

`AdminInitiateAuth`

Das folgende Codebeispiel zeigt die VerwendungAdminInitiateAuth.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
   }
```

- Einzelheiten zur API finden Sie AdminInitiateAuthin der AWS SDK für C++ API-Referenz.

`AdminRespondToAuthChallenge`

Das folgende Codebeispiel zeigt die VerwendungAdminRespondToAuthChallenge.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
       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;
        }
```

- Einzelheiten zur API finden Sie AdminRespondToAuthChallengein der AWS SDK für C++ API-Referenz.

`AssociateSoftwareToken`

Das folgende Codebeispiel zeigt die Verwendung`AssociateSoftwareToken`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
        }
```

- Einzelheiten zur API finden Sie AssociateSoftwareTokenin der AWS SDK für C++ API-Referenz.

`ConfirmSignUp`

Das folgende Codebeispiel zeigt die Verwendung`ConfirmSignUp`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
        }
```

- Einzelheiten zur API finden Sie ConfirmSignUpin der AWS SDK für C++ API-Referenz.

`DeleteUser`

Das folgende Codebeispiel zeigt die Verwendung`DeleteUser`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
```

```
        }
```

- Einzelheiten zur API finden Sie DeleteUserin der AWS SDK für C++ API-Referenz.

`ResendConfirmationCode`

Das folgende Codebeispiel zeigt die Verwendung`ResendConfirmationCode`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
       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;
        }
```

- Einzelheiten zur API finden Sie ResendConfirmationCodein der AWS SDK für C++ API-Referenz.

`SignUp`

Das folgende Codebeispiel zeigt die VerwendungSignUp.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
        }
```

- Einzelheiten zur API finden Sie SignUpin der AWS SDK für C++ API-Referenz.

`VerifySoftwareToken`

Das folgende Codebeispiel zeigt die Verwendung`VerifySoftwareToken`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
    }
```

- Einzelheiten zur API finden Sie VerifySoftwareTokenin der AWS SDK für C++ API-Referenz.

Szenarien

Registrieren eines Benutzers bei einem Benutzerpool, der MFA erfordert

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Registrieren Sie einen Benutzer mit einem Benutzernamen, einem Passwort und einer E-Mail-Adresse und bestätigen Sie ihn.
- Richten Sie die Multi-Faktor-Authentifizierung durch Zuordnung einer MFA-Anwendung zu dem Benutzer ein.
- Melden Sie sich unter Verwendung eines Passworts und eines MFA-Codes an.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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();
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

DynamoDB-Beispiele unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für C++ mit DynamoDB Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien

Erste Schritte

Hello DynamoDB

Das folgenden Codebeispiel veranschaulicht die ersten Schritte mit DynamoDB.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMake Datei CMake Lists.txt.

```
# 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})
```

Code für die Quelldatei „hello_dynamodb.cpp“.

```
#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;
}
```

- Einzelheiten zur API finden Sie ListTablesunter AWS SDK für C++ API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen einer Tabelle, die Filmdaten enthalten kann.
- Einfügen, Abrufen und Aktualisieren eines einzelnen Films in der Tabelle.
- Schreiben von Filmdaten in die Tabelle anhand einer JSON-Beispieldatei.
- Abfragen nach Filmen, die in einem bestimmten Jahr veröffentlicht wurden.
- Scan nach Filmen, die in mehreren Jahren veröffentlicht wurden.
- Löschen eines Films aus der Tabelle und anschließendes Löschen der Tabelle.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    {
        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;
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem

- PutItem
- Query
- Scan
- UpdateItem

Aktionen

`BatchExecuteStatement`

Das folgende Codebeispiel zeigt, wie man es benutzt`BatchExecuteStatement`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie Stapel von INSERT-Anweisungen, um Elemente hinzuzufügen.

```
    // 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;
        }
    }
```

Verwenden Sie Stapel von SELECT-Anweisungen, um Elemente abzurufen.

```
    // 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;
        }
    }
```

Verwenden Sie Stapel von UPDATE-Anweisungen, um Elemente zu aktualisieren.

```
    // 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;
       }
   }
```

Verwenden Sie Stapel von DELETE-Anweisungen, um Elemente zu löschen.

```
    // 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;
        }
    }
```

- Einzelheiten zur API finden Sie BatchExecuteStatementin der AWS SDK für C++ API-Referenz.

`BatchGetItem`

Das folgende Codebeispiel zeigt die Verwendung`BatchGetItem`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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;
}
```

- Einzelheiten zur API finden Sie BatchGetItemin der AWS SDK für C++ API-Referenz.

`BatchWriteItem`

Das folgende Codebeispiel zeigt die Verwendung`BatchWriteItem`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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;
}
```

- Einzelheiten zur API finden Sie BatchWriteItemin der AWS SDK für C++ API-Referenz.

`CreateTable`

Das folgende Codebeispiel zeigt die Verwendung`CreateTable`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

Code, der darauf wartet, dass die Tabelle aktiv wird.

```
//! 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;
}
```

- Einzelheiten zur API finden Sie CreateTablein der AWS SDK für C++ API-Referenz.

`DeleteItem`

Das folgende Codebeispiel zeigt die Verwendung`DeleteItem`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

Code, der darauf wartet, dass die Tabelle aktiv wird.

```
//! 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;
}
```

- Einzelheiten zur API finden Sie DeleteItemin der AWS SDK für C++ API-Referenz.

`DeleteTable`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTable`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie DeleteTablein der AWS SDK für C++ API-Referenz.

`DescribeTable`

Das folgende Codebeispiel zeigt die Verwendung`DescribeTable`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie DescribeTablein der AWS SDK für C++ API-Referenz.

`ExecuteStatement`

Das folgende Codebeispiel zeigt die Verwendung`ExecuteStatement`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Verwenden Sie eine INSERT-Anweisung, um ein Element hinzuzufügen.

```
    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;
        }
    }
```

Verwenden Sie eine SELECT-Anweisung, um ein Element abzurufen.

```
    //  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;
            }
        }
    }
```

Verwenden Sie eine UPDATE-Anweisung, um ein Element zu aktualisieren.

```
    //  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;
        }
    }
```

Verwenden Sie eine DELETE-Anweisung, um ein Element zu löschen.

```
    // 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;
        }
    }
```

- Einzelheiten zur API finden Sie ExecuteStatementin der AWS SDK für C++ API-Referenz.

`GetItem`

Das folgende Codebeispiel zeigt die Verwendung`GetItem`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie GetItemin der AWS SDK für C++ API-Referenz.

`ListTables`

Das folgende Codebeispiel zeigt die Verwendung`ListTables`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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;
}
```

- Einzelheiten zur API finden Sie ListTablesin der AWS SDK für C++ API-Referenz.

`PutItem`

Das folgende Codebeispiel zeigt die Verwendung`PutItem`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

Code, der darauf wartet, dass die Tabelle aktiv wird.

```
//! 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;
}
```

- Einzelheiten zur API finden Sie PutItemin der AWS SDK für C++ API-Referenz.

`Query`

Das folgende Codebeispiel zeigt die Verwendung`Query`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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;
}
```

- Weitere API-Informationen finden Sie unter Abfragen in der AWS SDK für C++ -API-Referenz.

Scan

Das folgende Codebeispiel zeigt, wie man es benutztScan.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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;
```

```
}
```

- Weitere API-Informationen finden Sie unter Scan in der AWS SDK für C++ -API-Referenz.

`UpdateItem`

Das folgende Codebeispiel zeigt, wie man es benutzt`UpdateItem`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

Code, der darauf wartet, dass die Tabelle aktiv wird.

```
//! 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;
}
```

- Einzelheiten zur API finden Sie UpdateItemin der AWS SDK für C++ API-Referenz.

`UpdateTable`

Das folgende Codebeispiel zeigt die VerwendungUpdateTable.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

Code, der darauf wartet, dass die Tabelle aktiv wird.

```
//! 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;
}
```

- Einzelheiten zur API finden Sie UpdateTablein der AWS SDK für C++ API-Referenz.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für C++

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Abfragen einer Tabelle mithilfe von Stapeln von PartiQL-Anweisungen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Abrufen eines Stapels von Elementen mithilfe mehrerer SELECT-Anweisungen.
- Hinzufügen eines Stapels von Elementen hinzu, indem mehrere INSERT-Anweisungen ausgeführt werden.
- Aktualisieren eines Stapels von Elementen mithilfe mehrerer UPDATE-Anweisungen.
- Löschen eines Stapels von Elementen mithilfe mehrerer DELETE-Anweisungen.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
}
```

- Einzelheiten zur API finden Sie BatchExecuteStatementin der AWS SDK für C++ API-Referenz.

Abfragen einer Tabelle mit PartiQL

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Abrufen eines Elementes durch Ausführen einer SELECT-Anweisung.
- Hinzufügen eines Elementes durch Ausführung einer INSERT-Anweisung.
- Aktualisieren eines Elementes durch Ausführung einer UPDATE-Anweisung.
- Löschen eines Elementes durch Ausführung einer DELETE-Anweisung.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        //  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;
}
```

- Einzelheiten zur API finden Sie ExecuteStatementin der AWS SDK für C++ API-Referenz.

EC2 Amazon-Beispiele mit SDK for C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für C++ mit Amazon Aktionen ausführen und allgemeine Szenarien implementieren EC2.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen

Erste Schritte

Hallo Amazon EC2

Das folgende Codebeispiel zeigt, wie Sie mit Amazon beginnen können EC2.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMake Datei CMake Lists.txt.

```
# 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})
```

Code für die Quelldatei hello_ec2.cpp.

```
#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;
}
```

- Einzelheiten zur API finden Sie DescribeSecurityGroupsunter AWS SDK für C++ API-Referenz.

Aktionen

`AllocateAddress`

Das folgende Codebeispiel zeigt die Verwendung`AllocateAddress`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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;
}
```

- Einzelheiten zur API finden Sie AllocateAddressin der AWS SDK für C++ API-Referenz.

`AssociateAddress`

Das folgende Codebeispiel zeigt die Verwendung`AssociateAddress`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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();
}
```

- Einzelheiten zur API finden Sie AssociateAddressin der AWS SDK für C++ API-Referenz.

`AuthorizeSecurityGroupIngress`

Das folgende Codebeispiel zeigt die Verwendung`AuthorizeSecurityGroupIngress`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

Hilfsfunktion zum Erstellen einer Eingangsregel.

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

- Einzelheiten zur API finden Sie AuthorizeSecurityGroupIngressin der AWS SDK für C++ API-Referenz.

`CreateKeyPair`

Das folgende Codebeispiel zeigt die Verwendung`CreateKeyPair`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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();

}
```

- Einzelheiten zur API finden Sie CreateKeyPairin der AWS SDK für C++ API-Referenz.

`CreateSecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateSecurityGroup`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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;
}
```

- Einzelheiten zur API finden Sie CreateSecurityGroupin der AWS SDK für C++ API-Referenz.

`CreateTags`

Das folgende Codebeispiel zeigt die Verwendung`CreateTags`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie CreateTagsin der AWS SDK für C++ API-Referenz.

`DeleteKeyPair`

Das folgende Codebeispiel zeigt die Verwendung`DeleteKeyPair`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

```
}
```

- Einzelheiten zur API finden Sie DeleteKeyPairin der AWS SDK für C++ API-Referenz.

`DeleteSecurityGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteSecurityGroup`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

```
}
```

- Einzelheiten zur API finden Sie DeleteSecurityGroupin der AWS SDK für C++ API-Referenz.

`DescribeAddresses`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAddresses`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie DescribeAddressesin der AWS SDK für C++ API-Referenz.

`DescribeAvailabilityZones`

Das folgende Codebeispiel zeigt die Verwendung`DescribeAvailabilityZones`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie DescribeAvailabilityZonesin der AWS SDK für C++ API-Referenz.

`DescribeInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeInstances`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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;
}
```

- Einzelheiten zur API finden Sie DescribeInstancesin der AWS SDK für C++ API-Referenz.

`DescribeKeyPairs`

Das folgende Codebeispiel zeigt die Verwendung`DescribeKeyPairs`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie DescribeKeyPairsin der AWS SDK für C++ API-Referenz.

`DescribeRegions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeRegions`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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();

}
```

- Einzelheiten zur API finden Sie DescribeRegionsin der AWS SDK für C++ API-Referenz.

`DescribeSecurityGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeSecurityGroups`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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;
}
```

- Einzelheiten zur API finden Sie DescribeSecurityGroupsin der AWS SDK für C++ API-Referenz.

`MonitorInstances`

Das folgende Codebeispiel zeigt die Verwendung`MonitorInstances`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie MonitorInstancesin der AWS SDK für C++ API-Referenz.

`RebootInstances`

Das folgende Codebeispiel zeigt die Verwendung`RebootInstances`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie RebootInstancesin der AWS SDK für C++ API-Referenz.

`ReleaseAddress`

Das folgende Codebeispiel zeigt die Verwendung`ReleaseAddress`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie ReleaseAddressin der AWS SDK für C++ API-Referenz.

`RunInstances`

Das folgende Codebeispiel zeigt die Verwendung`RunInstances`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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;
}
```

- Einzelheiten zur API finden Sie RunInstancesin der AWS SDK für C++ API-Referenz.

`StartInstances`

Das folgende Codebeispiel zeigt die Verwendung`StartInstances`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie StartInstancesin der AWS SDK für C++ API-Referenz.

`StopInstances`

Das folgende Codebeispiel zeigt die Verwendung`StopInstances`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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;
}
```

- Einzelheiten zur API finden Sie StopInstancesin der AWS SDK für C++ API-Referenz.

`TerminateInstances`

Das folgende Codebeispiel zeigt die Verwendung`TerminateInstances`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie TerminateInstancesin der AWS SDK für C++ API-Referenz.

`UnmonitorInstances`

Das folgende Codebeispiel zeigt die Verwendung`UnmonitorInstances`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie UnmonitorInstancesin der AWS SDK für C++ API-Referenz.

EventBridge Beispiele mit SDK for C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für C++ with Aktionen ausführen und allgemeine Szenarien implementieren EventBridge.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`PutEvents`

Das folgende Codebeispiel zeigt, wie Sie`PutEvents`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#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>
```

Senden Sie das Ereignis.

```
        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;
        }
```

- Einzelheiten zur API finden Sie PutEventsin der AWS SDK für C++ API-Referenz.

`PutRule`

Das folgende Codebeispiel zeigt die Verwendung`PutRule`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#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>
```

Erstellen Sie die Regel.

```
        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;
        }
```

- Einzelheiten zur API finden Sie PutRulein der AWS SDK für C++ API-Referenz.

`PutTargets`

Das folgende Codebeispiel zeigt die Verwendung`PutTargets`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Binden Sie die erforderlichen Dateien ein.

```
#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>
```

Fügen Sie das Ziel hinzu.

```
        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;
        }
```

- Einzelheiten zur API finden Sie PutTargetsin der AWS SDK für C++ API-Referenz.

AWS Glue Beispiele mit SDK for C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für C++ with Aktionen ausführen und allgemeine Szenarien implementieren AWS Glue.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo AWS Glue

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS Glue beginnen.

SDK für C++

> **Note**
>
> Da ist noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMake Datei CMake Lists.txt.

```
# 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})
```

Code für die Quelldatei hello_glue.cpp.

```
#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;
}
```

- Einzelheiten zur API finden Sie ListJobsunter AWS SDK für C++ API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Crawler, der einen öffentlichen Amazon-S3-Bucket crawlt und eine Datenbank mit CSV-formatierten Metadaten generiert.
- Führen Sie Informationen zu Datenbanken und Tabellen in Ihrem auf AWS Glue Data Catalog.
- Erstellen Sie einen Auftrag, um CSV-Daten aus dem S3-Bucket zu extrahieren, die Daten umzuwandeln und die JSON-formatierte Ausgabe in einen anderen S3-Bucket zu laden.
- Listen Sie Informationen zu Auftragsausführungen auf, zeigen Sie transformierte Daten an und bereinigen Sie Ressourcen.

Weitere Informationen finden Sie unter Tutorial: Erste Schritte mit AWS Glue Studio.

SDK für C++

> **Note**
>
> Weitere Informationen finden Sie unter GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - CreateCrawler
  - CreateJob
  - DeleteCrawler
  - DeleteDatabase
  - DeleteJob
  - DeleteTable
  - GetCrawler
  - GetDatabase
  - GetDatabases
  - GetJob
  - GetJobRun
  - GetJobRuns
  - GetTables
  - ListJobs
  - StartCrawler
  - StartJobRun

Aktionen

`CreateCrawler`

Das folgende Codebeispiel zeigt, wie man es benutzt`CreateCrawler`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
        }
```

- Einzelheiten zur API finden Sie CreateCrawlerin der AWS SDK für C++ API-Referenz.

`CreateJob`

Das folgende Codebeispiel zeigt die Verwendung`CreateJob`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
        }
```

- Einzelheiten zur API finden Sie CreateJobin der AWS SDK für C++ API-Referenz.

`DeleteCrawler`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCrawler`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
        }
```

- Einzelheiten zur API finden Sie DeleteCrawlerin der AWS SDK für C++ API-Referenz.

`DeleteDatabase`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDatabase`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
        }
```

- Einzelheiten zur API finden Sie DeleteDatabasein der AWS SDK für C++ API-Referenz.

`DeleteJob`

Das folgende Codebeispiel zeigt die Verwendung`DeleteJob`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
        }
```

- Einzelheiten zur API finden Sie DeleteJobin der AWS SDK für C++ API-Referenz.

`GetCrawler`

Das folgende Codebeispiel zeigt die Verwendung`GetCrawler`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
       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;
        }
```

- Einzelheiten zur API finden Sie GetCrawlerin der AWS SDK für C++ API-Referenz.

`GetDatabase`

Das folgende Codebeispiel zeigt die Verwendung`GetDatabase`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
        }
```

- Einzelheiten zur API finden Sie GetDatabasein der AWS SDK für C++ API-Referenz.

`GetJobRun`

Das folgende Codebeispiel zeigt die Verwendung`GetJobRun`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
        }
```

- Einzelheiten zur API finden Sie GetJobRunin der AWS SDK für C++ API-Referenz.

`GetJobRuns`

Das folgende Codebeispiel zeigt die Verwendung`GetJobRuns`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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());
```

- Einzelheiten zur API finden Sie GetJobRunsin der AWS SDK für C++ API-Referenz.

`GetTables`

Das folgende Codebeispiel zeigt die Verwendung`GetTables`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
       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();
        }
```

- Einzelheiten zur API finden Sie GetTablesin der AWS SDK für C++ API-Referenz.

`ListJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListJobs`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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());
```

- Einzelheiten zur API finden Sie ListJobsin der AWS SDK für C++ API-Referenz.

`StartCrawler`

Das folgende Codebeispiel zeigt die Verwendung`StartCrawler`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
       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;
        }
```

- Einzelheiten zur API finden Sie StartCrawlerin der AWS SDK für C++ API-Referenz.

`StartJobRun`

Das folgende Codebeispiel zeigt die VerwendungStartJobRun.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
    }
```

- Einzelheiten zur API finden Sie StartJobRunin der AWS SDK für C++ API-Referenz.

HealthImaging Beispiele mit SDK for C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für C++ with Aktionen ausführen und allgemeine Szenarien implementieren HealthImaging.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hallo HealthImaging

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von HealthImaging beginnen.

SDK für C++

Code für die CMake Datei CMake Lists.txt.

```
# 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})
```

Code für die Quelldatei „hello_health_imaging.cpp“.

```
#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;
}
```

- Einzelheiten zur API finden Sie ListDatastoresunter AWS SDK für C++ API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Aktionen

`DeleteImageSet`

Das folgende Codebeispiel zeigt, wie man es benutzt`DeleteImageSet`.

SDK für 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();
}
```

- Einzelheiten zur API finden Sie DeleteImageSetunter AWS SDK für C++ API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`GetDICOMImportJob`

Das folgende Codebeispiel zeigt, wie man es benutzt`GetDICOMImportJob`.

SDK für 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;
}
```

- Einzelheiten zur API finden Sie unter Get DICOMImport Job in der AWS SDK für C++ API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`GetImageFrame`

Das folgende Codebeispiel zeigt, wie man es benutzt`GetImageFrame`.

SDK für 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();
}
```

- Einzelheiten zur API finden Sie GetImageFrameunter AWS SDK für C++ API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`GetImageSetMetadata`

Das folgende Codebeispiel zeigt, wie man es benutzt`GetImageSetMetadata`.

SDK für C++

Hilfsfunktion zum Abrufen von Bildsatz-Metadaten.

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

```
}
```

Rufen Sie Bildsatz-Metadaten ohne Version ab.

```
        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;
        }
```

Rufen Sie Bildsatz-Metadaten mit Version ab.

```
        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;
        }
```

- Einzelheiten zur API finden Sie GetImageSetMetadataunter AWS SDK für C++ API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`SearchImageSets`

Das folgende Codebeispiel zeigt, wie man es benutzt`SearchImageSets`.

SDK für C++

Die Hilfsfunktion für die Suche nach Bildsätzen.

```
//! 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;
}
```

Anwendungsfall 1: EQUAL-Operator.

```
       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;
           }
       }
```

Anwendungsfall #2: BETWEEN-Operator mit DICOMStudy Datum und DICOMStudy Uhrzeit.

```
        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;
            }
        }
```

Anwendungsfall 3: BETWEEN-Operator mit createdAt. Zeitstudien wurden bisher fortgeführt.

```
        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;
            }
        }
```

Anwendungsfall #4: EQUAL-Operator für DICOMSeries instanceUID und BETWEEN für updatedAt und sortiere die Antwort in ASC-Reihenfolge für das Feld updatedAt.

```
        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;
            }
        }
```

- Einzelheiten zur API finden Sie in der API-Referenz. SearchImageSetsAWS SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

`StartDICOMImportJob`

Das folgende Codebeispiel zeigt, wie man es benutzt`StartDICOMImportJob`.

SDK für 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();
}
```

- Einzelheiten zur API finden Sie unter DICOMImportJob starten in der AWS SDK für C++ API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Szenarien

Erste Schritte mit Bildsätzen und Bild-Frames

Das folgende Codebeispiel zeigt, wie DICOM-Dateien importiert und Bildrahmen heruntergeladen werden. HealthImaging

Die Implementierung ist als Befehlszeilenanwendung strukturiert.

- Richten Sie Ressourcen für einen DICOM-Import ein.
- Importieren Sie DICOM-Dateien in einen Datenspeicher.
- Rufen Sie den Bildsatz IDs für den Importauftrag ab.
- Rufen Sie den Bildrahmen IDs für die Bildsätze ab.
- Laden Sie die Bild-Frames herunter, dekodieren Sie sie und überprüfen Sie sie.
- Bereinigen Sie die Ressourcen.

SDK für C++

Erstellen Sie einen CloudFormation Stapel mit den erforderlichen Ressourcen.

```
    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: ");
    }
```

Kopieren Sie DICOM-Dateien in den Amazon-S3-Import-Bucket.

```
   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;
    }
```

Importieren Sie die DICOM-Dateien in den Amazon-S3-Datenspeicher.

```
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;
}
```

Rufen Sie Bildsätze ab, die durch den DICOM-Importauftrag erstellt wurden.

```
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;
}
```

Rufen Sie Bild-Frame-Informationen für Bildsätze ab.

```
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();
}
```

Laden Sie Bild-Frames herunter, dekodieren Sie sie und überprüfen Sie sie.

```
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;
}
```

Bereinigen von Ressourcen.

```
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;
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - DeleteImageSet
  - DICOMImportJob bekommen
  - GetImageFrame
  - GetImageSetMetadata
  - SearchImageSets
  - DICOMImportJob starten

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

IAM-Beispiele unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für C++ mit IAM Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hello IAM

Das folgende Codebeispiel zeigt, wie Sie mit IAM beginnen.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMake Datei CMake Lists.txt.

```
# 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})
```

Code für die iam.cpp-Quellendatei.

```
#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;
}
```

- Einzelheiten zur API finden Sie ListPoliciesunter AWS SDK für C++ API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Das folgende Codebeispiel veranschaulicht, wie Sie einen Benutzer erstellen und eine Rolle annehmen lassen.

> **Warning**
>
> Um Sicherheitsrisiken zu vermeiden, sollten Sie IAM-Benutzer nicht zur Authentifizierung verwenden, wenn Sie speziell entwickelte Software entwickeln oder mit echten Daten arbeiten. Verwenden Sie stattdessen den Verbund mit einem Identitätsanbieter wie AWS IAM Identity Center.

- Erstellen Sie einen Benutzer ohne Berechtigungen.
- Erstellen einer Rolle, die die Berechtigung zum Auflisten von Amazon-S3-Buckets für das Konto erteilt.
- Hinzufügen einer Richtlinie, damit der Benutzer die Rolle übernehmen kann.
- Übernehmen Sie die Rolle und listen Sie S3-Buckets mit temporären Anmeldeinformationen auf, und bereinigen Sie dann die Ressourcen.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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;
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser

- DeleteAccessKey
- DeletePolicy
- DeleteRole
- DeleteUser
- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

Aktionen

`AttachRolePolicy`

Das folgende Codebeispiel zeigt, wie man es benutzt`AttachRolePolicy`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie AttachRolePolicyin der AWS SDK für C++ API-Referenz.

`CreateAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`CreateAccessKey`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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;
}
```

- Einzelheiten zur API finden Sie CreateAccessKeyin der AWS SDK für C++ API-Referenz.

`CreateAccountAlias`

Das folgende Codebeispiel zeigt die Verwendung`CreateAccountAlias`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie CreateAccountAliasin der AWS SDK für C++ API-Referenz.

`CreatePolicy`

Das folgende Codebeispiel zeigt die Verwendung`CreatePolicy`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie CreatePolicyin der AWS SDK für C++ API-Referenz.

`CreateRole`

Das folgende Codebeispiel zeigt die Verwendung`CreateRole`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie CreateRolein der AWS SDK für C++ API-Referenz.

`CreateUser`

Das folgende Codebeispiel zeigt die Verwendung`CreateUser`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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();
```

- Einzelheiten zur API finden Sie CreateUserin der AWS SDK für C++ API-Referenz.

`DeleteAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAccessKey`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie DeleteAccessKeyin der AWS SDK für C++ API-Referenz.

`DeleteAccountAlias`

Das folgende Codebeispiel zeigt die Verwendung`DeleteAccountAlias`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie DeleteAccountAliasin der AWS SDK für C++ API-Referenz.

`DeletePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeletePolicy`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie DeletePolicyin der AWS SDK für C++ API-Referenz.

`DeleteServerCertificate`

Das folgende Codebeispiel zeigt die Verwendung`DeleteServerCertificate`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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;
}
```

- Einzelheiten zur API finden Sie DeleteServerCertificatein der AWS SDK für C++ API-Referenz.

`DeleteUser`

Das folgende Codebeispiel zeigt die Verwendung`DeleteUser`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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();
```

- Einzelheiten zur API finden Sie DeleteUserin der AWS SDK für C++ API-Referenz.

`DetachRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`DetachRolePolicy`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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();
```

- Einzelheiten zur API finden Sie DetachRolePolicyin der AWS SDK für C++ API-Referenz.

`GetAccessKeyLastUsed`

Das folgende Codebeispiel zeigt die Verwendung`GetAccessKeyLastUsed`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie GetAccessKeyLastUsedin der AWS SDK für C++ API-Referenz.

`GetPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetPolicy`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie GetPolicyin der AWS SDK für C++ API-Referenz.

`GetServerCertificate`

Das folgende Codebeispiel zeigt die Verwendung`GetServerCertificate`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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;
}
```

- Einzelheiten zur API finden Sie GetServerCertificatein der AWS SDK für C++ API-Referenz.

`ListAccessKeys`

Das folgende Codebeispiel zeigt die Verwendung`ListAccessKeys`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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;
}
```

- Einzelheiten zur API finden Sie ListAccessKeysin der AWS SDK für C++ API-Referenz.

`ListAccountAliases`

Das folgende Codebeispiel zeigt die Verwendung`ListAccountAliases`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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;
}
```

- Einzelheiten zur API finden Sie ListAccountAliasesin der AWS SDK für C++ API-Referenz.

`ListPolicies`

Das folgende Codebeispiel zeigt die Verwendung`ListPolicies`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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;
}
```

- Einzelheiten zur API finden Sie ListPoliciesin der AWS SDK für C++ API-Referenz.

`ListServerCertificates`

Das folgende Codebeispiel zeigt die Verwendung`ListServerCertificates`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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;
}
```

- Einzelheiten zur API finden Sie ListServerCertificatesin der AWS SDK für C++ API-Referenz.

`ListUsers`

Das folgende Codebeispiel zeigt die Verwendung`ListUsers`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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;
}
```

- Einzelheiten zur API finden Sie ListUsersin der AWS SDK für C++ API-Referenz.

`PutRolePolicy`

Das folgende Codebeispiel zeigt die Verwendung`PutRolePolicy`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie PutRolePolicyin der AWS SDK für C++ API-Referenz.

`UpdateAccessKey`

Das folgende Codebeispiel zeigt die Verwendung`UpdateAccessKey`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie UpdateAccessKeyin der AWS SDK für C++ API-Referenz.

`UpdateServerCertificate`

Das folgende Codebeispiel zeigt die VerwendungUpdateServerCertificate.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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;
}
```

- Einzelheiten zur API finden Sie UpdateServerCertificatein der AWS SDK für C++ API-Referenz.

`UpdateUser`

Das folgende Codebeispiel zeigt die Verwendung`UpdateUser`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie UpdateUserin der AWS SDK für C++ API-Referenz.

AWS IoT Beispiele mit SDK for C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für C++ with Aktionen ausführen und allgemeine Szenarien implementieren AWS IoT.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen

Erste Schritte

Hallo AWS IoT

Das folgende Codebeispiel veranschaulicht, wie Sie mit der Verwendung von AWS IoT beginnen.

SDK für C++

Code für die CMake Datei CMake Lists.txt.

```
# 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})
```

Code für die Quelldatei „hello_iot.cpp“.

```
#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;
}
```

- Weitere API-Informationen finden Sie unter listThings in der AWS SDK für C++ -API-Referenz.

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erschaffe ein AWS IoT Ding.
- Generieren Sie ein Gerätezertifikat.
- Aktualisiere ein AWS IoT Ding mit Attributen.
- Geben Sie einen eindeutigen Endpunkt zurück.
- Listen Sie Ihre AWS IoT Zertifikate auf.
- Aktualisiere einen AWS IoT Schatten.
- Schreiben Sie Statusinformationen aus.
- Erstellt eine Regel.
- Listen Sie Ihre Regeln auf.
- Suchen Sie Objekte anhand des Objektnamens.
- Lösche ein AWS IoT Ding.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Erschaffe AWS IoT etwas.

```
    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();
}
```

Generieren Sie ein Gerätezertifikat und fügen Sie es an.

```
    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();
}
```

Führe verschiedene Operationen an dem AWS IoT Ding durch.

```
   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;
}
```

Bereinigen Sie die Ressourcen.

```
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();
}
```

Aktionen

`AttachThingPrincipal`

Das folgende Codebeispiel zeigt die Verwendung`AttachThingPrincipal`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie AttachThingPrincipalin der AWS SDK für C++ API-Referenz.

`CreateKeysAndCertificate`

Das folgende Codebeispiel zeigt die Verwendung`CreateKeysAndCertificate`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie CreateKeysAndCertificatein der AWS SDK für C++ API-Referenz.

`CreateThing`

Das folgende Codebeispiel zeigt die Verwendung`CreateThing`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie CreateThingin der AWS SDK für C++ API-Referenz.

`CreateTopicRule`

Das folgende Codebeispiel zeigt die Verwendung`CreateTopicRule`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie CreateTopicRulein der AWS SDK für C++ API-Referenz.

`DeleteCertificate`

Das folgende Codebeispiel zeigt die Verwendung`DeleteCertificate`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie DeleteCertificatein der AWS SDK für C++ API-Referenz.

`DeleteThing`

Das folgende Codebeispiel zeigt die Verwendung`DeleteThing`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie DeleteThingin der AWS SDK für C++ API-Referenz.

`DeleteTopicRule`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTopicRule`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie DeleteTopicRulein der AWS SDK für C++ API-Referenz.

`DescribeEndpoint`

Das folgende Codebeispiel zeigt die Verwendung`DescribeEndpoint`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie DescribeEndpointin der AWS SDK für C++ API-Referenz.

`DescribeThing`

Das folgende Codebeispiel zeigt die Verwendung`DescribeThing`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie DescribeThingin der AWS SDK für C++ API-Referenz.

`DetachThingPrincipal`

Das folgende Codebeispiel zeigt die Verwendung`DetachThingPrincipal`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie DetachThingPrincipalin der AWS SDK für C++ API-Referenz.

`ListCertificates`

Das folgende Codebeispiel zeigt die Verwendung`ListCertificates`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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;
}
```

- Einzelheiten zur API finden Sie ListCertificatesin der AWS SDK für C++ API-Referenz.

`SearchIndex`

Das folgende Codebeispiel zeigt die Verwendung`SearchIndex`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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;
}
```

- Einzelheiten zur API finden Sie SearchIndexin der AWS SDK für C++ API-Referenz.

`UpdateIndexingConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`UpdateIndexingConfiguration`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie UpdateIndexingConfigurationin der AWS SDK für C++ API-Referenz.

`UpdateThing`

Das folgende Codebeispiel zeigt die Verwendung`UpdateThing`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie UpdateThingin der AWS SDK für C++ API-Referenz.

AWS IoT data Beispiele mit SDK for C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für C++ with Aktionen ausführen und allgemeine Szenarien implementieren AWS IoT data.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`GetThingShadow`

Das folgende Codebeispiel zeigt, wie Sie`GetThingShadow`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie GetThingShadowin der AWS SDK für C++ API-Referenz.

`UpdateThingShadow`

Das folgende Codebeispiel zeigt die Verwendung`UpdateThingShadow`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie UpdateThingShadowin der AWS SDK für C++ API-Referenz.

Lambda-Beispiele unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für C++ mit Lambda Aktionen ausführen und allgemeine Szenarien implementieren.

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien

Erste Schritte

Hallo Lambda

Das folgende Codebeispiel zeigt, wie Sie mit Lambda beginnen können.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMake Datei CMake Lists.txt.

```
# 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})
```

Code für die Quelldatei „hello_lambda.cpp“.

```
#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;
}
```

- Einzelheiten zur API finden Sie ListFunctionsunter AWS SDK für C++ API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine IAM-Rolle und eine Lambda-Funktion und laden Sie den Handlercode hoch.
- Rufen Sie die Funktion mit einem einzigen Parameter auf und erhalten Sie Ergebnisse.
- Aktualisieren Sie den Funktionscode und konfigurieren Sie mit einer Umgebungsvariablen.
- Rufen Sie die Funktion mit neuen Parametern auf und erhalten Sie Ergebnisse. Zeigt das zurückgegebene Ausführungsprotokoll an.
- Listen Sie die Funktionen für Ihr Konto auf und bereinigen Sie dann die Ressourcen.

Weitere Informationen zur Verwendung von Lambda finden Sie unter Erstellen einer Lambda-Funktion mit der Konsole.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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;
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Invoke
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Aktionen

`CreateFunction`

Das folgende Codebeispiel zeigt, wie man es benutzt`CreateFunction`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
        }
```

- Einzelheiten zur API finden Sie CreateFunctionin der AWS SDK für C++ API-Referenz.

`DeleteFunction`

Das folgende Codebeispiel zeigt die Verwendung`DeleteFunction`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
    }
```

- Einzelheiten zur API finden Sie DeleteFunctionin der AWS SDK für C++ API-Referenz.

`GetFunction`

Das folgende Codebeispiel zeigt die Verwendung`GetFunction`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
       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;
        }
```

- Einzelheiten zur API finden Sie GetFunctionin der AWS SDK für C++ API-Referenz.

`Invoke`

Das folgende Codebeispiel zeigt die VerwendungInvoke.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
        }
```

- Weitere API-Informationen finden Sie unter Invoke in der AWS SDK für C++ -API-Referenz.

`ListFunctions`

Das folgende Codebeispiel zeigt, wie man es benutzt`ListFunctions`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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());
```

- Einzelheiten zur API finden Sie ListFunctionsin der AWS SDK für C++ API-Referenz.

`UpdateFunctionCode`

Das folgende Codebeispiel zeigt die Verwendung`UpdateFunctionCode`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
        }
```

- Einzelheiten zur API finden Sie UpdateFunctionCodein der AWS SDK für C++ API-Referenz.

`UpdateFunctionConfiguration`

Das folgende Codebeispiel zeigt die Verwendung`UpdateFunctionConfiguration`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
        }
```

- Einzelheiten zur API finden Sie UpdateFunctionConfigurationin der AWS SDK für C++ API-Referenz.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für C++

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

MediaConvert Beispiele mit SDK for C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für C++ with Aktionen ausführen und allgemeine Szenarien implementieren MediaConvert.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`CreateJob`

Das folgende Codebeispiel zeigt, wie Sie`CreateJob`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie CreateJobin der AWS SDK für C++ API-Referenz.

`GetJob`

Das folgende Codebeispiel zeigt die Verwendung`GetJob`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie GetJobin der AWS SDK für C++ API-Referenz.

`ListJobs`

Das folgende Codebeispiel zeigt die Verwendung`ListJobs`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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;
}
```

- Einzelheiten zur API finden Sie ListJobsin der AWS SDK für C++ API-Referenz.

Amazon-RDS-Beispiele unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie Amazon RDS verwenden. AWS SDK für C++

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Grundlagen
- Aktionen
- Szenarien

Erste Schritte

Hello Amazon RDS

Das folgende Codebeispiel zeigt die ersten Schritte mit Amazon RDS.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMake Datei CMake Lists.txt.

```
# 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})
```

Code für die Quelldatei „hello_rds.cpp“.

```
#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;
}
```

- Einzelheiten zur API finden Sie unter Describe DBInstances in der AWS SDK für C++ API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie eine benutzerdefinierte DB-Parametergruppe und legen Sie Parameterwerte fest.
- Erstellen Sie eine DB-Instance, die zur Verwendung der Parametergruppe konfiguriert ist. Die DB-Instance enthält auch eine Datenbank.
- Erstellen Sie einen Snapshot der Instance.
- Löschen Sie die Instance und die Parametergruppe.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - CreateDBInstance
  - DBParameterGruppe erstellen
  - CreateDBSnapshot
  - LöschenDBInstance
  - DBParameterGruppe löschen
  - DBEngineVersionen beschreiben
  - Beschreiben DBInstances
  - Beschreiben Sie DBParameter Gruppen
  - Beschreiben DBParameters
  - Beschreiben DBSnapshots
  - DescribeOrderableDBInstanceOptionen
  - DBParameterGruppe ändern

Aktionen

`CreateDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBInstance`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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;
    }
```

- Einzelheiten zur API finden Sie unter DBInstance In der AWS SDK für C++ API-Referenz erstellen.

`CreateDBParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBParameterGroup`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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;
    }
```

- Einzelheiten zur API finden Sie unter DBParameterGruppe erstellen in der AWS SDK für C++ API-Referenz.

`CreateDBSnapshot`

Das folgende Codebeispiel zeigt die Verwendung`CreateDBSnapshot`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
            }
```

- Einzelheiten zur API finden Sie unter DBSnapshot In der AWS SDK für C++ API-Referenz erstellen.

`DeleteDBInstance`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBInstance`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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;
        }
```

- Einzelheiten zur API finden Sie unter Löschen DBInstance in der AWS SDK für C++ API-Referenz.

`DeleteDBParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`DeleteDBParameterGroup`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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;
    }
```

- Einzelheiten zur API finden Sie unter DBParameterGruppe löschen in der AWS SDK für C++ API-Referenz.

`DescribeDBEngineVersions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBEngineVersions`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
}
```

- Einzelheiten zur API finden Sie unter DBEngineVersionen beschreiben in der AWS SDK für C++ API-Referenz.

`DescribeDBInstances`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBInstances`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
}
```

- Einzelheiten zur API finden Sie unter Describe DBInstances in der AWS SDK für C++ API-Referenz.

`DescribeDBParameterGroups`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBParameterGroups`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
        }
```

- Einzelheiten zur API finden Sie unter DBParameterGruppen beschreiben in der AWS SDK für C ++ API-Referenz.

`DescribeDBParameters`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBParameters`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
}
```

- Einzelheiten zur API finden Sie unter Describe DBParameters in der AWS SDK für C++ API-Referenz.

`DescribeDBSnapshots`

Das folgende Codebeispiel zeigt die Verwendung`DescribeDBSnapshots`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
            }
```

- Einzelheiten zur API finden Sie unter Describe DBSnapshots in der AWS SDK für C++ API-Referenz.

`DescribeOrderableDBInstanceOptions`

Das folgende Codebeispiel zeigt die Verwendung`DescribeOrderableDBInstanceOptions`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
}
```

- Einzelheiten zur API finden Sie unter DescribeOrderableDBInstanceOptionen in der AWS SDK für C++ API-Referenz.

`ModifyDBParameterGroup`

Das folgende Codebeispiel zeigt die Verwendung`ModifyDBParameterGroup`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
        }
```

- Einzelheiten zur API finden Sie unter DBParameterGruppe ändern in der AWS SDK für C++ API-Referenz.

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für C++

Zeigt, wie eine Webanwendung erstellt wird, die in einer Datenbank von Amazon Aurora Serverless gespeicherte Arbeitselemente verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer C++-REST-API, die Amazon Aurora Aurora-Serverless-Daten abfragt und von einer React-Anwendung verwendet werden kann, finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Beispiele für Amazon RDS Data Service unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon RDS Data Service Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für C++

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Szenarien

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für C++

Zeigt, wie eine Webanwendung erstellt wird, die in einer Datenbank von Amazon Aurora Serverless gespeicherte Arbeitselemente verfolgt und darüber berichtet.

Den vollständigen Quellcode und Anweisungen zur Einrichtung einer C++-REST-API, die Amazon Aurora Aurora-Serverless-Daten abfragt und von einer React-Anwendung verwendet werden kann, finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Amazon-Rekognition-Beispiele unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Rekognition Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für C++

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hello Amazon Rekognition

Die folgenden Codebeispiele zeigen, wie Sie mit der Verwendung von Amazon Rekognition beginnen.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMake Datei CMake Lists.txt.

```
# 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})
```

Code für die Quelldatei „hello_rekognition.cpp“.

```
#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;
}
```

- Einzelheiten zur API finden Sie ListCollectionsunter AWS SDK für C++ API-Referenz.

Aktionen

`DetectLabels`

Das folgende Codebeispiel zeigt die Verwendung`DetectLabels`.

Weitere Informationen finden Sie unter Erkennen von Labels in einem Bild.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie DetectLabelsin der AWS SDK für C++ API-Referenz.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für C++

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Amazon-S3-Beispiele unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie Aktionen ausführen und allgemeine Szenarien implementieren, indem Sie Amazon S3 verwenden. AWS SDK für C++

Bei Grundlagen handelt es sich um Codebeispiele, die Ihnen zeigen, wie Sie die wesentlichen Vorgänge innerhalb eines Services ausführen.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte

- Grundlagen
- Aktionen
- Szenarien

Erste Schritte

Hello Amazon S3

Das folgende Codebeispiel zeigt die ersten Schritte mit Amazon S3.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMake Datei CMake Lists.txt.

```
# 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})
```

Code für die Quelldatei „hello_s3.cpp“.

```
#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;
}
```

- Einzelheiten zur API finden Sie ListBucketsunter AWS SDK für C++ API-Referenz.

Grundlagen

Kennenlernen der Grundlagen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie einen Bucket und laden Sie eine Datei in ihn hoch.
- Laden Sie ein Objekt aus einem Bucket herunter.
- Kopieren Sie ein Objekt in einen Unterordner eines Buckets.
- Listen Sie die Objekte in einem Bucket auf.
- Löschen Sie die Bucket-Objekte und den Bucket.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
#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();
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2

- PutObject

Aktionen

`AbortMultipartUpload`

Das folgende Codebeispiel zeigt, wie man es benutzt`AbortMultipartUpload`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie AbortMultipartUploadin der AWS SDK für C++ API-Referenz.

`CompleteMultipartUpload`

Das folgende Codebeispiel zeigt die Verwendung`CompleteMultipartUpload`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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;
}
```

- Einzelheiten zur API finden Sie CompleteMultipartUploadin der AWS SDK für C++ API-Referenz.

`CopyObject`

Das folgende Codebeispiel zeigt die Verwendung`CopyObject`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie CopyObjectin der AWS SDK für C++ API-Referenz.

`CreateBucket`

Das folgende Codebeispiel zeigt die Verwendung`CreateBucket`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie CreateBucketin der AWS SDK für C++ API-Referenz.

`CreateMultipartUpload`

Das folgende Codebeispiel zeigt die Verwendung`CreateMultipartUpload`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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;
}
```

- Einzelheiten zur API finden Sie CreateMultipartUploadin der AWS SDK für C++ API-Referenz.

`DeleteBucket`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucket`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie DeleteBucketin der AWS SDK für C++ API-Referenz.

`DeleteBucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucketPolicy`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie DeleteBucketPolicyin der AWS SDK für C++ API-Referenz.

`DeleteBucketWebsite`

Das folgende Codebeispiel zeigt die Verwendung`DeleteBucketWebsite`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie DeleteBucketWebsitein der AWS SDK für C++ API-Referenz.

`DeleteObject`

Das folgende Codebeispiel zeigt die Verwendung`DeleteObject`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie DeleteObjectin der AWS SDK für C++ API-Referenz.

`DeleteObjects`

Das folgende Codebeispiel zeigt die Verwendung`DeleteObjects`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie DeleteObjectsin der AWS SDK für C++ API-Referenz.

`GetBucketAcl`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketAcl`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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";
}
```

- Einzelheiten zur API finden Sie GetBucketAclin der AWS SDK für C++ API-Referenz.

`GetBucketPolicy`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketPolicy`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie GetBucketPolicyin der AWS SDK für C++ API-Referenz.

`GetBucketWebsite`

Das folgende Codebeispiel zeigt die Verwendung`GetBucketWebsite`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie GetBucketWebsitein der AWS SDK für C++ API-Referenz.

`GetObject`

Das folgende Codebeispiel zeigt die Verwendung`GetObject`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie GetObjectin der AWS SDK für C++ API-Referenz.

`GetObjectAcl`

Das folgende Codebeispiel zeigt die Verwendung`GetObjectAcl`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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";
    }
}
```

- Einzelheiten zur API finden Sie GetObjectAclin der AWS SDK für C++ API-Referenz.

`GetObjectAttributes`

Das folgende Codebeispiel zeigt die Verwendung`GetObjectAttributes`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
// ! 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;
}
```

- Einzelheiten zur API finden Sie GetObjectAttributesin der AWS SDK für C++ API-Referenz.

`ListBuckets`

Das folgende Codebeispiel zeigt die Verwendung`ListBuckets`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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;
}
```

- Einzelheiten zur API finden Sie ListBucketsin der AWS SDK für C++ API-Referenz.

`ListObjectsV2`

Das folgende Codebeispiel zeigt die Verwendung`ListObjectsV2`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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;
}
```

- Einzelheiten zur API finden Sie unter ListObjectsV2 in der AWS SDK für C++ API-Referenz.

`PutBucketAcl`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketAcl`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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;
}
```

- Einzelheiten zur API finden Sie PutBucketAclin der AWS SDK für C++ API-Referenz.

`PutBucketPolicy`

Das folgende Codebeispiel zeigt die VerwendungPutBucketPolicy.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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"
            "}";
}
```

- Einzelheiten zur API finden Sie PutBucketPolicyin der AWS SDK für C++ API-Referenz.

`PutBucketWebsite`

Das folgende Codebeispiel zeigt die Verwendung`PutBucketWebsite`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie PutBucketWebsitein der AWS SDK für C++ API-Referenz.

`PutObject`

Das folgende Codebeispiel zeigt die Verwendung`PutObject`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie PutObjectin der AWS SDK für C++ API-Referenz.

`PutObjectAcl`

Das folgende Codebeispiel zeigt die Verwendung`PutObjectAcl`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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;
}
```

- Einzelheiten zur API finden Sie PutObjectAclin der AWS SDK für C++ API-Referenz.

`UploadPart`

Das folgende Codebeispiel zeigt die Verwendung`UploadPart`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie UploadPartin der AWS SDK für C++ API-Referenz.

Szenarien

Eine vorsignierte URL erstellen

Das folgende Codebeispiel veranschaulicht, wie Sie eine vorsignierte URL für Amazon S3 erstellen und ein Objekt hochladen.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Generieren Sie eine vorsignierte URL, um ein Objekt herunterzuladen.

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

Führen Sie einen Download unter Verwendung von libcurl durch.

```
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;
}
```

Generieren Sie eine vorsignierte URL, um ein Objekt hochzuladen.

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

Führen Sie einen Upload unter Verwendung von libcurl durch.

```
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;
    }
}
```

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für C++

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Arbeiten mit Amazon-S3-Objektintegrität

Das folgende Codebeispiel zeigt, wie Sie mit Features der S3-Objektintegrität arbeiten.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Führen Sie ein interaktives Szenario aus, das die Features der Amazon-S3-Objektintegrität zeigt.

```
//! 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;
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - AbortMultipartUpload
  - CompleteMultipartUpload
  - CreateMultipartUpload
  - DeleteObject
  - GetObjectAttributes
  - PutObject
  - UploadPart

Secrets-Manager-Beispiele unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für C++ mit Secrets Manager Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`GetSecretValue`

Das folgende Codebeispiel zeigt, wie Sie`GetSecretValue`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

```
}
```

- Einzelheiten zur API finden Sie GetSecretValuein der AWS SDK für C++ API-Referenz.

Amazon-SES-Beispiele unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon SES Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für C++

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`CreateReceiptFilter`

Das folgende Codebeispiel zeigt, wie Sie`CreateReceiptFilter`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie CreateReceiptFilterin der AWS SDK für C++ API-Referenz.

`CreateReceiptRule`

Das folgende Codebeispiel zeigt die Verwendung`CreateReceiptRule`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie CreateReceiptRulein der AWS SDK für C++ API-Referenz.

`CreateReceiptRuleSet`

Das folgende Codebeispiel zeigt die Verwendung`CreateReceiptRuleSet`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie CreateReceiptRuleSetin der AWS SDK für C++ API-Referenz.

`CreateTemplate`

Das folgende Codebeispiel zeigt die Verwendung`CreateTemplate`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie CreateTemplatein der AWS SDK für C++ API-Referenz.

`DeleteIdentity`

Das folgende Codebeispiel zeigt die Verwendung`DeleteIdentity`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie DeleteIdentityin der AWS SDK für C++ API-Referenz.

`DeleteReceiptFilter`

Das folgende Codebeispiel zeigt die Verwendung`DeleteReceiptFilter`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie DeleteReceiptFilterin der AWS SDK für C++ API-Referenz.

`DeleteReceiptRule`

Das folgende Codebeispiel zeigt die Verwendung`DeleteReceiptRule`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie DeleteReceiptRulein der AWS SDK für C++ API-Referenz.

`DeleteReceiptRuleSet`

Das folgende Codebeispiel zeigt die Verwendung`DeleteReceiptRuleSet`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie DeleteReceiptRuleSetin der AWS SDK für C++ API-Referenz.

`DeleteTemplate`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTemplate`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie DeleteTemplatein der AWS SDK für C++ API-Referenz.

`GetTemplate`

Das folgende Codebeispiel zeigt die Verwendung`GetTemplate`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie GetTemplatein der AWS SDK für C++ API-Referenz.

`ListIdentities`

Das folgende Codebeispiel zeigt die Verwendung`ListIdentities`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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;
}
```

- Einzelheiten zur API finden Sie ListIdentitiesin der AWS SDK für C++ API-Referenz.

`ListReceiptFilters`

Das folgende Codebeispiel zeigt die Verwendung`ListReceiptFilters`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie ListReceiptFiltersin der AWS SDK für C++ API-Referenz.

`SendEmail`

Das folgende Codebeispiel zeigt die Verwendung`SendEmail`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

```
}
```

- Einzelheiten zur API finden Sie SendEmailin der AWS SDK für C++ API-Referenz.

`SendTemplatedEmail`

Das folgende Codebeispiel zeigt die Verwendung`SendTemplatedEmail`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

```
}
```

- Einzelheiten zur API finden Sie SendTemplatedEmailin der AWS SDK für C++ API-Referenz.

`UpdateTemplate`

Das folgende Codebeispiel zeigt die Verwendung`UpdateTemplate`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie UpdateTemplatein der AWS SDK für C++ API-Referenz.

`VerifyEmailIdentity`

Das folgende Codebeispiel zeigt die Verwendung`VerifyEmailIdentity`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie VerifyEmailIdentityin der AWS SDK für C++ API-Referenz.

Szenarien

Erstellen eines Trackers für Aurora-Serverless-Arbeitsaufgaben

Das folgende Codebeispiel zeigt, wie eine Webanwendung erstellt wird, die Arbeitselemente in einer Datenbank von Amazon Aurora Serverless verfolgt und mithilfe von Amazon Simple Email Service (Amazon SES) Berichte sendet.

SDK für C++

> Zeigt, wie eine Webanwendung erstellt wird, die in einer Datenbank von Amazon Aurora Serverless gespeicherte Arbeitselemente verfolgt und darüber berichtet.
>
> Den vollständigen Quellcode und Anweisungen zur Einrichtung einer C++-REST-API, die Amazon Aurora Aurora-Serverless-Daten abfragt und von einer React-Anwendung verwendet werden kann, finden Sie im vollständigen Beispiel unter GitHub.

In diesem Beispiel verwendete Dienste

- Aurora
- Amazon RDS
- Amazon RDS Data Service
- Amazon SES

Amazon-SNS-Beispiele unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon SNS Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für C++

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hello Amazon SNS

Das folgende Codebeispiel zeigt die ersten Schritte mit Amazon SNS.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMake Datei CMake Lists.txt.

```
# 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})
```

Code für die Quelldatei „hello_sns.cpp“.

```
#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;
```

```
}
```

- Einzelheiten zur API finden Sie ListTopicsunter AWS SDK für C++ API-Referenz.

Aktionen

`CreateTopic`

Das folgende Codebeispiel zeigt die Verwendung`CreateTopic`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie CreateTopicin der AWS SDK für C++ API-Referenz.

`DeleteTopic`

Das folgende Codebeispiel zeigt die Verwendung`DeleteTopic`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie DeleteTopicin der AWS SDK für C++ API-Referenz.

`GetSMSAttributes`

Das folgende Codebeispiel zeigt die Verwendung`GetSMSAttributes`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie unter Get SMSAttributes in AWS SDK für C++ API-Referenz.

`GetTopicAttributes`

Das folgende Codebeispiel zeigt die Verwendung`GetTopicAttributes`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Einzelheiten zur API finden Sie GetTopicAttributesin der AWS SDK für C++ API-Referenz.

`ListSubscriptions`

Das folgende Codebeispiel zeigt die Verwendung`ListSubscriptions`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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;
}
```

- Einzelheiten zur API finden Sie ListSubscriptionsin der AWS SDK für C++ API-Referenz.

`ListTopics`

Das folgende Codebeispiel zeigt die Verwendung`ListTopics`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
//! 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;
}
```

- Einzelheiten zur API finden Sie ListTopicsin der AWS SDK für C++ API-Referenz.

`Publish`

Das folgende Codebeispiel zeigt die Verwendung`Publish`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

Veröffentlichen Sie eine Nachricht mit einem Attribut.

```
        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;
        }
```

- Weitere API-Informationen finden Sie unter Publish in der AWS SDK für C++ -API-Referenz.

`SetSMSAttributes`

Das folgende Codebeispiel zeigt, wie man es benutzt`SetSMSAttributes`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

So verwenden Sie Amazon SNS, um das SMSType Standardattribut festzulegen.

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

- Einzelheiten zur API finden Sie unter Set SMSAttributes in der AWS SDK für C++ API-Referenz.

`Subscribe`

Das folgende Codebeispiel zeigt die Verwendung`Subscribe`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Abonnieren Sie eine E-Mail-Adresse für ein Thema.

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

Abonnieren Sie eine mobile Anwendung für ein Thema.

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

Abonnieren Sie eine Lambda-Funktion für ein Thema.

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

Abonnieren Sie eine SQS-Warteschlange für ein Thema.

```
        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;
            }
```

Abonnieren Sie ein Thema mit einem Filter.

```
       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;
}
```

- Weitere API-Informationen finden Sie unter Subscribe in der AWS SDK für C++ -API-Referenz.

`Unsubscribe`

Das folgende Codebeispiel zeigt, wie man es benutzt`Unsubscribe`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

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

- Weitere API-Informationen finden Sie unter Unsubscribe in der AWS SDK für C++ -API-Referenz.

Szenarien

Erstellen einer Serverless-Anwendung zur Verwaltung von Fotos

Das folgende Codebeispiel zeigt, wie eine Serverless-Anwendung erstellt wird, mit der Benutzer Fotos mithilfe von Labels erstellen können.

SDK für C++

Zeigt, wie eine Anwendung zur Verwaltung von Fotobeständen entwickelt wird, die mithilfe von Amazon Rekognition Labels in Bildern erkennt und sie für einen späteren Abruf speichert.

Den vollständigen Quellcode und Anweisungen zur Einrichtung und Ausführung finden Sie im vollständigen Beispiel unter GitHub.

Einen tiefen Einblick in den Ursprung dieses Beispiels finden Sie im Beitrag in der AWS - Community.

In diesem Beispiel verwendete Dienste

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Veröffentlichen einer SMS-Nachricht

Das folgende Codebeispiel zeigt, wie SMS-Nachrichten über Amazon SNS veröffentlicht werden.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
/**
```

```
 * 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();
}
```

- Weitere API-Informationen finden Sie unter Publish in der AWS SDK für C++ -API-Referenz.

Veröffentlichen Sie Nachrichten in Warteschlangen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Thema (FIFO oder Nicht-FIFO).
- Abonnieren Sie mehrere Warteschlangen für das Thema mit der Option, einen Filter anzuwenden.
- Veröffentlichen Sie eine Nachricht im Thema.
- Fragen Sie die Warteschlangen nach empfangenen Nachrichten ab.

SDK für C++

> **Note**
>
> Es gibt noch mehr GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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();
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue

- DeleteTopic
- GetQueueAttributes
- Veröffentlichen
- ReceiveMessage
- SetQueueAttributes
- Abonnieren
- Unsubscribe

Amazon-SQS-Beispiele unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie AWS SDK für C++ mit Amazon SQS Aktionen ausführen und allgemeine Szenarien implementieren.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Erste Schritte
- Aktionen
- Szenarien

Erste Schritte

Hello Amazon SQS

Das folgende Codebeispiel zeigt die ersten Schritte mit Amazon SQS.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu. GitHub Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

Code für die CMake Datei CMake Lists.txt.

```
# 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})
```

Code für die Quelldatei „hello_sqs.cpp“.

```
#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;
}
```

- Einzelheiten zur API finden Sie ListQueuesunter AWS SDK für C++ API-Referenz.

Aktionen

`ChangeMessageVisibility`

Das folgende Codebeispiel zeigt die Verwendung`ChangeMessageVisibility`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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();
}
```

- Einzelheiten zur API finden Sie ChangeMessageVisibilityin der AWS SDK für C++ API-Referenz.

`CreateQueue`

Das folgende Codebeispiel zeigt die Verwendung`CreateQueue`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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();
}
```

- Einzelheiten zur API finden Sie CreateQueuein der AWS SDK für C++ API-Referenz.

`DeleteMessage`

Das folgende Codebeispiel zeigt die Verwendung`DeleteMessage`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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();
}
```

- Einzelheiten zur API finden Sie DeleteMessagein der AWS SDK für C++ API-Referenz.

`DeleteMessageBatch`

Das folgende Codebeispiel zeigt die Verwendung`DeleteMessageBatch`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
    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;
            }
```

- Einzelheiten zur API finden Sie DeleteMessageBatchin der AWS SDK für C++ API-Referenz.

`DeleteQueue`

Das folgende Codebeispiel zeigt die Verwendung`DeleteQueue`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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();
}
```

- Einzelheiten zur API finden Sie DeleteQueuein der AWS SDK für C++ API-Referenz.

`GetQueueAttributes`

Das folgende Codebeispiel zeigt die Verwendung`GetQueueAttributes`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;


            }
```

- Einzelheiten zur API finden Sie GetQueueAttributesin der AWS SDK für C++ API-Referenz.

`GetQueueUrl`

Das folgende Codebeispiel zeigt die Verwendung`GetQueueUrl`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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();
}
```

- Einzelheiten zur API finden Sie GetQueueUrlin der AWS SDK für C++ API-Referenz.

`ListQueues`

Das folgende Codebeispiel zeigt die Verwendung`ListQueues`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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;
}
```

- Einzelheiten zur API finden Sie ListQueuesin der AWS SDK für C++ API-Referenz.

`ReceiveMessage`

Das folgende Codebeispiel zeigt die Verwendung`ReceiveMessage`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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();
}
```

- Einzelheiten zur API finden Sie ReceiveMessagein der AWS SDK für C++ API-Referenz.

`SendMessage`

Das folgende Codebeispiel zeigt die Verwendung`SendMessage`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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();
}
```

- Einzelheiten zur API finden Sie SendMessagein der AWS SDK für C++ API-Referenz.

`SetQueueAttributes`

Das folgende Codebeispiel zeigt die Verwendung`SetQueueAttributes`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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();
}
```

Konfigurieren Sie eine Warteschlange für unzustellbare Nachrichten.

```
        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();
}
```

Konfigurieren Sie eine Amazon-SQS-Warteschlange für die Verwendung einer Langabfrage.

```
        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();
}
```

- Einzelheiten zur API finden Sie SetQueueAttributesin der AWS SDK für C++ API-Referenz.

Szenarien

Veröffentlichen Sie Nachrichten in Warteschlangen

Wie das aussehen kann, sehen Sie am nachfolgenden Beispielcode:

- Erstellen Sie ein Thema (FIFO oder Nicht-FIFO).
- Abonnieren Sie mehrere Warteschlangen für das Thema mit der Option, einen Filter anzuwenden.
- Veröffentlichen Sie eine Nachricht im Thema.

- Fragen Sie die Warteschlangen nach empfangenen Nachrichten ab.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
        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();
}
```

- Weitere API-Informationen finden Sie in den folgenden Themen der AWS SDK für C++ -API-Referenz.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Veröffentlichen
  - ReceiveMessage
  - SetQueueAttributes
  - Abonnieren
  - Unsubscribe

AWS STS Beispiele mit SDK for C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von AWS SDK für C++ with Aktionen ausführen und allgemeine Szenarien implementieren AWS STS.

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen

Aktionen

`AssumeRole`

Das folgende Codebeispiel zeigt, wie Sie`AssumeRole`.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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();
}
```

- Einzelheiten zur API finden Sie AssumeRolein der AWS SDK für C++ API-Referenz.

Beispiele für Amazon Transcribe Streaming unter Verwendung von SDK für C++

Die folgenden Codebeispiele zeigen Ihnen, wie Sie mithilfe von Amazon Transcribe Streaming Aktionen ausführen und allgemeine Szenarien implementieren. AWS SDK für C++

Aktionen sind Codeauszüge aus größeren Programmen und müssen im Kontext ausgeführt werden. Während Aktionen Ihnen zeigen, wie Sie einzelne Service-Funktionen aufrufen, können Sie Aktionen im Kontext der zugehörigen Szenarien anzeigen.

Szenarien sind Codebeispiele, die Ihnen zeigen, wie Sie bestimmte Aufgaben ausführen, indem Sie mehrere Funktionen innerhalb eines Service aufrufen oder mit anderen AWS-Services kombinieren.

Jedes Beispiel enthält einen Link zum vollständigen Quellcode, wo Sie Anweisungen zum Einrichten und Ausführen des Codes im Kodex finden.

Themen

- Aktionen
- Szenarien

Aktionen

`StartStreamTranscription`

Das folgende Codebeispiel zeigt, wie Sie. `StartStreamTranscription`

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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;
}
```

- Einzelheiten zur API finden Sie StartStreamTranscriptionin der AWS SDK für C++ API-Referenz.

Szenarien

Transkribieren einer Audiodatei

Das folgende Code-Beispiel zeigt, wie mit Amazon Transcribe-Streaming eine Transkription einer Quell-Audiodatei generiert wird.

SDK für C++

> **Note**
>
> Es gibt noch mehr dazu GitHub. Hier finden Sie das vollständige Beispiel und erfahren, wie Sie das AWS -Code-Beispiel- einrichten und ausführen.

```
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;
}
```

- Einzelheiten zur API finden Sie StartStreamTranscriptionin der AWS SDK für C++ API-Referenz.

Sicherheit für AWS SDK for C++

Cloud-Sicherheit genießt bei Amazon Web Services (AWS) höchste Priorität. Als AWS -Kunde profitieren Sie von einer Rechenzentrums- und Netzwerkarchitektur, die zur Erfüllung der Anforderungen von Organisationen entwickelt wurden, für die Sicherheit eine kritische Bedeutung hat. Sicherheit ist eine gemeinsame Verantwortung von Ihnen AWS und Ihnen. Im Modell der übergreifenden Verantwortlichkeit wird Folgendes mit „Sicherheit der Cloud“ bzw. „Sicherheit in der Cloud“ umschrieben:

Sicherheit der Cloud — AWS ist verantwortlich für den Schutz der Infrastruktur, auf der alle in der AWS Cloud angebotenen Dienste ausgeführt werden, und für die Bereitstellung von Diensten, die Sie sicher nutzen können. Unsere Sicherheitsverantwortung hat bei uns höchste Priorität AWS, und die Wirksamkeit unserer Sicherheit wird im Rahmen der AWS Compliance-Programme regelmäßig von externen Prüfern getestet und verifiziert.

Sicherheit in der Cloud — Ihre Verantwortung richtet sich nach dem von Ihnen genutzten AWS Dienst und anderen Faktoren, wie der Sensibilität Ihrer Daten, den Anforderungen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften.

Dieses AWS Produkt oder dieser Service folgt dem Modell der gemeinsamen Verantwortung in Bezug auf die spezifischen Amazon Web Services (AWS) -Services, die es unterstützt. Informationen zur AWS Servicesicherheit finden Sie auf der Seite mit der Dokumentation zur AWS Servicesicherheit und den AWS Services, für die das AWS Compliance-Programm zur Einhaltung der Vorschriften zuständig ist.

Themen

- Datenschutz im AWS SDK for C++
- Identitäts- und Zugriffsverwaltung
- Konformitätsvalidierung für dieses AWS Produkt oder diese Dienstleistung
- Resilienz für dieses AWS Produkt oder diese Dienstleistung
- Infrastruktursicherheit für dieses AWS Produkt oder diese Dienstleistung
- Erzwingen einer TLS-Mindestversion in der AWS SDK für C++
- Migration des Amazon S3 S3-Verschlüsselungsclients (V1 auf V2)
- Migration des Amazon S3 S3-Verschlüsselungsclients (V2 auf V3)

Datenschutz im AWS SDK for C++

Das AWS Modell der mit gilt für den Datenschutz in AWS SDK for C++. Wie in diesem Modell beschrieben, AWS ist verantwortlich für den Schutz der globalen Infrastruktur, auf der die gesamte Infrastruktur ausgeführt wird AWS Cloud. Sie sind dafür verantwortlich, die Kontrolle über Ihre in dieser Infrastruktur gehosteten Inhalte zu behalten. Sie sind auch für die Sicherheitskonfiguration und die Verwaltungsaufgaben für die von Ihnen verwendeten AWS-Services verantwortlich. Weitere Informationen zum Datenschutz finden Sie unter Häufig gestellte Fragen zum Datenschutz. Informationen zum Datenschutz in Europa finden Sie im Blog-Beitrag AWS -Modell der geteilten Verantwortung und in der DSGVO im AWS -Sicherheitsblog.

Aus Datenschutzgründen empfehlen wir, dass Sie AWS-Konto Anmeldeinformationen schützen und einzelne Benutzer mit AWS IAM Identity Center oder AWS Identity and Access Management (IAM) einrichten. So erhält jeder Benutzer nur die Berechtigungen, die zum Durchführen seiner Aufgaben erforderlich sind. Außerdem empfehlen wir, die Daten mit folgenden Methoden schützen:

- Verwenden Sie für jedes Konto die Multi-Faktor-Authentifizierung (MFA).
- Wird verwendet SSL/TLS , um mit AWS Ressourcen zu kommunizieren. Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Richten Sie die API und die Protokollierung von Benutzeraktivitäten mit ein AWS CloudTrail. Informationen zur Verwendung von CloudTrail Pfaden zur Erfassung von AWS Aktivitäten finden Sie unter Arbeiten mit CloudTrail Pfaden im AWS CloudTrail Benutzerhandbuch.
- Verwenden Sie AWS Verschlüsselungslösungen zusammen mit allen darin enthaltenen Standardsicherheitskontrollen AWS-Services.
- Verwenden Sie erweiterte verwaltete Sicherheitsservices wie Amazon Macie, die dabei helfen, in Amazon S3 gespeicherte persönliche Daten zu erkennen und zu schützen.
- Wenn Sie für den Zugriff AWS über eine Befehlszeilenschnittstelle oder eine API FIPS 140-3-validierte kryptografische Module benötigen, verwenden Sie einen FIPS-Endpunkt. Weitere Informationen über verfügbare FIPS-Endpunkte finden Sie unter Federal Information Processing Standard (FIPS) 140-3.

Wir empfehlen dringend, in Freitextfeldern, z. B. im Feld Name, keine vertraulichen oder sensiblen Informationen wie die E-Mail-Adressen Ihrer Kunden einzugeben. Dies gilt auch, wenn Sie mit SDK for C++ oder einem anderen Programm AWS-Services über die Konsole AWS CLI, API oder arbeiten AWS SDKs. Alle Daten, die Sie in Tags oder Freitextfelder eingeben, die für Namen verwendet werden, können für Abrechnungs- oder Diagnoseprotokolle verwendet werden. Wenn Sie eine URL

für einen externen Server bereitstellen, empfehlen wir dringend, keine Anmeldeinformationen zur Validierung Ihrer Anforderung an den betreffenden Server in die URL einzuschließen.

Identitäts- und Zugriffsverwaltung

AWS Identity and Access Management (IAM) hilft einem Administrator AWS-Service , den Zugriff auf Ressourcen sicher zu AWS kontrollieren. IAM-Administratoren kontrollieren, wer authentifiziert (angemeldet) und autorisiert werden kann (über Berechtigungen verfügt), um Ressourcen zu verwenden. AWS IAM ist ein Programm AWS-Service , das Sie ohne zusätzliche Kosten nutzen können.

Themen

- Zielgruppe
- Authentifizierung mit Identitäten
- Verwalten des Zugriffs mit Richtlinien
- Wie AWS-Services arbeiten Sie mit IAM
- Problembehebung bei AWS Identität und Zugriff

Zielgruppe

Die Art und Weise, wie Sie AWS Identity and Access Management (IAM) verwenden, hängt von der Arbeit ab, in der Sie tätig sind. AWS

Dienstbenutzer — Wenn Sie dies AWS-Services für Ihre Arbeit verwenden, stellt Ihnen Ihr Administrator die erforderlichen Anmeldeinformationen und Berechtigungen zur Verfügung. Wenn Sie für Ihre Arbeit mehr AWS Funktionen verwenden, benötigen Sie möglicherweise zusätzliche Berechtigungen. Wenn Sie die Funktionsweise der Zugriffskontrolle nachvollziehen, wissen Sie bereits, welche Berechtigungen Sie von Ihrem Administrator anfordern müssen. Falls Sie auf eine Funktion in der von Ihnen verwendeten Version nicht zugreifen können AWS, finden Problembehebung bei AWS Identität und Zugriff Sie weitere Informationen in der Bedienungsanleitung des AWS-Service Geräts, das Sie verwenden.

Serviceadministrator — Wenn Sie in Ihrem Unternehmen für die AWS Ressourcen verantwortlich sind, haben Sie wahrscheinlich vollen Zugriff auf AWS. Es ist Ihre Aufgabe, zu bestimmen, auf welche AWS Funktionen und Ressourcen Ihre Servicebenutzer zugreifen sollen. Anschließend müssen Sie Anforderungen an Ihren IAM-Administrator senden, um die Berechtigungen der

Servicebenutzer zu ändern. Lesen Sie die Informationen auf dieser Seite, um die Grundkonzepte von IAM nachzuvollziehen. Weitere Informationen darüber, wie Ihr Unternehmen IAM verwenden kann AWS, finden Sie in der Benutzeranleitung des von AWS-Service Ihnen verwendeten.

IAM-Administrator: Wenn Sie als IAM-Administrator fungieren, sollten Sie Einzelheiten dazu kennen, wie Sie Richtlinien zur Verwaltung des Zugriffs auf AWS verfassen können. Beispiele für AWS identitätsbasierte Richtlinien, die Sie in IAM verwenden können, finden Sie im Benutzerhandbuch der AWS-Service von Ihnen verwendeten.

Authentifizierung mit Identitäten

Authentifizierung ist die Art und Weise, wie Sie sich AWS mit Ihren Identitätsdaten anmelden. Sie müssen sich als IAM-Benutzer authentifizieren oder eine IAM-Rolle annehmen. Root-Benutzer des AWS-Kontos

Sie können sich als föderierte Identität anmelden, indem Sie Anmeldeinformationen aus einer Identitätsquelle wie AWS IAM Identity Center (IAM Identity Center), Single Sign-On-Authentifizierung oder Anmeldeinformationen verwenden. Google/Facebook Weitere Informationen zum Anmelden finden Sie unter So melden Sie sich bei Ihrem AWS-Konto an im Benutzerhandbuch für AWS-Anmeldung .

AWS Bietet für den programmatischen Zugriff ein SDK und eine CLI zum kryptografischen Signieren von Anfragen. Weitere Informationen finden Sie unter AWS Signature Version 4 for API requests im IAM-Benutzerhandbuch.

AWS-Konto Root-Benutzer

Wenn Sie einen erstellen AWS-Konto, beginnen Sie mit einer Anmeldeidentität, dem sogenannten AWS-Konto Root-Benutzer, der vollständigen Zugriff auf alle AWS-Services Ressourcen hat. Wir raten ausdrücklich davon ab, den Root-Benutzer für Alltagsaufgaben zu verwenden. Eine Liste der Aufgaben, für die Sie sich als Root-Benutzer anmelden müssen, finden Sie unter Tasks that require root user credentials im IAM-Benutzerhandbuch.

Verbundidentität

Es hat sich bewährt, dass menschliche Benutzer für den Zugriff AWS-Services mithilfe temporärer Anmeldeinformationen einen Verbund mit einem Identitätsanbieter verwenden müssen.

Eine föderierte Identität ist ein Benutzer aus Ihrem Unternehmensverzeichnis, Ihrem Directory Service Web-Identitätsanbieter oder der AWS-Services mithilfe von Anmeldeinformationen

aus einer Identitätsquelle zugreift. Verbundene Identitäten übernehmen Rollen, die temporäre Anmeldeinformationen bereitstellen.

Für die zentrale Zugriffsverwaltung empfehlen wir AWS IAM Identity Center. Weitere Informationen finden Sie unter Was ist IAM Identity Center? im AWS IAM Identity Center -Benutzerhandbuch.

IAM-Benutzer und -Gruppen

Ein IAM-Benutzer ist eine Identität mit bestimmten Berechtigungen für eine einzelne Person oder Anwendung. Wir empfehlen die Verwendung temporärer Anmeldeinformationen anstelle von IAM-Benutzern mit langfristigen Anmeldeinformationen. Weitere Informationen finden Sie im IAM-Benutzerhandbuch unter Erfordern, dass menschliche Benutzer den Verbund mit einem Identitätsanbieter verwenden müssen, um AWS mithilfe temporärer Anmeldeinformationen darauf zugreifen zu können.

Eine IAM-Gruppe spezifiziert eine Sammlung von IAM-Benutzern und erleichtert die Verwaltung von Berechtigungen für große Gruppen von Benutzern. Weitere Informationen finden Sie unter Anwendungsfälle für IAM-Benutzer im IAM-Benutzerhandbuch.

IAM-Rollen

Eine IAM-Rolle ist eine Identität mit spezifischen Berechtigungen, die temporäre Anmeldeinformationen bereitstellt. Sie können eine Rolle übernehmen, indem Sie von einer Benutzer- zu einer IAM-Rolle (Konsole) wechseln AWS CLI oder einen AWS API-Vorgang aufrufen. Weitere Informationen finden Sie unter Methoden, um eine Rolle zu übernehmen im IAM-Benutzerhandbuch.

IAM-Rollen sind nützlich für den Verbundbenutzer-Zugriff, temporäre IAM-Benutzerberechtigungen, kontoübergreifenden Zugriff, serviceübergreifenden Zugriff und Anwendungen, die auf Amazon EC2 laufen. Weitere Informationen finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Verwalten des Zugriffs mit Richtlinien

Sie kontrollieren den Zugriff, AWS indem Sie Richtlinien erstellen und diese an AWS Identitäten oder Ressourcen anhängen. Eine Richtlinie definiert Berechtigungen, wenn sie mit einer Identität oder Ressource verknüpft sind. AWS bewertet diese Richtlinien, wenn ein Principal eine Anfrage stellt. Die meisten Richtlinien werden AWS als JSON-Dokumente gespeichert. Weitere Informationen zu JSON-Richtliniendokumenten finden Sie unter Übersicht über JSON-Richtlinien im IAM-Benutzerhandbuch.

Mit Hilfe von Richtlinien legen Administratoren fest, wer Zugriff auf was hat, indem sie definieren, welches Prinzipal welche Aktionen auf welchen Ressourcenund unter welchen Bedingungendurchführen darf.

Standardmäßig haben Benutzer, Gruppen und Rollen keine Berechtigungen. Ein IAM-Administrator erstellt IAM-Richtlinien und fügt sie zu Rollen hinzu, die die Benutzer dann übernehmen können. IAM-Richtlinien definieren Berechtigungen unabhängig von der Methode, die zur Ausführung der Operation verwendet wird.

Identitätsbasierte Richtlinien

Identitätsbasierte Richtlinien sind JSON-Berechtigungsrichtliniendokumente, die Sie einer Identität (Benutzer, Gruppe oder Rolle) anfügen können. Diese Richtlinien steuern, welche Aktionen Identitäten für welche Ressourcen und unter welchen Bedingungen ausführen können. Informationen zum Erstellen identitätsbasierter Richtlinien finden Sie unter Definieren benutzerdefinierter IAM-Berechtigungen mit vom Kunden verwalteten Richtlinien im IAM-Benutzerhandbuch.

Identitätsbasierte Richtlinien können Inline-Richtlinien (direkt in eine einzelne Identität eingebettet) oder verwaltete Richtlinien (eigenständige Richtlinien, die mit mehreren Identitäten verbunden sind) sein. Informationen dazu, wie Sie zwischen verwalteten und Inline-Richtlinien wählen, finden Sie unter Choose between managed policies and inline policies im IAM-Benutzerhandbuch.

Ressourcenbasierte Richtlinien

Ressourcenbasierte Richtlinien sind JSON-Richtliniendokumente, die Sie an eine Ressource anfügen. Beispiele hierfür sind Vertrauensrichtlinien für IAM-Rollen und Amazon S3-Bucket-Richtlinien. In Services, die ressourcenbasierte Richtlinien unterstützen, können Service-Administratoren sie verwenden, um den Zugriff auf eine bestimmte Ressource zu steuern. Sie müssen in einer ressourcenbasierten Richtlinie einen Prinzipal angeben.

Ressourcenbasierte Richtlinien sind Richtlinien innerhalb dieses Diensts. Sie können AWS verwaltete Richtlinien von IAM nicht in einer ressourcenbasierten Richtlinie verwenden.

Zugriffskontrolllisten () ACLs

Zugriffskontrolllisten (ACLs) steuern, welche Principals (Kontomitglieder, Benutzer oder Rollen) über Zugriffsberechtigungen für eine Ressource verfügen. ACLs ähneln ressourcenbasierten Richtlinien, verwenden jedoch nicht das JSON-Richtliniendokumentformat.

Amazon S3 und Amazon VPC sind Beispiele für Dienste, die Unterstützung ACLs bieten. AWS WAF Weitere Informationen finden Sie unter Übersicht über ACLs die Zugriffskontrollliste (ACL) im Amazon Simple Storage Service Developer Guide.

Weitere Richtlinientypen

AWS unterstützt zusätzliche Richtlinientypen, mit denen die maximalen Berechtigungen festgelegt werden können, die durch gängigere Richtlinientypen gewährt werden:

- Berechtigungsgrenzen – Eine Berechtigungsgrenze legt die maximalen Berechtigungen fest, die eine identitätsbasierte Richtlinie einer IAM-Entität erteilen kann. Weitere Informationen finden Sie unter Berechtigungsgrenzen für IAM-Entitäten im -IAM-Benutzerhandbuch.
- Richtlinien zur Dienstkontrolle (SCPs) — Geben Sie die maximalen Berechtigungen für eine Organisation oder Organisationseinheit in an AWS Organizations. Weitere Informationen finden Sie unter Service-Kontrollrichtlinien im AWS Organizations -Benutzerhandbuch.
- Richtlinien zur Ressourcenkontrolle (RCPs) — Legen Sie die maximal verfügbaren Berechtigungen für Ressourcen in Ihren Konten fest. Weitere Informationen finden Sie im AWS Organizations Benutzerhandbuch unter Richtlinien zur Ressourcenkontrolle (RCPs).
- Sitzungsrichtlinien – Sitzungsrichtlinien sind erweiterte Richtlinien, die als Parameter übergeben werden, wenn Sie eine temporäre Sitzung für eine Rolle oder einen Verbundbenutzer erstellen. Weitere Informationen finden Sie unter Sitzungsrichtlinien im IAM-Benutzerhandbuch.

Mehrere Richtlinientypen

Wenn für eine Anfrage mehrere Arten von Richtlinien gelten, sind die daraus resultierenden Berechtigungen schwieriger zu verstehen. Informationen darüber, wie AWS bestimmt wird, ob eine Anfrage zulässig ist, wenn mehrere Richtlinientypen betroffen sind, finden Sie unter Bewertungslogik für Richtlinien im IAM-Benutzerhandbuch.

Wie AWS-Services arbeiten Sie mit IAM

Einen allgemeinen Überblick darüber, wie die meisten IAM-Funktionen AWS-Services funktionieren, finden Sie im AWS IAM-Benutzerhandbuch unter Dienste, die mit IAM funktionieren.

Informationen zur Verwendung bestimmter Dienste AWS-Service mit IAM finden Sie im Abschnitt Sicherheit im Benutzerhandbuch des jeweiligen Dienstes.

Problembehebung bei AWS Identität und Zugriff

Verwenden Sie die folgenden Informationen, um häufig auftretende Probleme zu diagnostizieren und zu beheben, die bei der Arbeit mit AWS und IAM auftreten können.

Themen

- Ich bin nicht berechtigt, eine Aktion durchzuführen in AWS
- Ich bin nicht berechtigt, iam auszuführen: PassRole
- Ich möchte Personen außerhalb von mir den Zugriff AWS-Konto auf meine AWS Ressourcen ermöglichen

Ich bin nicht berechtigt, eine Aktion durchzuführen in AWS

Wenn Sie eine Fehlermeldung erhalten, dass Sie nicht zur Durchführung einer Aktion berechtigt sind, müssen Ihre Richtlinien aktualisiert werden, damit Sie die Aktion durchführen können.

Der folgende Beispielfehler tritt auf, wenn der IAM-Benutzer `mateojackson` versucht, über die Konsole Details zu einer fiktiven *`my-example-widget`*-Ressource anzuzeigen, jedoch nicht über `awes:`*`GetWidget`*-Berechtigungen verfügt.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 awes:GetWidget on resource: my-example-widget
```

In diesem Fall muss die Richtlinie für den Benutzer `mateojackson` aktualisiert werden, damit er mit der `awes:`*`GetWidget`*-Aktion auf die *`my-example-widget`*-Ressource zugreifen kann.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich bin nicht berechtigt, iam auszuführen: PassRole

Wenn Sie die Fehlermeldung erhalten, dass Sie nicht zum Durchführen der `iam:PassRole`-Aktion autorisiert sind, müssen Ihre Richtlinien aktualisiert werden, um eine Rolle an AWSübergeben zu können.

Einige AWS-Services ermöglichen es Ihnen, eine bestehende Rolle an diesen Dienst zu übergeben, anstatt eine neue Servicerolle oder eine dienstverknüpfte Rolle zu erstellen. Hierzu benötigen Sie Berechtigungen für die Übergabe der Rolle an den Dienst.

Der folgende Beispielfehler tritt auf, wenn ein IAM-Benutzer mit dem Namen `marymajor` versucht, die Konsole zu verwenden, um eine Aktion in AWS auszuführen. Die Aktion erfordert jedoch, dass der Service über Berechtigungen verfügt, die durch eine Servicerolle gewährt werden. Mary besitzt keine Berechtigungen für die Übergabe der Rolle an den Dienst.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In diesem Fall müssen die Richtlinien von Mary aktualisiert werden, um die Aktion `iam:PassRole` ausführen zu können.

Wenn Sie Hilfe benötigen, wenden Sie sich an Ihren AWS Administrator. Ihr Administrator hat Ihnen Ihre Anmeldeinformationen zur Verfügung gestellt.

Ich möchte Personen außerhalb von mir den Zugriff AWS-Konto auf meine AWS Ressourcen ermöglichen

Sie können eine Rolle erstellen, mit der Benutzer in anderen Konten oder Personen außerhalb Ihrer Organisation auf Ihre Ressourcen zugreifen können. Sie können festlegen, wem die Übernahme der Rolle anvertraut wird. Für Dienste, die ressourcenbasierte Richtlinien oder Zugriffskontrolllisten (ACLs) unterstützen, können Sie diese Richtlinien verwenden, um Personen Zugriff auf Ihre Ressourcen zu gewähren.

Weitere Informationen dazu finden Sie hier:

- Informationen darüber, ob diese Funktionen AWS unterstützt werden, finden Sie unter. Wie AWS-Services arbeiten Sie mit IAM
- Informationen dazu, wie Sie Zugriff auf Ihre Ressourcen gewähren können, AWS-Konten die Ihnen gehören, finden Sie im IAM-Benutzerhandbuch unter Gewähren des Zugriffs auf einen IAM-Benutzer in einem anderen AWS-Konto , den Sie besitzen.
- Informationen dazu, wie Sie Dritten Zugriff auf Ihre Ressourcen gewähren können AWS-Konten, finden Sie AWS-Konten im IAM-Benutzerhandbuch unter Gewähren des Zugriffs für Dritte.
- Informationen dazu, wie Sie über einen Identitätsverbund Zugriff gewähren, finden Sie unter Gewähren von Zugriff für extern authentifizierte Benutzer (Identitätsverbund) im IAM-Benutzerhandbuch.
- Informationen zum Unterschied zwischen der Verwendung von Rollen und ressourcenbasierten Richtlinien für den kontoübergreifenden Zugriff finden Sie unter Kontoübergreifender Ressourcenzugriff in IAM im IAM-Benutzerhandbuch.

Konformitätsvalidierung für dieses AWS Produkt oder diese Dienstleistung

Informationen darüber, ob AWS-Service ein AWS-Services in den Geltungsbereich bestimmter Compliance-Programme fällt, finden Sie unter Umfang nach Compliance-Programm AWS-Services unter . Wählen Sie dort das Compliance-Programm aus, an dem Sie interessiert sind. Allgemeine Informationen finden Sie unter AWS Compliance-Programme AWS .

Sie können Prüfberichte von Drittanbietern unter herunterladen AWS Artifact. Weitere Informationen finden Sie unter Berichte herunterladen unter .

Ihre Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services hängt von der Vertraulichkeit Ihrer Daten, den Compliance-Zielen Ihres Unternehmens und den geltenden Gesetzen und Vorschriften ab. Weitere Informationen zu Ihrer Verantwortung für die Einhaltung der Vorschriften bei der Nutzung AWS-Services finden Sie in der AWS Sicherheitsdokumentation.

Dieses AWS Produkt oder dieser Service folgt dem Modell der gemeinsamen Verantwortung in Bezug auf die spezifischen Amazon Web Services (AWS) -Services, die es unterstützt. Informationen zur AWS Servicesicherheit finden Sie auf der Seite mit der Dokumentation zur AWS Servicesicherheit und den AWS Services, für die das AWS Compliance-Programm zur Einhaltung der Vorschriften zuständig ist.

Resilienz für dieses AWS Produkt oder diese Dienstleistung

Die AWS globale Infrastruktur basiert auf AWS-Regionen Availability Zones.

AWS-Regionen bieten mehrere physisch getrennte und isolierte Availability Zones, die über Netzwerke mit niedriger Latenz, hohem Durchsatz und hoher Redundanz miteinander verbunden sind.

Mithilfe von Availability Zones können Sie Anwendungen und Datenbanken erstellen und ausführen, die automatisch Failover zwischen Zonen ausführen, ohne dass es zu Unterbrechungen kommt. Availability Zones sind besser verfügbar, fehlertoleranter und skalierbarer als herkömmliche Infrastrukturen mit einem oder mehreren Rechenzentren.

Weitere Informationen zu AWS Regionen und Availability Zones finden Sie unter AWS Globale Infrastruktur.

Dieses AWS Produkt oder dieser Service folgt dem Modell der gemeinsamen Verantwortung in Bezug auf die spezifischen Amazon Web Services (AWS) -Services, die es unterstützt. Informationen zur AWS Servicesicherheit finden Sie auf der Seite mit der Dokumentation zur AWS Servicesicherheit und den AWS Services, für die das AWS Compliance-Programm zur Einhaltung der Vorschriften zuständig ist.

Infrastruktursicherheit für dieses AWS Produkt oder diese Dienstleistung

Dieses AWS Produkt oder dieser Dienst verwendet Managed Services und ist daher durch die AWS globale Netzwerksicherheit geschützt. Informationen zu AWS Sicherheitsdiensten und zum AWS Schutz der Infrastruktur finden Sie unter AWS Cloud-Sicherheit. Informationen zum Entwerfen Ihrer AWS Umgebung unter Verwendung der bewährten Methoden für die Infrastruktursicherheit finden Sie unter Infrastructure Protection in Security Pillar AWS Well-Architected Framework.

Sie verwenden AWS veröffentlichte API-Aufrufe, um über das Netzwerk auf dieses AWS Produkt oder diesen Service zuzugreifen. Kunden müssen Folgendes unterstützen:

- Transport Layer Security (TLS). Wir benötigen TLS 1.2 und empfehlen TLS 1.3.
- Verschlüsselungs-Suiten mit Perfect Forward Secrecy (PFS) wie DHE (Ephemeral Diffie-Hellman) oder ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). Die meisten modernen Systeme wie Java 7 und höher unterstützen diese Modi.

Außerdem müssen Anforderungen mit einer Zugriffsschlüssel-ID und einem geheimen Zugriffsschlüssel signiert sein, der einem IAM-Prinzipal zugeordnet ist. Alternativ können Sie mit AWS -Security-Token-Service (AWS STS) temporäre Sicherheitsanmeldeinformationen erstellen, um die Anforderungen zu signieren.

Dieses AWS Produkt oder dieser Service folgt dem Modell der gemeinsamen Verantwortung in Bezug auf die spezifischen Amazon Web Services (AWS) -Services, die es unterstützt. Informationen zur AWS Servicesicherheit finden Sie auf der Seite mit der Dokumentation zur AWS Servicesicherheit und den AWS Services, für die das AWS Compliance-Programm zur Einhaltung der Vorschriften zuständig ist.

Erzwingen einer TLS-Mindestversion in der AWS SDK für C++

Um die Sicherheit bei der Kommunikation mit AWS Diensten zu erhöhen, sollten Sie das SDK for C++ so konfigurieren, dass es TLS 1.2 oder höher verwendet. Wir empfehlen die Verwendung von TLS 1.3.

Die AWS SDK für C++ ist eine plattformübergreifende Bibliothek. Sie können Ihre Anwendung auf den gewünschten Plattformen erstellen und ausführen. Verschiedene Plattformen hängen möglicherweise von unterschiedlichen zugrunde liegenden HTTP-Clients ab.

Standardmäßig verwenden macOS, Linux, Android und andere Nicht-Windows-Plattformen libcurl. Wenn die libcurl-Version neuer als 7.34.0 ist, ist TLS 1.0 die Mindestversion, die von den zugrunde liegenden HTTP-Clients verwendet wird.

Für Windows ist die Standardbibliothek. WinHttp Windows entscheidet unter den verfügbaren Protokollen TLS 1.0, TLS 1.1, TLS 1.2 und TLS 1.3, welches Protokoll tatsächlich verwendet werden soll. Win INet und IXMLHttpRequest2 sind die anderen beiden Optionen, die unter Windows verfügbar sind. Sie können Ihre Anwendung so konfigurieren, dass sie die Standardbibliothek während CMake und zur Laufzeit ersetzt. Für diese beiden HTTP-Clients entscheidet Windows auch über das sichere Protokoll.

Das bietet AWS SDK für C++ auch die Flexibilität, die Standard-HTTP-Clients zu überschreiben. Sie können beispielsweise libcurl erzwingen oder beliebige HTTP-Clients verwenden, indem Sie eine benutzerdefinierte HTTP-Client-Factory verwenden. Um also TLS 1.2 als Mindestversion zu verwenden, müssen Sie sich der HTTP-Clientbibliothek bewusst sein, die Sie verwenden.

Erzwingen Sie eine bestimmte TLS-Version mit libcurl auf allen Plattformen

In diesem Abschnitt wird davon ausgegangen, dass libcurl als Abhängigkeit für die HTTP-Protokollunterstützung verwendet AWS SDK für C++ wird. Um die TLS-Version explizit anzugeben, benötigen Sie mindestens die libcurl-Version 7.34.0. Darüber hinaus müssen Sie möglicherweise den Quellcode von ändern AWS SDK für C++ und ihn dann neu erstellen.

Das folgende Verfahren zeigt Ihnen, wie Sie diese Aufgaben ausführen.

Um TLS 1.2 mit libcurl durchzusetzen

1. Stellen Sie sicher, dass Ihre Installation von libcurl mindestens Version 7.34.0 ist.
2. Laden Sie den Quellcode für das Formular herunter. AWS SDK für C++ GitHub

3. Öffne `aws-cpp-sdk-core/source/http/curl/CurlHttpClient.cpp` und finde die folgenden Codezeilen.

```
#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. Ändern Sie bei Bedarf den letzten Parameter im Funktionsaufruf wie folgt.

```
#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. Wenn Sie die obigen Codeänderungen vorgenommen haben, erstellen und installieren Sie den Code AWS SDK für C++ gemäß den Anweisungen unter https://github.com/aws/aws-sdk-cpp#building-the-sdk.
6. Aktivieren Sie die Option für den Service-Client in Ihrer Anwendung `verifySSL` in seiner Client-Konfiguration, falls diese Option nicht bereits aktiviert ist.

Um TLS 1.3 mit libcurl durchzusetzen

Um TLS 1.3 durchzusetzen, folgen Sie den Schritten im vorherigen Abschnitt und setzen Sie die `CURL_SSLVERSION_TLSv1_3` Option anstelle von. `CURL_SSLVERSION_TLSv1_2`

Erzwingen Sie eine bestimmte TLS-Version unter Windows

Die folgenden Verfahren zeigen Ihnen, wie Sie TLS 1.2 oder TLS 1.3 mit WinHttpINet, Win oder erzwingen IXMLHTTPRequest2.

Voraussetzung: Stellen Sie fest, ob Windows TLS unterstützt

- Ermitteln Sie, welche Version des TLS-Protokolls für Ihr System verfügbar ist, wie unter https://docs.microsoft.com/en-us/windows/win32/secauthn/protocolsin-tls-ssl- schannel-ssp - beschrieben.
- Wenn Sie Windows 7 SP1 oder Windows Server 2008 R2 verwenden, müssen Sie sicherstellen SP1, dass die TLS 1.2-Unterstützung in der Registrierung aktiviert ist, wie unter https://

docs.microsoft.com/en-us/windows - -registry-settings #tls -12 beschrieben. server/security/tls/tls Wenn Sie eine frühere Verteilung ausführen, müssen Sie Ihr Betriebssystem aktualisieren.

Um TLS 1.2 oder TLS 1.3 zu erzwingen mit WinHttp

WinHttp stellt eine API bereit, um die akzeptablen sicheren Protokolle explizit festzulegen. Um dies jedoch zur Laufzeit konfigurierbar zu machen, müssen Sie den Quellcode von ändern AWS SDK für C++ und ihn dann neu erstellen.

1. Laden Sie den Quellcode für das Formular AWS SDK für C++ herunter GitHub.
2. Öffne `aws-cpp-sdk-core/source/http/windows/WinHttpSyncHttpClient.cpp` und finde die folgenden Codezeilen.

```
#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());
}
```

   Das `WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_3` Optionsflag ist definiert, wenn TLS 1.3 auf dem aktuellen Build-System vorhanden ist. Weitere Informationen finden Sie unter WINHTTP_OPTION_SECURE_PROTOCOLS und Unterstützung der TLS-Protokollversion auf der Microsoft-Website.

3. Wählen Sie eine der folgenden Optionen:

   - So erzwingen Sie TLS 1.2:

     Ändern Sie `#else` gemäß der Direktive den Wert der `flags` Variablen wie folgt.

```
DWORD flags = WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_2;
```

- Um TLS 1.3 durchzusetzen:

  Ändern Sie `#if defined(WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_3)` gemäß der Direktive den Wert der `flags` Variablen wie folgt.

  ```
  DWORD flags = WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_3;
  ```

4. Wenn Sie die oben genannten Codeänderungen vorgenommen haben, erstellen und installieren Sie den Code AWS SDK für C++ gemäß den Anweisungen unter https://github.com/aws/aws-sdk-cpp# building-the-sdk.
5. Aktivieren Sie die Option für den Service-Client in Ihrer Anwendung `verifySSL` in seiner Client-Konfiguration, falls diese Option nicht bereits aktiviert ist.

Um TLS 1.2 mit Win durchzusetzen INet und IXMLHTTPRequest2

Es gibt keine API, um das sichere Protokoll für Win INet und IXMLHTTPRequest2 Bibliotheken anzugeben. Das AWS SDK für C++ verwendet also die Standardeinstellung für das Betriebssystem. Sie können die Windows-Registrierung aktualisieren, um die Verwendung von TLS 1.2 zu erzwingen, wie im folgenden Verfahren gezeigt. Beachten Sie jedoch, dass das Ergebnis eine globale Änderung ist, die sich auf alle Anwendungen auswirkt, die von Schannel abhängig sind.

1. Öffnen Sie den Registrierungseditor und gehen Sie zu. `Computer\HKEY_LOCAL_MACHINE \SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols`
2. Falls sie noch nicht existieren, erstellen Sie die folgenden Unterschlüssel: `TLS 1.0`,`TLS 1.1`, und`TLS 1.2`.
3. Erstellen Sie unter jedem der Unterschlüssel einen Unterschlüssel und einen `Client` Unterschlüssel. `Server`
4. Erstellen Sie die folgenden Schlüssel und Werte.

```
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
```

Beachten Sie, dass dies `TLS 1.2\Client\Enabled` der einzige Schlüssel ist, der auf 1 gesetzt ist. Wenn Sie diesen Schlüssel auf 1 setzen, wird TLS 1.2 als einzig akzeptables sicheres Protokoll durchgesetzt.

Um TLS 1.3 mit Win INet durchzusetzen und IXMLHTTPRequest2

Es gibt keine API, um das sichere Protokoll für Win INet und IXMLHTTPRequest2 Bibliotheken anzugeben. Das AWS SDK für C++ verwendet also die Standardeinstellung für das Betriebssystem. Sie können die Windows-Registrierung aktualisieren, um die Verwendung von TLS 1.3 zu erzwingen, wie im folgenden Verfahren gezeigt. Beachten Sie jedoch, dass das Ergebnis eine globale Änderung ist, die sich auf alle Anwendungen auswirkt, die von Schannel abhängig sind.

1. Öffnen Sie den Registrierungseditor und gehen Sie zu. `Computer\HKEY_LOCAL_MACHINE \SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols`
2. Falls sie noch nicht existieren, erstellen Sie die folgenden Unterschlüssel: `TLS 1.0`,`TLS 1.1`, `TLS 1.2` und`TLS 1.3`.
3. Erstellen Sie unter jedem der Unterschlüssel einen Unterschlüssel und einen `Client` Unterschlüssel. `Server`
4. Erstellen Sie die folgenden Schlüssel und Werte.

```
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
```

Beachten Sie, dass dies `TLS 1.3\Client\Enabled` der einzige Schlüssel ist, der auf 1 gesetzt ist. Wenn Sie diesen Schlüssel auf 1 setzen, wird TLS 1.3 als einzig akzeptables sicheres Protokoll durchgesetzt.

Migration des Amazon S3 S3-Verschlüsselungsclients (V1 auf V2)

> **Note**
>
> Wenn Sie Version 2 des Amazon S3 S3-Verschlüsselungsclients verwenden und zu Version 3 migrieren möchten, finden Sie weitere Informationen unterMigration des Amazon S3 S3-Verschlüsselungsclients (V2 auf V3).

In diesem Thema wird gezeigt, wie Sie Ihre Anwendungen von Version 1 (V1) des Amazon Simple Storage Service (Amazon S3) -Verschlüsselungsclients auf Version 2 (V2) migrieren und die Anwendungsverfügbarkeit während des gesamten Migrationsprozesses sicherstellen.

Überblick über die Migration

Diese Migration erfolgt in zwei Phasen:

1. Aktualisieren Sie bestehende Clients, damit sie neue Formate lesen können. Stellen Sie zunächst eine aktualisierte Version von AWS SDK für C++ für Ihre Anwendung bereit. Dadurch können bestehende V1-Verschlüsselungsclients Objekte entschlüsseln, die von den neuen V2-Clients geschrieben wurden. Wenn Ihre Anwendung mehrere verwendet AWS SDKs, müssen Sie jedes SDK separat aktualisieren.

2. Migrieren Sie Verschlüsselungs- und Entschlüsselungsclients auf V2. Sobald alle Ihre V1-Verschlüsselungsclients neue Formate lesen können, können Sie Ihre vorhandenen Verschlüsselungs- und Entschlüsselungsclients auf ihre jeweiligen V2-Versionen migrieren.

Aktualisieren Sie bestehende Clients, um neue Formate zu lesen

Sie müssen zuerst Ihre vorhandenen Clients auf die neueste SDK-Version aktualisieren. Nach Abschluss dieses Schritts können die V1-Clients Ihrer Anwendung Objekte entschlüsseln, die mit V2-Verschlüsselungsclients verschlüsselt wurden, ohne die Codebasis Ihrer Anwendung aktualisieren zu müssen.

Erstellen und installieren Sie die neueste Version von AWS SDK für C++

Anwendungen, die das SDK aus der Quelle nutzen

Wenn Sie die Quelle erstellen und installieren, laden Sie die SDK-Quelle AWS SDK für C++ von `aws/aws-sdk-cpp`on herunter oder klonen Sie sie GitHub. Wiederholen Sie dann Ihre normalen Build- und Installationsschritte.

Wenn Sie ein Upgrade AWS SDK für C++ von einer Version vor 1.8.x durchführen, finden Sie in diesem CHANGELOG die wichtigsten Änderungen, die in den einzelnen Hauptversionen eingeführt wurden. Weitere Informationen zum Erstellen und Installieren von finden Sie unter AWS SDK für C++. Den AWS SDK für C++ aus dem Quellcode holen

Anwendungen, die das SDK von Vcpkg nutzen

Wenn Ihre Anwendung Vcpkg zur Nachverfolgung von SDK-Updates verwendet, verwenden Sie einfach Ihre bestehende Vcpkg-Upgrade-Methode, um das SDK auf die neueste Version zu aktualisieren. Beachten Sie, dass es zwischen der Veröffentlichung einer Version und der Verfügbarkeit über einen Paketmanager eine Verzögerung gibt. Die neueste Version ist immer verfügbar, wenn sie von der Quelle aus installiert wird.

Sie können den folgenden Befehl ausführen, um das Paket zu aktualisieren`aws-sdk-cpp`:

```
vcpkg upgrade aws-sdk-cpp
```

Und überprüfen Sie die Version des Pakets`aws-sdk-cpp`:

```
vcpkg list aws-sdk-cpp
```

Die Version sollte mindestens 1.8.24 sein.

Weitere Hinweise zur Verwendung von Vcpkg mit dem finden Sie unter. AWS SDK für C++AWS SDK für C++ Von einem Paketmanager bekommen

Erstellen, installieren und implementieren Sie Ihre Anwendungen

Wenn Ihre Anwendung statisch mit dem verknüpft wird AWS SDK für C++, sind keine Codeänderungen in Ihrer Anwendung erforderlich. Sie müssen Ihre Anwendung jedoch erneut erstellen, um die neuesten SDK-Änderungen zu verwenden. Dieser Schritt ist für dynamisches Verknüpfen nicht erforderlich.

Nachdem Sie die Abhängigkeitsversion Ihrer Anwendung aktualisiert und die Funktionalität der Anwendung überprüft haben, fahren Sie mit der Bereitstellung Ihrer Anwendung in Ihrer Flotte fort.

Sobald die Anwendungsbereitstellung abgeschlossen ist, können Sie mit der nächsten Phase der Migration Ihrer Anwendung zur Verwendung der V2-Verschlüsselungs- und Entschlüsselungsclients fortfahren.

Migrieren Sie die Verschlüsselungs- und Entschlüsselungsclients zu V2

Die folgenden Schritte zeigen Ihnen, wie Sie Ihren Code erfolgreich von V1 auf V2 des Amazon S3 S3-Verschlüsselungsclients migrieren. Da Codeänderungen erforderlich sind, müssen Sie Ihre Anwendung neu erstellen, unabhängig davon, ob sie statisch oder dynamisch mit dem AWS SDK für C++ verknüpft wird.

Verwendung neuer Verschlüsselungsmaterialien

Mit Amazon S3 S3-Verschlüsselungsclients V2 und der V2-Kryptokonfiguration sind die folgenden Verschlüsselungsmaterialien veraltet:

- `SimpleEncryptionMaterials`
- `KMSEncryptionMaterials`

Sie wurden durch die folgenden sicheren Verschlüsselungsmaterialien ersetzt:

- `SimpleEncryptionMaterialsWithGCMAAD`
- `KMSWithContextEncryptionMaterials`

Die folgenden Codeänderungen sind erforderlich, um einen V2 S3-Verschlüsselungsclient zu erstellen:

- Wenn Sie **`KMSEncryptionMaterials`** beim Erstellen eines S3-Verschlüsselungsclients Folgendes verwenden:
    - Wenn Sie einen V2-S3-Verschlüsselungsclient erstellen, `KMSEncryptionMaterials` ersetzen Sie ihn durch `KMSWithContextEncryptionMaterials` und geben Sie ihn in der V2-Kryptokonfiguration an.
    - Wenn Sie ein Objekt mit V2-Amazon S3-Verschlüsselungsclients platzieren, müssen Sie explizit eine Zeichenfolgen-Kontext-Map als KMS-Kontext für die Verschlüsselung des CEK angeben. Dies könnte eine leere Map sein.

- Wenn Sie **`SimpleEncryptionMaterials`** beim Erstellen eines S3-Verschlüsselungsclients Folgendes verwenden:
  - Wenn Sie einen V2-Amazon S3-Verschlüsselungsclient erstellen, `SimpleEncryptionMaterials` ersetzen Sie ihn durch `SimpleEncryptionMaterialsWithGCMAAD` und geben Sie ihn in der V2-Kryptokonfiguration an.
  - Wenn Sie ein Objekt mit V2-Amazon S3-Verschlüsselungsclients platzieren, müssen Sie explizit eine leere Zeichenfolgen-Kontextmap angeben, da das SDK andernfalls einen Fehler zurückgibt.

Beispiel: Verwendung des KMS/KMSWithContext Key-Wrap-Algorithmus

Vor der Migration (KMS-Schlüsselumbruch)

```
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);
```

Nach der Migration (KMSWithKontext-Schlüsselumbruch)

```
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
 */);
```

Beispiel: Verwendung des AES/AES-GCM-Schlüsselumbruchalgorithmus

Vor der Migration (AES-Schlüsselumbruch)

```
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);
```

Nach der Migration (AES-GCM-Schlüsselumbruch)

```
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 */);
```

Weitere Beispiele

Die folgenden Beispiele zeigen, wie bestimmte Anwendungsfälle im Zusammenhang mit einer Migration von V1 zu V2 behandelt werden können.

Mit älteren Amazon S3 S3-Verschlüsselungsclients verschlüsselte Objekte entschlüsseln

Standardmäßig können Sie den Amazon S3 S3-Verschlüsselungsclient V2 nicht zum Entschlüsseln von Objekten verwenden, die mit veralteten Key-Wrap-Algorithmen oder veralteten Inhaltskryptoschemas verschlüsselt wurden.

Die folgenden Algorithmen zum Umbrechen von Schlüsseln sind veraltet:

- `KMS`
- `AES_KEY_WRAP`

Und die folgenden Inhaltskryptoschemas sind veraltet:

- `CBC`
- `CTR`

Wenn Sie ältere Amazon S3 S3-Verschlüsselungsclients verwenden, AWS SDK für C++ um die Objekte zu verschlüsseln, verwenden Sie wahrscheinlich die veralteten Methoden, wenn:

- Sie haben oder verwendet. `SimpleEncryptionMaterials` `KMSEncryptionMaterials`
- Sie haben `ENCRYPTION_ONLY` wie `Crypto Mode` in Ihrer Krypto-Konfiguration verwendet.

Um den Amazon S3 S3-Verschlüsselungsclient V2 zum Entschlüsseln von Objekten zu verwenden, die mit veralteten Schlüsselumbruchalgorithmen oder veralteten Inhaltskryptoschemas verschlüsselt wurden, müssen Sie den Standardwert von `SecurityProfile` in der V2-Kryptokonfiguration von bis überschreiben. `V2` `V2_AND_LEGACY`

Beispiel

Vor der Migration

```
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);
```

Nach der Migration

```
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);
```

Objekte mit Range entschlüsseln

Mit älteren Amazon S3 S3-Verschlüsselungsclients können Sie einen Bytebereich angeben, der beim Entschlüsseln eines S3-Objekts empfangen werden soll. Im Amazon S3 S3-Verschlüsselungsclient V2 ist diese Funktion `DISABLED` standardmäßig verfügbar. Daher müssen Sie den Standardwert `RangeGetMode` von `DISABLED` bis `ALL` in der V2-Kryptokonfiguration überschreiben.

Beispiel

Vor der Migration

```
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);
```

Nach der Migration

```
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);
```

Entschlüsseln Sie Objekte mit einem beliebigen CMK

Bei der Entschlüsselung von Objekten`KMSWithContextEncryptionMaterials`, mit denen verschlüsselt wurden, können Amazon S3 S3-V2-Verschlüsselungsclients KMS die Suche nach dem richtigen CMK ermöglichen, indem sie einen leeren Masterschlüssel bereitstellen. Diese Funktion ist `DISABLED` standardmäßig aktiviert. Sie müssen es explizit konfigurieren, indem Sie Ihre KMS-Verschlüsselungsmaterialien anfordern. `SetKMSDecryptWithAnyCMK(true)`

Beispiel

Vor der Migration

```
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);
```

Nach der Migration

```
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);
```

Den vollständigen Code für all diese Migrationsszenarien finden Sie im Amazon S3 S3-Verschlüsselungsbeispiel auf Github.

Migration des Amazon S3 S3-Verschlüsselungsclients (V2 auf V3)

> **Note**
>
> Wenn Sie Version 1 des Amazon S3 S3-Verschlüsselungsclients verwenden, müssen Sie zuerst zu V2 migrieren, bevor Sie zu V3 migrieren. Siehe Migration des Amazon S3 S3-Verschlüsselungsclients (V1 auf V2).

In diesem Thema erfahren Sie, wie Sie Ihre Anwendungen von Version 2 (V2) auf Version 3 (V3) des Amazon Simple Storage Service (Amazon S3) -Verschlüsselungsclients migrieren und die Anwendungsverfügbarkeit während des gesamten Migrationsprozesses sicherstellen. In Version 3 werden der `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` Algorithmus und die Commitment-Richtlinien eingeführt, um die Sicherheit zu erhöhen, indem sie vor der Manipulation von Datenschlüsseln in den Befehlsdateien schützen.

Überblick über die Migration

Diese Migration erfolgt in zwei Phasen:

1. Aktualisieren Sie bestehende Clients, damit sie neue Formate lesen können. Stellen Sie zunächst eine aktualisierte Version von AWS SDK für C++ für Ihre Anwendung bereit. Dadurch können bestehende V2-Verschlüsselungsclients Objekte entschlüsseln, die von den neuen V3-Clients geschrieben wurden. Wenn Ihre Anwendung mehrere verwendet AWS SDKs, müssen Sie jedes SDK separat aktualisieren.

2. Migrieren Sie Verschlüsselungs- und Entschlüsselungsclients auf V3. Sobald alle Ihre V2-Verschlüsselungsclients neue Formate lesen können, können Sie Ihre vorhandenen Verschlüsselungs- und Entschlüsselungsclients auf ihre jeweiligen V3-Versionen migrieren.

V3-Konzepte verstehen

Version 3 des Amazon S3 S3-Verschlüsselungsclients bietet neue Sicherheitsfunktionen, die den Schutz vor Manipulation von Datenschlüsseln verbessern. Das Verständnis dieser Konzepte ist für eine erfolgreiche Migration unerlässlich.

Politik des Engagements

Commitment-Richtlinien steuern, wie der Verschlüsselungsclient bei Verschlüsselungs- und Entschlüsselungsvorgängen mit der Schlüsselzusage umgeht. V3 bietet drei Richtlinienoptionen zur Unterstützung verschiedener Migrationsszenarien und Sicherheitsanforderungen:

`FORBID_ENCRYPT_ALLOW_DECRYPT`

Verschlüsselungsverhalten: Verschlüsselt Objekte ohne Schlüsselbindung und verwendet dieselben Algorithmen wie V2.

Entschlüsselungsverhalten: Ermöglicht die Entschlüsselung von Objekten, die mit und ohne Schlüsselbindung verschlüsselt wurden.

Auswirkungen auf die Sicherheit: Diese Richtlinie erzwingt keine Schlüsselbindung und ermöglicht möglicherweise die Manipulation des verschlüsselten Datenschlüssels in den Befehlsdateien. Verwenden Sie diese Richtlinie nur in der ersten Migrationsphase, wenn Sie V2-Clients benötigen, um neu verschlüsselte Objekte zu lesen.

Versionskompatibilität: Mit dieser Richtlinie verschlüsselte Objekte können von allen V2- und V3-Implementierungen gelesen werden.

`REQUIRE_ENCRYPT_ALLOW_DECRYPT` (Standard)

Verschlüsselungsverhalten: Verschlüsselt Objekte mit Schlüsselzusage mithilfe des `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` Algorithmus.

Entschlüsselungsverhalten: Ermöglicht die Entschlüsselung sowohl von Objekten, die mit Key Commitment verschlüsselt wurden, als auch von Objekten, die ohne Key Commitment verschlüsselt wurden.

Auswirkungen auf die Sicherheit: Diese Richtlinie bietet hohe Sicherheit für neu verschlüsselte Objekte und gewährleistet gleichzeitig die Abwärtskompatibilität beim Lesen älterer Objekte. Dies ist die empfohlene Richtlinie für die meisten Migrationsszenarien.

Versionskompatibilität: Mit dieser Richtlinie verschlüsselte Objekte können nur von V3 und den neuesten V2-Implementierungen gelesen werden. V2-Clients können diese Objekte nicht entschlüsseln. V3-Clients, die diese Richtlinie verwenden, können jedoch weiterhin Objekte entschlüsseln, die von V2-Clients verschlüsselt wurden.

`REQUIRE_ENCRYPT_REQUIRE_DECRYPT`

Verschlüsselungsverhalten: Verschlüsselt Objekte mit Schlüsselzusage mithilfe des `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` Algorithmus.

Entschlüsselungsverhalten: Erlaubt nur die Entschlüsselung von Objekten, die mit Schlüsselzuweisung verschlüsselt wurden. Weist Objekte ab, die ohne Schlüsselzuweisung verschlüsselt wurden.

Auswirkungen auf die Sicherheit: Diese Richtlinie bietet ein Höchstmaß an Sicherheit, indem sie bei allen Vorgängen eine Schlüsselbindung erzwingt. Verwenden Sie diese Richtlinie erst, nachdem alle Objekte mit der Schlüsselzuweisung erneut verschlüsselt wurden und Sie keine älteren V1- oder V2-verschlüsselten Objekte mehr lesen müssen.

Versionskompatibilität: Mit dieser Richtlinie verschlüsselte Objekte können nur von V3 und den neuesten V2-Implementierungen gelesen werden. Darüber hinaus können Clients, die diese Richtlinie verwenden, keine Objekte entschlüsseln, die von V1- oder V2-Clients verschlüsselt wurden.

Überlegungen zur Migration: Beginnen Sie bei der Migration damit, `FORBID_ENCRYPT_ALLOW_DECRYPT` ob V2-Clients neue Objekte lesen müssen, und wechseln Sie dann zu, `REQUIRE_ENCRYPT_ALLOW_DECRYPT` sobald alle Clients auf Version 3 aktualisiert wurden. Schließlich sollten Sie dies `REQUIRE_ENCRYPT_REQUIRE_DECRYPT` erst in Betracht ziehen, nachdem alle Legacy-Objekte erneut verschlüsselt wurden.

ALG_AES_256_GCM_HKDF_ _COMMIT_KEY-Algorithmus SHA512

Der `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` Algorithmus ist ein neuer Verschlüsselungsalgorithmus, der in Version 3 eingeführt wurde und verbesserte Sicherheit für verschlüsselte Datenschlüssel bietet, die in Befehlsdateien gespeichert sind.

Auswirkung auf die Befehlsdatei: Der `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` Algorithmus wirkt sich nur auf Befehlsdateien aus, bei denen es sich um separate S3-Objekte handelt, die Verschlüsselungsmetadaten einschließlich des verschlüsselten Datenschlüssels speichern. Objekte, die Verschlüsselungsmetadaten in Objektmetadaten (der Standardspeichermethode) speichern, sind von dieser Algorithmusänderung nicht betroffen.

Schutz vor Manipulation: Der **`ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY`** Algorithmus schützt vor Manipulation von Datenschlüsseln, indem er den verschlüsselten Datenschlüssel kryptografisch an den Verschlüsselungskontext bindet. Dadurch wird verhindert, dass Angreifer einen anderen verschlüsselten Datenschlüssel in der Befehlsdatei einsetzen, was möglicherweise zur Entschlüsselung durch einen unbeabsichtigten Schlüssel führen könnte.

Versionskompatibilität: Mit dem `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` Algorithmus verschlüsselte Objekte können nur mit V3-Implementierungen und den neuesten V2-Übergangsversionen des SDK entschlüsselt werden, die V3-Entschlüsselung unterstützen.

> **Warning**
>
> Wichtig: Bevor Sie die Verschlüsselung mit dem `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` Algorithmus aktivieren (mithilfe von Richtlinien `REQUIRE_ENCRYPT_ALLOW_DECRYPT` oder `REQUIRE_ENCRYPT_REQUIRE_DECRYPT` Commitment-Richtlinien), müssen Sie sicherstellen, dass alle Clients, die diese Objekte lesen, auf Version 3 oder die neueste V2-Übergangsversion aktualisiert wurden, die V3-Entschlüsselung unterstützt. Wenn Sie nicht zuerst alle Lesegeräte aktualisieren, schlägt die Entschlüsselung neu verschlüsselter Objekte fehl.

Aktualisieren Sie bestehende Clients, um neue Formate zu lesen

Sie müssen zuerst Ihre vorhandenen Clients auf die neueste SDK-Version aktualisieren. Nach Abschluss dieses Schritts können die V2-Clients Ihrer Anwendung Objekte entschlüsseln, die mit V3-Verschlüsselungsclients verschlüsselt wurden, ohne die Codebasis Ihrer Anwendung aktualisieren zu müssen.

Erstellen und installieren Sie die neueste Version von AWS SDK für C++

Anwendungen, die das SDK aus der Quelle nutzen

Wenn Sie die Quelle erstellen und installieren, laden Sie die SDK-Quelle AWS SDK für C++ von `aws/aws-sdk-cpp`on herunter oder klonen Sie sie GitHub. Wiederholen Sie dann Ihre normalen Build- und Installationsschritte.

Wenn Sie ein Upgrade AWS SDK für C++ von einer Version vor 1.11.x durchführen, finden Sie in diesem CHANGELOG die wichtigsten Änderungen, die in den einzelnen Hauptversionen eingeführt wurden. Weitere Informationen zum Erstellen und Installieren von finden Sie unter AWS SDK für C++. Den AWS SDK für C++ aus dem Quellcode holen

Anwendungen, die das SDK von Vcpkg nutzen

Wenn Ihre Anwendung Vcpkg zur Nachverfolgung von SDK-Updates verwendet, verwenden Sie einfach Ihre bestehende Vcpkg-Upgrade-Methode, um das SDK auf die neueste Version zu aktualisieren. Beachten Sie, dass es zwischen der Veröffentlichung einer Version und der Verfügbarkeit über einen Paketmanager eine Verzögerung gibt. Die neueste Version ist immer über die Installation aus dem Quellcode verfügbar.

Sie können den folgenden Befehl ausführen, um das Paket zu aktualisieren`aws-sdk-cpp`:

```
vcpkg upgrade aws-sdk-cpp
```

Und überprüfen Sie die Version des Paket`aws-sdk-cpp`:

```
vcpkg list aws-sdk-cpp
```

Die Version sollte mindestens 1.11.x sein, um die Entschlüsselung von V3-verschlüsselten Objekten zu unterstützen.

Weitere Hinweise zur Verwendung von Vcpkg mit dem finden Sie unter. AWS SDK für C++AWS SDK für C++ Von einem Paketmanager bekommen

Erstellen, installieren und implementieren Sie Ihre Anwendungen

Wenn Ihre Anwendung statisch mit dem verknüpft wird AWS SDK für C++, sind keine Codeänderungen in Ihrer Anwendung erforderlich. Sie müssen Ihre Anwendung jedoch erneut erstellen, um die neuesten SDK-Änderungen zu verwenden. Dieser Schritt ist für dynamisches Verknüpfen nicht erforderlich.

Nachdem Sie die Abhängigkeitsversion Ihrer Anwendung aktualisiert und die Funktionalität der Anwendung überprüft haben, fahren Sie mit der Bereitstellung Ihrer Anwendung in Ihrer Flotte fort.

Sobald die Anwendungsbereitstellung abgeschlossen ist, können Sie mit der nächsten Phase der Migration Ihrer Anwendung zur Verwendung der V3-Verschlüsselungs- und Entschlüsselungsclients fortfahren.

Migrieren Sie die Verschlüsselungs- und Entschlüsselungsclients zu V3

Die folgenden Schritte zeigen Ihnen, wie Sie Ihren Code erfolgreich von V2 auf V3 des Amazon S3 S3-Verschlüsselungsclients migrieren. Da Codeänderungen erforderlich sind, müssen Sie Ihre Anwendung neu erstellen, unabhängig davon, ob sie statisch oder dynamisch mit dem AWS SDK für C++ verknüpft wird.

Verwenden von V3-Verschlüsselungsclients

V3 führt die `S3EncryptionClientV3` Klasse ein und ersetzt `CryptoConfigurationV3` die V2-Äquivalente. Die wichtigsten Unterschiede in V3 sind:

- V3 verwendet `KMSWithContextEncryptionMaterials` (wie V2), erfordert jedoch eine explizite Konfiguration in`CryptoConfigurationV3`.
- Alle `PutObject` Operationen erfordern eine Verschlüsselungskontextmap (kann leer sein).
- V3 führt Commitment-Richtlinien ein, um das Verhalten bei der Verschlüsselung und Entschlüsselung zu kontrollieren.
- Standardmäßig verschlüsselt V3 mithilfe des Algorithmus mit Key Commitment. `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY`
- Die bisherige API zur Konfiguration der Algorithmus-Entschlüsselung ändert sich von bis`config.SetSecurityProfile(SecurityProfile::V2_AND_LEGACY);`. `config.AllowLegacy();`

Beispiel: Migration von V2 zu V3 mit KMS-Verschlüsselung

Vor der Migration (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);
```

Während der Migration (V3 mit Abwärtskompatibilität)

```
// 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);
```

Nach der Migration (V3 mit zentraler Verpflichtung)

```
// 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);
```

Weitere Beispiele

Dieser Abschnitt enthält weitere Beispiele für die Konfiguration der V3-Verschlüsselungsclient-Optionen zur Unterstützung verschiedener Migrationsszenarien und -anforderungen.

Legacy-Support aktivieren

V3-Clients können von V2-Clients verschlüsselte Objekte nur dann entschlüsseln, wenn sie die Richtlinien `REQUIRE_ENCRYPT_ALLOW_DECRYPT` oder `FORBID_ENCRYPT_ALLOW_DECRYPT` Commitment verwenden. Wenn Sie jedoch Objekte entschlüsseln müssen, die mit V1-Clients verschlüsselt wurden, müssen Sie die Legacy-Unterstützung mit dieser Methode explizit aktivieren. `AllowLegacy()`

Wann sollte die Unterstützung für ältere Versionen verwendet werden:

- Sie haben Objekte in S3, die mit Version 1 des S3 Encryption Client verschlüsselt wurden.
- Sie müssen diese V1-verschlüsselten Objekte während des Migrationsprozesses mit Ihrem V3-Client lesen.
- Sie verwenden die Richtlinie `REQUIRE_ENCRYPT_ALLOW_DECRYPT` oder `FORBID_ENCRYPT_ALLOW_DECRYPT` Commitment.

> **Warning**
>
> Legacy-Support sollte während der Migration nur vorübergehend aktiviert werden. Sobald alle V1-Objekte mit V2 oder V3 erneut verschlüsselt wurden, deaktivieren Sie die Legacy-Unterstützung, um maximale Sicherheit zu gewährleisten.

Beispiel: Legacy-Unterstützung aktivieren

```
// 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);
```

Speichermethode konfigurieren

Der S3 Encryption Client kann Verschlüsselungsmetadaten auf zwei Arten speichern: als Objektmetadaten (Standard) oder in einer separaten Befehlsdatei. Sie können die Speichermethode mit der `SetStorageMethod()` Methode on konfigurieren`CryptoConfigurationV3`.

Optionen für die Speichermethode:

`METADATA` (Standard)

Verschlüsselungsmetadaten werden in den Metadaten-Headern des Objekts gespeichert. Dies ist die gebräuchlichste und praktischste Methode, da alle Verschlüsselungsinformationen zusammen mit dem Objekt selbst gespeichert werden.

Verwendungszweck: Verwenden Sie diese Methode für die meisten Szenarien. Sie vereinfacht die Objektverwaltung, da Verschlüsselungsmetadaten zusammen mit dem Objekt übertragen werden.

`INSTRUCTION_FILE`

Verschlüsselungsmetadaten werden in einem separaten S3-Objekt (der Befehlsdatei) mit dem Suffix `.instruction` gespeichert.

Verwendungszweck: Verwenden Sie diese Methode, wenn die Größe der Objektmetadaten ein Problem darstellt oder wenn Sie Verschlüsselungsmetadaten vom verschlüsselten Objekt trennen müssen. Beachten Sie, dass für die Verwendung von Befehlsdateien zwei S3-Objekte (das verschlüsselte Objekt und die zugehörige Befehlsdatei) statt eines verwaltet werden müssen.

Beispiel: Konfiguration der Speichermethode

```
// 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**
>
> Denken Sie bei der Verwendung der `INSTRUCTION_FILE` Speichermethode daran, dass beim Löschen des verschlüsselten Objekts die Befehlsdatei nicht automatisch gelöscht wird. Sie müssen beide Objekte getrennt verwalten.

Dokumentenverlauf für das AWS SDK für C++ Developer Guide

In diesem Thema werden wichtige Änderungen am AWS SDK für C++ Entwicklerhandbuch aufgeführt. Für Benachrichtigungen über Aktualisierungen dieser Dokumentation können Sie einen RSS-Feed abonnieren.

Änderung	Beschreibung	Datum
Optionen für die Client-Konfiguration	Die Liste der Client-Konfigurationsvariablen wurde aktualisiert.	8. September 2025
Aktualisierungen der Speicherverwaltung und des Inhaltsverzeichnisses	Die Speicherverwaltungsparameter wurden auf den neuesten Stand gebracht. Standardisiertes Inhaltsverzeichnis, um einheitlicher zu sein AWS SDKs.	14. März 2025
Migration zum Amazon S3 Encryption Client V3	Es wurden Informationen zur Migration des Amazon S3 S3-Verschlüsselungsclients von V2 zu V3 hinzugefügt.	2. Dezember 2024
Benutzerdefiniertes libcrypto	Inhalt für die benutzerdefinierte Verwendung von libcrypto hinzugefügt. Die CMake maximale Beschränkung wurde entfernt. Die verfügbaren CMake Parameter wurden aktualisiert.	20. Februar 2024
Inhaltsverzeichnis	Das Inhaltsverzeichnis wurde aktualisiert, um den Zugriff auf Codebeispiele zu erleichtern.	01. Juni 2023

Aktualisierungen der bewährten Methoden für IAM	Aktualisierung des Leitfadens zur Ausrichtung an bewährten IAM-Methoden. Weitere Informationen finden Sie unter Bewährte Sicherheitsmethoden in IAM.	1. März 2023
Nuget wird entfernt	Die Erwähnung von Nuget als praktikable Paketmanager-Option wurde entfernt, da die neueste verfügbare Version zu alt ist.	2. Dezember 2022
Aktualisierungen für Getting Started	Der Inhalt von vcpkg wurde aktualisiert, um deutlich zu machen, dass dieser nicht von einer externen Option unterstützt wird AWS und es sich um eine externe Option handelt. Die Anweisungen zum Erstellen des SDK für Windows mit curl wurden aktualisiert.	18. Oktober 2022
Aktualisiert auf `ClientConfiguration`	Die Struktur der wurde aktualisiert, `ClientConfiguration` um die neueste API genau widerzuspiegeln.	22. September 2022
Verbesserungen bei „Erste Schritte“	Die Anweisungen für die ersten Schritte zum Erstellen des SDK unter Windows und Linux wurden übersichtlicher gestaltet.	24. Juni 2022

Windows Curl-Unterstützung	Es wurden Hinweise zum Erstellen des SDK für Windows mit Curl hinzugefügt.	15. Juni 2022
EC2-Classic wird eingestellt	Hinweise zur Außerbetriebnahme von EC2-Classic hinzugefügt.	13. April 2022
SDK-Metriken aktivieren	Informationen zur Aktivierung von SDK-Metriken wurden entfernt, da diese inzwischen veraltet sind.	20. Januar 2022
Mit Diensten arbeiten AWS	Es sind Listen der Codebeispiele enthalten, die GitHub im Codebeispiel-Repository verfügbar sind.	11. Januar 2022
Verbesserungen	Verbesserungen des Abschnitts Erste Schritte, des Abschnitts mit den Codebeispielen und der allgemeinen Standardisierung.	9. Juni 2021
Aktualisierung der Version	Entsprach dem Update auf die allgemeine Release-Version des SDK auf 1.9.	20. April 2021
Erste Schritte mit dem AWS SDK für C++	Der Abschnitt wurde mit neuer Organisation und neuen Details aktualisiert.	17. März 2021
Migration des Amazon S3 S3-Verschlüsselungsclients	Es wurden Informationen zur Migration Ihrer Anwendungen von V1 auf V2 des Amazon S3 S3-Verschlüsselungsclients hinzugefügt.	7. August 2020

Inhalt zur Sicherheit	Sicherheitsinhalte hinzugefügt.	6. Februar 2020
Buckets erstellen, auflisten und löschen	Das Amazon S3 `CreateBucket` S3-Beispiel wurde zur Unterstützung aktualisiert AWS-Regionen.	20. Juni 2019
Anweisungen erstellen	Die SDK-Build-Anweisungen wurden aktualisiert.	16. April 2019
Asynchrone Methoden	Neuer Abschnitt hinzugefügt.	16. April 2019
Service-Client-Klassen	Verschiedene Aktualisierungen	5. April 2019
Amazon S3 S3-Zugriffsberechtigungen verwalten	Verschiedene Aktualisierungen	3. April 2019
Updates für die Erstellung und für Konfigurationsvariablen	Die Anweisungen zum Erstellen des SDK wurden aktualisiert. Die verfügbaren AWS Client-Konfigurationsvariablen wurden aktualisiert.	1. März 2019
vcpkg C++-Paketmanager	Die Anweisungen zum Einrichten des vcpkg C++-Paketmanagers wurden aktualisiert.	19. Januar 2019

Die vorliegende Übersetzung wurde maschinell erstellt. Im Falle eines Konflikts oder eines Widerspruchs zwischen dieser übersetzten Fassung und der englischen Fassung (einschließlich infolge von Verzögerungen bei der Übersetzung) ist die englische Fassung maßgeblich.