

Guía para desarrolladores

AWS SDK para C++



AWS SDK para C++: Guía para desarrolladores

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

Las marcas comerciales y la imagen comercial de Amazon no se pueden utilizar en relación con ningún producto o servicio que no sea de Amazon de ninguna manera que pueda causar confusión entre los clientes y que menosprecie o desacredite a Amazon. Todas las demás marcas registradas que no son propiedad de Amazon son propiedad de sus respectivos propietarios, que pueden o no estar afiliados, conectados o patrocinados por Amazon.

Table of Contents

¿Qué es la Guía para desarrolladores de AWS SDK para C++?	1
Documentación y recursos adicionales	1
Mantenimiento y compatibilidad de las versiones principales del SDK	2
Introducción	3
Autenticarse con AWS	3
Uso de las credenciales de la consola	4
Uso del Centro de identidades de IAM	4
Información adicional de autenticación	6
Obtención del SDK desde el código fuente	7
Compilación en Windows	7
Compilación en Linux/macOS	12
Creación de una aplicación simple	16
Obtención del SDK de un administrador de paquetes	22
Requisitos previos	8
Obtención del SDK mediante vcpkg	23
Solución de problemas de compilación	24
Error de CMake: Could not find a package configuration file provided by "AWSSDK"	25
Error de CMake: Could not find load file (and you're on SDK version 1.8)	25
Error de CMake: Could not find load file	26
Runtime Error: cannot proceed because aws-*.dll was not found	27
Configuración de clientes de servicio	28
Configuración del SDK	29
Configuración externa de clientes	30
Configuración de clientes en código	32
Declaraciones	33
Descripciones	35
Región de AWS	41
Proveedores de credencial	42
La cadena de proveedores de credenciales	42
Proveedor de credenciales explícito	46
Almacenamiento en caché de la identidad	46
Parámetros de CMake	46
Variables y opciones generales de CMake	47
Variables y opciones de CMake para Android	62

Registro	65
HTTP	69
Control de los iostreams utilizados por el HttpClient y el AWSCliient	69
Uso de una libcrypto personalizada	71
Cómo compilar una libcrypto personalizada en el SDK para C++	71
Veámoslo todo en una imagen de Docker	73
Uso del SDK	75
Inicialización y cierre del SDK	75
Realización de solicitudes de Servicio de AWS	76
Programación asíncrona	77
Métodos asíncronos para el SDK	77
Llamada a métodos asíncronos del SDK	77
Notificación de la finalización de una operación asíncrona	79
Módulos de utilidades	82
Pila HTTP	82
Utilidades de cadenas	82
Utilidades de hash	82
Analizador JSON	83
Analizador XML	83
Administración de la memoria	83
Asignación y desasignación de memoria	84
STL y cadenas y vectores de AWS	85
Problemas pendientes	86
Desarrolladores de SDK nativos y controles de memoria	87
Gestión de errores	87
Llamada a Servicios de AWS	89
Introducción a los ejemplos de código	89
Estructura de los ejemplos de código	89
Ejemplos de código de compilación y depuración en Visual Studio	91
Introducción a la resolución de errores de tiempo de ejecución	93
Ejemplos guiados	96
Ejemplos de Amazon CloudWatch	97
Ejemplos de Amazon DynamoDB	114
Ejemplos de Amazon EC2	130
Ejemplos de Amazon S3	155
Ejemplos de Amazon SQS	190

Ejemplos de código	206
ACM	207
Acciones	207
API Gateway	237
Escenarios	238
Aurora	238
Introducción	239
Conceptos básicos	242
Acciones	207
Escenarios	238
escalado automático	283
Introducción	239
Conceptos básicos	242
Acciones	207
CloudTrail	314
Acciones	207
CloudWatch	319
Acciones	207
CloudWatch Registros	330
Acciones	207
CodeBuild	334
Acciones	207
Amazon Cognito Identity Provider	339
Introducción	239
Acciones	207
Escenarios	238
DynamoDB	363
Introducción	239
Conceptos básicos	242
Acciones	207
Escenarios	238
Amazon EC2	442
Introducción	239
Acciones	207
EventBridge	477
Acciones	207

AWS Glue	481
Introducción	239
Conceptos básicos	242
Acciones	207
HealthImaging	520
Introducción	239
Acciones	207
Escenarios	238
IAM	555
Introducción	239
Conceptos básicos	242
Acciones	207
AWS IoT	597
Introducción	239
Conceptos básicos	242
Acciones	207
AWS IoT data	640
Acciones	207
Lambda	642
Introducción	239
Conceptos básicos	242
Acciones	207
Escenarios	238
MediaConvert	667
Acciones	207
Amazon RDS	676
Introducción	239
Conceptos básicos	242
Acciones	207
Escenarios	238
Servicio de datos de Amazon RDS	713
Escenarios	238
Amazon Rekognition	714
Introducción	239
Acciones	207
Escenarios	238

Amazon S3	720
Introducción	239
Conceptos básicos	242
Acciones	207
Escenarios	238
Secrets Manager	804
Acciones	207
Amazon SES	805
Acciones	207
Escenarios	238
Amazon SNS	828
Introducción	239
Acciones	207
Escenarios	238
Amazon SQS	870
Introducción	239
Acciones	207
Escenarios	238
AWS STS	907
Acciones	207
Amazon Transcribe Streaming	909
Acciones	207
Escenarios	238
Seguridad	918
Protección de los datos	919
Identity and Access Management	920
Público	920
Autenticación con identidades	921
Administración de acceso mediante políticas	922
Cómo funcionan los Servicios de AWS con IAM	924
Solución de problemas de identidades y accesos en AWS	924
Validación de la conformidad	926
Resiliencia	927
Seguridad de infraestructuras	927
Aplicación de una versión mínima de TLS	928
Aplicación de una versión específica de TLS con libcurl en todas las plataformas	929

Aplicación de una versión específica de TLS en Windows	930
Migración del cliente de cifrado Amazon S3 (V1 a V2)	933
Información general sobre la migración	933
Actualizar los clientes existentes para leer nuevos formatos	934
Migrar clientes de cifrado y descifrado a la versión V2	935
Ejemplos adicionales	937
Migración del cliente de cifrado Amazon S3 (V2 a V3)	940
Información general sobre la migración	940
Comprensión de los conceptos de la V3	941
Actualizar los clientes existentes para leer nuevos formatos	943
Migre los clientes de cifrado y descifrado a la V3	944
Ejemplos adicionales	947
Historial de revisión	951
.....	cmlv

¿Qué es la Guía para desarrolladores de AWS SDK para C++?

Bienvenido a la Guía para desarrolladores de AWS SDK para C++.

El AWS SDK para C++ proporciona una interfaz C++ moderna (versión C++ 11 o posteriores) para Amazon Web Services (AWS). Proporciona API de alto y bajo nivel para casi todas las funciones de AWS, lo que minimiza las dependencias y permite la portabilidad de la plataforma en Windows, macOS, Linux y dispositivos móviles.

[Introducción a \(\) AWS SDK para C++](#)

Note

Los SDK de `aws-iot-device-sdk-cpp` y el archivo AWS IoT son independientes de este SDK. El SDK de AWS IoT Device para C++ v2 está disponible en [aws-iot-device-sdk-cpp-v2](#) en GitHub.

Para obtener más información sobre AWS IoT, consulte [¿Qué es AWS IoT?](#) en la Guía para desarrolladores de AWS IoT.

Documentación y recursos adicionales

Además de esta guía, estos son algunos otros recursos online útiles para los desarrolladores de AWS SDK para C++:

- [Guía de referencia de las herramientas y los SDK de AWS](#): incluye configuraciones, funciones y otros conceptos básicos comunes a los SDK de AWS.
- GitHub:
 - [Código fuente del SDK](#)
 - [Problemas del SDK](#)
- [AWS SDK para C++ Referencia de la API de](#)
- [Blog de desarrolladores de C++ para AWS](#)
- la [AWS Code Sample Catalog](#)
- [Licencia de SDK](#)

- Vídeo: [Presentación del AWS SDK para C++ en AWS re:invent 2015](#)

Mantenimiento y compatibilidad de las versiones principales del SDK

Para obtener información sobre el mantenimiento y la compatibilidad con las principales versiones del SDK y sus dependencias subyacentes, consulte lo siguiente en la [Guía de Referencia de SDK y herramientas de AWS](#):

- [AWS Política de mantenimiento de SDK y herramientas de AWS](#)
- [AWS Matriz de compatibilidad para versiones de SDK y herramientas de AWS](#)

Introducción a () AWS SDK para C++

AWS SDK para C++ es una biblioteca modular, multiplataforma y de código abierto que puede utilizar para conectarse a Amazon Web Services.

AWS SDK para C++ utiliza [CMake](#) para admitir múltiples plataformas en diferentes dominios, incluidos videojuegos, sistemas, dispositivos móviles y dispositivos integrados. CMake es una herramienta de compilación que puede usar para administrar las dependencias de su aplicación y crear archivos makefiles que se correspondan con la plataforma en la que está realizando la compilación. CMake elimina las partes de la compilación que no se utilizan en su plataforma o aplicación.

Antes de ejecutar código para acceder a los recursos de AWS, debe establecer cómo se autentica el código con AWS.

- [Autenticación AWS mediante AWS SDK para C++](#)

Para usar AWS SDK para C++ en su código, obtenga los ejecutables del SDK compilando directamente el código fuente del SDK o usando un administrador de paquetes.

- [Obtención del AWS SDK para C++ desde el código fuente](#)
- [Obtención del AWS SDK para C++ de un administrador de paquetes](#)

Si tiene problemas de compilación relacionados con CMake, consulte [Solución de problemas de compilación del AWS SDK para C++](#).

Autenticación AWS mediante AWS SDK para C++

Debe establecer cómo se autentica su código AWS al desarrollar con. Servicios de AWS Puedes configurar el acceso programático a AWS los recursos de diferentes maneras en función del entorno y del AWS acceso del que dispongas. Para conocer las opciones de todos los métodos principales de autenticación y obtener instrucciones sobre cómo configurarlos para el SDK, consulte [Autenticación y acceso](#) en la Guía de referencia AWS SDKs y herramientas.

Uso de las credenciales de la consola

Para el desarrollo local, recomendamos que los nuevos usuarios utilicen sus credenciales de inicio de sesión AWS de Management Console existentes para acceder a AWS los servicios mediante programación. Tras un flujo de autenticación basado en un navegador, AWS genera credenciales temporales que funcionan en todas las herramientas de desarrollo local, como la AWS CLI y Herramientas de AWS para PowerShell . AWS SDKs Esta función simplifica el proceso de configuración y administración de las credenciales de AWS CLI, especialmente si prefiere la autenticación interactiva en lugar de administrar las claves de acceso a largo plazo.

Si elige este método, siga las instrucciones para iniciar sesión con las credenciales de la consola mediante la AWS CLI. Consulte [Iniciar sesión para el desarrollo AWS local con las credenciales de la consola](#) para obtener más información.

Una vez configurada con la AWS CLI, [la cadena de proveedores de credenciales predeterminada](#) comenzará a utilizar automáticamente el token de inicio de sesión almacenado en caché por la AWS CLI para realizar solicitudes.

Uso del Centro de identidades de IAM

Este método incluye la instalación del AWS CLI para facilitar la configuración y para iniciar sesión con regularidad en el portal de AWS acceso.

Si elige este método, complete el procedimiento de [autenticación del Centro de Identidad de IAM](#) que se indica en la Guía de referencia de herramientas AWS SDKs y herramientas. Una vez completado, el entorno debe contener los siguientes elementos:

- El AWS CLI, que se utiliza para iniciar una sesión en el portal de AWS acceso antes de ejecutar la aplicación.
- Un [archivo config compartido de AWS](#) que tiene un perfil [default] con un conjunto de valores de configuración a los que se puede hacer referencia desde el SDK. Para encontrar la ubicación de este archivo, consulte [Ubicación de los archivos compartidos](#) en la Guía de referencia de AWS SDKs and Tools.
- El archivo compartido de config establece la configuración de [region](#). Esto establece el valor predeterminado Región de AWS que el SDK usa para AWS las solicitudes. Esta región se usa para las solicitudes de servicio del SDK que no tienen especificadas una región.
- El SDK utiliza la [configuración de proveedor de token de SSO](#) del perfil para adquirir las credenciales antes de enviar las solicitudes a AWS. El `sso_role_name` valor, que es un rol de

IAM conectado a un conjunto de permisos del Centro de Identidad de IAM, debería permitir el acceso al Servicios de AWS utilizado en la aplicación.

El siguiente archivo config de ejemplo muestra la configuración de un perfil predeterminado con el proveedor de token de SSO. La configuración `sso_session` del perfil hace referencia a la [sección llamada `sso-session`](#). La `sso-session` sección contiene la configuración para iniciar una sesión en el portal de AWS acceso.

```
[default]
sso_session = my-sso
sso_account_id = 111122223333
sso_role_name = SampleRole
region = us-east-1
output = json

[sso-session my-sso]
sso_region = us-east-1
sso_start_url = https://provided-domain.awsapps.com/start
sso_registration_scopes = sso:account:access
```

AWS SDK para C++ No es necesario añadir paquetes adicionales (por ejemplo, `SSO` y `SSO0IDC`) a la aplicación para utilizar la autenticación del IAM Identity Center.

Inicie una sesión en el portal de AWS acceso

Antes de ejecutar una aplicación para acceder Servicios de AWS, necesita una sesión activa en el portal de AWS acceso para que el SDK utilice la autenticación del IAM Identity Center a fin de resolver las credenciales. En función de la duración de las sesiones configuradas, el acceso terminará por caducar y SDK detectará un error de autenticación. Para iniciar sesión en el portal de AWS acceso, ejecute el siguiente comando en. AWS CLI

```
aws sso login
```

Como tiene una configuración de perfil predeterminada, no necesita llamar al comando con una opción `--profile`. Si la configuración del proveedor de token de SSO utiliza un perfil con nombre, el comando es `aws sso login --profile named-profile`.

Para comprobar si ya tiene una sesión activa, ejecute el siguiente AWS CLI comando.

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

La respuesta a este comando debe indicar la cuenta y el conjunto de permisos del Centro de identidades de IAM configurados en el archivo compartido `config`.

Note

Si ya tiene una sesión activa en el portal de AWS acceso y la ejecuta `aws sso login`, no tendrá que proporcionar credenciales.

Es posible que el proceso de inicio de sesión le pida que permita el AWS CLI acceso a sus datos. Como AWS CLI se basa en el SDK para Python, los mensajes de permiso pueden contener variaciones del `botocore` nombre.

Información adicional de autenticación

Los usuarios humanos, que también reciben el nombre de identidades humanas, son las personas, los administradores, los desarrolladores, los operadores y los consumidores de las aplicaciones. Deben tener una identidad para acceder a sus AWS entornos y aplicaciones. Los usuarios humanos que son miembros de su organización (es decir, usted, el desarrollador) se conocen como identidades de personal. Utilice credenciales temporales al acceder AWS. Puede usar un proveedor de identidad para que sus usuarios humanos proporcionen acceso federado a AWS las cuentas asumiendo funciones, que proporcionan credenciales temporales. Para una administración centralizada del acceso, le recomendamos que utilice AWS IAM Identity Center (IAM Identity Center) para administrar el acceso a sus cuentas y los permisos dentro de esas cuentas. Para obtener más alternativas, consulte lo siguiente:

- Para obtener más información sobre las prácticas recomendadas, consulte [Prácticas recomendadas de seguridad en IAM](#) en la Guía del usuario de IAM.
- Para crear AWS credenciales de corta duración, consulte [Credenciales de seguridad temporales](#) en la Guía del usuario de IAM.
- Para obtener más información sobre otros proveedores de AWS SDK para C++ credenciales, consulte los proveedores de [credenciales estandarizados en la Guía](#) de referencia de herramientas AWS SDKs y herramientas.

Obtención del AWS SDK para C++ desde el código fuente

Para usar el AWS SDK para C++ desde su código fuente, debe compilar primero el SDK desde el código fuente y, a continuación, instalarlo localmente.

Información general del proceso

Proceso general	Detalles del proceso
<p>Compilación e instalación del código fuente del SDK</p> <ol style="list-style-type: none"> 1. Utilice CMake para generar los archivos de compilación del SDK. 2. Compile el SDK. 3. Instale el SDK. 	<p>Primero compile el SDK desde el código fuente e instálelo.</p> <ul style="list-style-type: none"> • Compilación en Windows • Compilación en Linux/macOS
<p>Compilación de la aplicación con el SDK</p> <ol style="list-style-type: none"> 1. Escriba su propio código para usar el SDK o utilice una aplicación de ejemplo y añada el paquete AWSSDK a su archivo cmake. 2. Utilice CMake para generar archivos de compilación para su aplicación. 3. Compile la aplicación. 4. Ejecute la aplicación. 	<p>Luego, desarrolle su propia aplicación con el SDK.</p> <ul style="list-style-type: none"> • Creación de una aplicación simple

Compilación del AWS SDK para C++ en Windows

Para configurar el AWS SDK para C++, puede compilar el SDK usted mismo directamente desde el código fuente o descargar las bibliotecas utilizando un administrador de paquetes.

El código fuente del SDK se divide en paquetes individuales por servicio. La instalación del SDK completo puede tardar hasta una hora. Si se instala solo el subconjunto específico de servicios que utiliza el programa, se reduce el tiempo de instalación y también el tamaño del disco. Para elegir qué servicios instalar, es necesario que conozca el nombre del paquete de cada servicio que utiliza el

programa. La lista de directorios de paquetes está en [aws/aws-sdk-cpp](https://github.com/aws/aws-sdk-cpp) en GitHub. El nombre del paquete es el sufijo del nombre del directorio del servicio.

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

Requisitos previos

Necesita 4 GB de RAM como mínimo para compilar algunos de los clientes de AWS de mayor tamaño. Es posible que el SDK no se pueda compilar en los tipos de instancia t2.micro, t2.small y otros tipos de instancias pequeñas de Amazon EC2 por falta de memoria.

Para utilizar el AWS SDK para C++, necesita tener una de estas herramientas:

- Microsoft Visual Studio 2015 o versiones posteriores
- GNU Compiler Collection (GCC) 4.9 o versiones posteriores o
- Clang 3.3 o versiones posteriores.

Compilación del SDK para Windows con curl

En Windows, el SDK se crea con [WinHTTP](#) como cliente HTTP predeterminado. Sin embargo, WinHTTP 1.0 no admite la transmisión bidireccional HTTP/2, que es obligatoria con algunos Servicios de AWS, como Amazon Transcribe y Amazon Lex. Por lo tanto, a veces será necesario desarrollar la compatibilidad con curl con el SDK. Para ver todas las opciones de descarga de curl disponibles, consulte [Versiones y descargas de curl](#). A continuación, se muestra un método para crear el SDK compatible con curl:

Para compilar el SDK con compatibilidad con la biblioteca curl incluida

1. Vaya a [curl para Windows](#) y descargue el paquete binario de curl para Microsoft Windows.
2. Desempaquete el paquete en una carpeta de su equipo; por ejemplo, C:\curl.
3. Vaya a los [certificados de CA extraídos de Mozilla](#) y descargue el archivo cacert.pem. Este archivo de Privacy Enhanced Mail (PEM) contiene un paquete de certificados digitales válidos que se utilizan para verificar la autenticidad de los sitios web seguros. Estos certificados los distribuyen empresas de entidades de certificación (CA), como GlobalSign y Verisign.
4. Mueva el archivo cacert.pem a la subcarpeta bin que ha desempaquetado en un paso anterior; por ejemplo, C:\curl\bin. Cambie el nombre del archivo a curl-ca-bundle.crt

Además, Microsoft Build Engine (MSBuild) debe poder localizar el `dll` de `curl` en el procedimiento que sigue a continuación. Por lo tanto, debe agregar la ruta de la carpeta del `bin` de `curl` a la variable de entorno `PATH` de Windows; por ejemplo, `set PATH=%PATH%;C:\curl\bin`. Debe agregarla cada vez que abra una nueva línea de comandos para compilar el SDK. O también, puede establecer la variable de entorno de forma global en la configuración del sistema de Windows para recordar la configuración.

Al compilar el SDK desde el código fuente en el procedimiento que sigue a continuación, consulte el paso 5 (Generar archivos de compilación) para conocer la sintaxis de comandos necesaria para incorporar `curl` a su SDK.

Al escribir el código, debe establecer la ubicación del archivo de certificado como `caFile` en [Configuración de clientes de servicio en código de AWS SDK para C++](#). Para ver un ejemplo de cómo usar Amazon Transcribe, consulte [transcribe-streaming](#) en el Repositorio de ejemplos de código de AWS en GitHub.

Compilación del SDK desde el código fuente

Utilice las herramientas de línea de comandos para compilar el SDK desde el código fuente. Puede personalizar la compilación del SDK con este método. Para obtener información sobre las opciones disponibles, consulte [Parámetros de CMake](#). Estos son los tres pasos principales: En primer lugar, compile los archivos con CMake. En segundo lugar, utilice MSBuild para crear los binarios del SDK que funcionan con su sistema operativo y para crear la cadena de herramientas. En tercer lugar, instale o copie los binarios en la ubicación correcta del equipo donde se está realizando el desarrollo.

Para compilar el SDK desde el código fuente

1. Instale [CMake](#) (versión 3.13 como mínimo) y las herramientas de compilación que corresponda a su plataforma. Le recomendamos añadir `cmake` al `PATH`. Para comprobar su versión de CMake, abra una línea de comandos y ejecute el comando `cmake --version`
2. Cuando se lo indique el sistema, abra la carpeta en la que desee almacenar el SDK.
3. Obtenga el código fuente más reciente.

La versión 1.11 utiliza submódulos de git para empaquetar las dependencias externas. Esto incluye las [bibliotecas CRT](#) descritas en la Guía de referencia de las herramientas y los SDK de AWS.

Descargue o clone el código fuente del SDK desde [aws/aws-sdk-cpp](#) en GitHub:

- Clonación con Git: HTTPS

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

- Clonación con Git: SSH

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

4. Le recomendamos que guarde los archivos de compilación generados fuera del directorio de origen del SDK. Cree un nuevo directorio para almacenar los archivos de compilación y vaya hasta esa carpeta.

```
mkdir sdk_build  
cd sdk_build
```

5. Ejecute cmake para generar los archivos de compilación. Especifique en la línea de comandos de cmake si desea crear una versión Debug o una Release. Elija Debug a lo largo de todo este procedimiento para ejecutar una configuración de depuración del código de la aplicación. Elija Release a lo largo de todo este procedimiento para ejecutar una configuración de lanzamiento del código de la aplicación. En Windows, la ubicación de instalación del SDK suele ser `\Program Files (x86)\aws-cpp-sdk-all\`. Sintaxis de comandos:

```
{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}
```

Para ver más formas de modificar el resultado de la compilación, consulte [Parámetros de CMake](#).

Para generar los archivos de compilación, lleve a cabo una de las siguientes acciones:

- Genere archivos de compilación (todos los Servicios de AWS): para compilar todo el SDK, ejecute cmake y especifique si desea compilar una versión Debug o Release. Por ejemplo:

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

- Genere archivos de compilación (subconjunto de los Servicios de AWS): para crear solo un servicio o paquetes de servicios específicos para el SDK, añada el parámetro [BUILD_ONLY](#)

de CMake, con los nombres de los servicios separados por punto y coma. El siguiente ejemplo compila únicamente el paquete de servicios Amazon S3:

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

- Genere archivos de compilación (con curl): después de completar los requisitos previos de curl, necesita tres opciones adicionales de línea de comandos de cmake para incluir la compatibilidad con curl en el SDK: [FORCE_CURL](#), [CURL_INCLUDE_DIR](#) y [CURL_LIBRARY](#). Por ejemplo:

```
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

Si recibe el mensaje de error Failed to build third-party libraries, ejecute **cmake --version** para comprobar su versión de CMake. Es necesario usar la versión 3.13 como mínimo de CMake.

6. Compile los binarios del SDK. Si va a compilar todo el SDK, este paso puede tardar una hora o más. Sintaxis de comandos:

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

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

Note

Si recibe el mensaje de error The code execution cannot proceed... dll not found. Reinstalling the program may fix this problem., pruebe a ejecutar el comando cmake de nuevo.

7. Abra una línea de comandos con privilegios de administrador para instalar el SDK en la ubicación especificada anteriormente mediante el parámetro CMAKE_PREFIX_PATH. Sintaxis de comandos:

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

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

Compilación para Android en Windows

Para compilar para Android, añade `-DTARGET_ARCH=ANDROID` a su línea de comandos de `cmake`. El AWS SDK para C++ incluye un archivo de cadena de herramientas de CMake que hace referencia a las variables de entorno adecuadas (`ANDROID_NDK`) para proporcionarle todo lo que necesita.

Para compilar el SDK para Android en Windows, debe ejecutar `cmake` desde la línea de comandos para desarrolladores de Visual Studio (2015 o versiones posteriores). También necesitará tener NMAKE [NMAKE](#) instalado y los comandos `git` y `patch` en su ruta. Si ha instalado `git` en un sistema Windows, lo más probable es que encuentre `patch` en un directorio secundario (`.../Git/usr/bin/`). Una vez que haya verificado estos requisitos, su línea de comandos de `cmake` cambiará ligeramente para usar NMAKE.

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

NMAKE compila en serie. Para compilar más rápidamente, le recomendamos que instale JOM como alternativa a NMAKE y que cambie, a continuación, la invocación de `cmake` de la siguiente manera:

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

Para ver un ejemplo de aplicación, consulte [Cómo configurar una aplicación de Android con AWS SDK para C++](#).

Compilación del AWS SDK para C++ en Linux/macOS

Para configurar el AWS SDK para C++, puede compilar el SDK usted mismo directamente desde el código fuente o descargar las bibliotecas utilizando un administrador de paquetes.

El código fuente del SDK se divide en paquetes individuales por servicio. La instalación del SDK completo puede tardar hasta una hora. Si se instala solo el subconjunto específico de servicios que utiliza el programa, se reduce el tiempo de instalación y también el tamaño del disco. Para elegir qué servicios instalar, es necesario que conozca el nombre del paquete de cada servicio que utiliza el

programa. La lista de directorios de paquetes está en [aws/aws-sdk-cpp](https://github.com/aws/aws-sdk-cpp) en GitHub. El nombre del paquete es el sufijo del nombre del directorio del servicio.

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

Requisitos previos

Necesita 4 GB de RAM como mínimo para compilar algunos de los clientes de AWS de mayor tamaño. Es posible que el SDK no se pueda compilar en los tipos de instancia t2.micro, t2.small y otros tipos de instancias pequeñas de Amazon EC2 por falta de memoria.

Para utilizar el AWS SDK para C++, necesita tener una de estas herramientas:

- GNU Compiler Collection (GCC) 4.9 o versiones posteriores o
- Clang 3.3 o versiones posteriores.

Requisitos adicionales para los sistemas de Linux

También tiene que tener los archivos de encabezado (paquetes -dev) para `libcurl`, `libopenssl`, `libuuid`, `zlib` y, de manera opcional, `libpulse` para la compatibilidad con Amazon Polly. Puede encontrar los paquetes con el administrador de paquetes del sistema.

Para instalar los paquetes en sistemas basados en Debian/Ubuntu

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

Para instalar los paquetes en sistemas basados en Amazon Linux/RedHat/Fedora/CentOS

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

Compilación del SDK desde el código fuente

Utilice las herramientas de la línea de comandos para compilar el SDK desde el código fuente como alternativa al uso de `vcpkg`. Puede personalizar la compilación del SDK con este método. Para obtener información sobre las opciones disponibles, consulte [Parámetros de CMake](#).

Para compilar el SDK desde el código fuente

1. Instale [CMake](#) (versión 3.13 como mínimo) y las herramientas de compilación que corresponda a su plataforma. Le recomendamos añadir cmake al PATH. Para comprobar su versión de CMake, abra una línea de comandos y ejecute el comando **cmake --version**
2. Cuando se lo indique el sistema, abra la carpeta en la que desee almacenar el SDK.
3. Obtenga el código fuente más reciente.

La versión 1.11 utiliza submódulos de git para empaquetar las dependencias externas. Esto incluye las [bibliotecas CRT](#) descritas en la Guía de referencia de las herramientas y los SDK de AWS.

Descargue o clone el código fuente del SDK desde [aws/aws-sdk-cpp](https://github.com/aws/aws-sdk-cpp) en GitHub:

- Clonación con Git: HTTPS

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

- Clonación con Git: SSH

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


4. Le recomendamos que guarde los archivos de compilación generados fuera del directorio de origen del SDK. Cree un nuevo directorio para almacenar los archivos de compilación y vaya hasta esa carpeta.

```
mkdir sdk_build  
cd sdk_build
```

5. Ejecute cmake para generar los archivos de compilación. Especifique en la línea de comandos de cmake si desea crear una versión Debug o una Release. Elija Debug a lo largo de todo este procedimiento para ejecutar una configuración de depuración del código de la aplicación. Elija Release a lo largo de todo este procedimiento para ejecutar una configuración de lanzamiento del código de la aplicación. Sintaxis de comandos:

```
{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}
```

Para ver más formas de modificar el resultado de la compilación, consulte [Parámetros de CMake](#).

 Note

Al compilar en un Mac con un sistema de archivos que no distinga entre mayúsculas y minúsculas, compruebe el resultado del comando `pwd` en el directorio donde ejecute la compilación. Asegúrese de que el resultado `pwd` utilice mayúsculas y minúsculas para los nombres de directorio, como `/Users` y `Documents`.


Para generar los archivos de compilación, lleve a cabo una de las siguientes acciones:

- Genere archivos de compilación (todos los Servicios de AWS): para compilar todo el SDK, ejecute `cmake` y especifique si desea compilar una versión `Debug` o `Release`. Por ejemplo:

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

- Genere archivos de compilación (subconjunto de los Servicios de AWS): para crear solo un servicio o paquetes de servicios específicos para el SDK, añada el parámetro [BUILD_ONLY](#) de CMake, con los nombres de los servicios separados por punto y coma. El siguiente ejemplo compila únicamente el paquete de servicios Amazon S3:

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

 Note

Si recibe el mensaje de error `Failed to build third-party libraries`, ejecute `cmake --version` para comprobar su versión de CMake. Es necesario usar la versión 3.13 como mínimo de CMake.

6. Compile los binarios del SDK. Si va a compilar todo el SDK, la operación puede tardar una hora o más.

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

7. Instale el SDK. Es posible que tenga que aumentar los privilegios en función de la ubicación en la que decida realizar la instalación.

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

Compilación para Android en Linux

Para compilar para Android, añada `-DTARGET_ARCH=ANDROID` a su línea de comandos de `cmake`. El AWS SDK para C++ incluye un archivo de cadena de herramientas de CMake que hace referencia a las variables de entorno adecuadas (`ANDROID_NDK`) para proporcionarle todo lo que necesita. Para ver un ejemplo de aplicación, consulte [Cómo configurar una aplicación de Android con AWS SDK para C++](#).

Creación de una aplicación simple mediante el AWS SDK para C++

[CMake](#) es una herramienta de compilación que puede usar para administrar las dependencias de su aplicación y crear archivos `makefiles` que se correspondan con la plataforma en la que está realizando la compilación. Puede usar CMake para crear y compilar proyectos con el AWS SDK para C++.

En este ejemplo, se muestran los buckets de Amazon S3 cuyo propietario es usted. Para este ejemplo, no es necesario que tenga un bucket de Amazon S3 en su cuenta de AWS, pero será mucho más interesante si tiene al menos uno. Consulte [Crear un bucket](#) en la Guía del usuario de Amazon Simple Storage Service si no tiene uno todavía.

Paso 1: escritura del código

Este ejemplo consta de una carpeta que contiene un archivo de origen (`hello_s3.cpp`) y un archivo `CMakeLists.txt`. El programa utiliza Amazon S3 para generar informes con la información del bucket de almacenamiento. Este código también está disponible en el [Repositorio de ejemplos de código de AWS](#) en GitHub.

En un archivo de configuración de compilación `CMakeLists.txt`, puede configurar muchas opciones. Para obtener más información, consulte el [tutorial de CMake](#) en el sitio web de CMake.

Note**Deep Dive: configuración de CMAKE_PREFIX_PATH**

De forma predeterminada, en macOS, Linux, Android y otras plataformas distintas de Windows, el AWS SDK para C++ se instala en `/usr/local`, mientras que en Windows, lo hace en `\Program Files (x86)\aws-cpp-sdk-all`.

Al instalar el SDK de AWS en estas ubicaciones estándar, CMake encuentra automáticamente los recursos necesarios. Sin embargo, si instala el SDK de AWS en una ubicación personalizada, tendrá que indicarle a CMake dónde encontrar los siguientes recursos que se crean al compilar el SDK:

- `AWSSDKConfig.cmake`: un archivo de configuración que indica a CMake cómo buscar y usar las bibliotecas del SDK de AWS en su proyecto. Sin este archivo, CMake no puede localizar los archivos de encabezado del SDK de AWS, vincular a las bibliotecas del SDK de AWS ni configurar los indicadores de compilación adecuados.
- (Para la versión 1.8 y anteriores) La ubicación de las dependencias: `aws-c-event-stream`, `aws-c-common` y `aws-checksums`

Para configurar una ruta de instalación personalizada:

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

Si no configura `CMAKE_PREFIX_PATH` para realizar una instalación personalizada, la compilación fallará con errores como "Could not find AWSSDK" cuando CMake intente procesar `find_package(AWSSDK)` en `CMakeLists.txt`.


Note**Deep Dive: bibliotecas de tiempo de ejecución de Windows**

Para ejecutar el programa, se necesitan varias DLL en la ubicación del ejecutable del programa (`aws-c-common.dll`, `aws-c-event-stream.dll`, `aws-checksums.dll` y `yaws-cpp-sdk-core.dll`), así como cualquier DLL específica basada en los componentes de su programa (en este ejemplo, también es obligatorio `aws-cpp-sdk-s3` porque utiliza Amazon S3). La segunda instrucción `if` del archivo `CMakeLists.txt` copia estas bibliotecas desde la ubicación de instalación a la ubicación del ejecutable para cumplir con

este requisito. `AWSSDK_COPY_DYN_LIBS` es una macro definida por el AWS SDK para C++ que copia los archivos DLL del SDK desde la ubicación de instalación a la ubicación del ejecutable de su programa. Si estos archivos DLL no están en la ubicación del ejecutable, se producen excepciones de tiempo de ejecución del tipo "archivo no encontrado". Si encuentra estos errores, revise esta parte del archivo `CMakeLists.txt` para comprobar necesita hacer cambios en su entorno único.

Para crear la carpeta y los archivos de origen

1. Cree un directorio `hello_s3` o un proyecto para guardar los archivos de origen.

 Note

Para completar este ejemplo en Visual Studio: seleccione Create New Project y, a continuación, elija CMake Project. Asigne un nombre al proyecto `hello_s3`. Este nombre de proyecto se usa en el archivo `CMakeLists.txt`.

2. En esa carpeta, añada un archivo `hello_s3.cpp` que incluya el siguiente código, en el que se indican los buckets de Amazon S3 de su propiedad.

```
#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. Agregue un archivo `CMakeLists.txt` que especifique el nombre del proyecto, los ejecutables, los archivos de origen y las bibliotecas vinculadas.

```

# 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})
```

Paso 2: compilación con CMake

CMake usa la información incluida en `CMakeLists.txt` para crear un programa ejecutable.

Le recomendamos que compile la aplicación siguiendo las prácticas estándar de su IDE.

Para compilar la aplicación en la línea de comandos

1. Cree un directorio donde **cmake** compilará su aplicación.

```
mkdir my_project_build
```

2. Cambie al directorio de compilación y ejecute **cmake** usando la ruta al directorio de origen de su proyecto.

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

3. Cuando **cmake** haga generado el directorio de compilación, podrá usar **make** (**nmake** en Windows) o MSBUILD (`msbuild ALL_BUILD.vcxproj` o `cmake --build . --config=Debug`) para compilar su aplicación.

Paso 3: ejecución

Al ejecutar esta aplicación, se mostrará la salida de la consola con el número total de buckets de Amazon S3 y el nombre de cada bucket.

Le recomendamos que ejecute la aplicación siguiendo las prácticas estándar de su IDE.

Note

¡Recuerde iniciar sesión! Si utiliza IAM Identity Center para autenticarse, recuerde iniciar sesión con el comando `aws sso login` de AWS CLI.

Para ejecutar el programa mediante la línea de comandos

1. Cambie al directorio de depuración donde se generó el resultado de la compilación.
2. Ejecute el programa con el nombre del ejecutable.

```
hello_s3
```

Para ver otros ejemplos con el uso de AWS SDK para C++, consulte [Ejemplos guiados de llamadas a los Servicios de AWS mediante el AWS SDK para C++](#).

Obtención del AWS SDK para C++ de un administrador de paquetes

Important

Si utiliza un administrador de paquetes como homebrew o vcpkg: Después de actualizar el SDK para C++ a una nueva versión, debe volver a compilar cualquier biblioteca o ejecutable que dependa del SDK.

Para configurar el AWS SDK para C++, puede compilar el SDK usted mismo directamente desde el código fuente o descargar las bibliotecas utilizando un administrador de paquetes.

El código fuente del SDK se divide en paquetes individuales por servicio. La instalación del SDK completo puede tardar hasta una hora. Si se instala solo el subconjunto específico de servicios que utiliza el programa, se reduce el tiempo de instalación y también el tamaño del disco. Para elegir qué servicios instalar, es necesario que conozca el nombre del paquete de cada servicio que utiliza el programa. La lista de directorios de paquetes está en [aws/aws-sdk-cpp](#) en GitHub. El nombre del paquete es el sufijo del nombre del directorio del servicio.

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

Requisitos previos

Necesita 4 GB de RAM como mínimo para compilar algunos de los clientes de AWS de mayor tamaño. Es posible que el SDK no se pueda compilar en los tipos de instancia t2.micro, t2.small y otros tipos de instancias pequeñas de Amazon EC2 por falta de memoria.

Linux/macOS

Para utilizar el AWS SDK para C++ en Linux/macOS, necesita tener una de estas herramientas:

- GNU Compiler Collection (GCC) 4.9 o versiones posteriores o
- Clang 3.3 o versiones posteriores.

Windows

Para usar el AWS SDK para C++ en Windows, necesita tener una de estas herramientas:

- Microsoft Visual Studio 2015 o versiones posteriores
- GNU Compiler Collection (GCC) 4.9 o versiones posteriores o
- Clang 3.3 o versiones posteriores.

Obtención del SDK mediante vcpkg

Important

La distribución de vcpkg disponible la proporcionan colaboradores externos, pero no se proporciona a través de AWS. La versión más reciente siempre está disponible en la [instalación desde el código fuente](#).

[vcpkg](#) es un administrador de paquetes actualizado y de cuyo mantenimiento se encargan colaboradores externos. Tenga en cuenta que este administrador de paquetes no se proporciona a través de AWS y es posible que no refleje la última versión disponible del AWS SDK para C++. Hay un retraso entre el momento en que se publica una versión por AWS y el momento en que está disponible a través de un administrador de paquetes. La versión más reciente siempre está disponible en la [instalación desde el código fuente](#).

Debe instalar [vcpkg](#) en su sistema.

- Descargue y arranque [vcpkg](#) siguiendo las instrucciones del archivo readme para vcpkg en GitHub y sustituya las siguientes opciones cuando se le solicite:

- Como parte de esas instrucciones, se le indicará que introduzca lo siguiente:

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

Para instalar el SDK entero, introduzca `.\vcpkg\vcpkg install "aws-sdk-cpp[*]" --recurse` o indique solo los servicios específicos del SDK que desea instalar añadiendo un nombre de paquete entre paréntesis; por ejemplo, `.\vcpkg\vcpkg install "aws-sdk-cpp[s3, ec2]" --recurse`.

El resultado muestra un mensaje que incluye lo siguiente:

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

- Copia el comando `-DCMAKE_TOOLCHAIN_FILE` entero para usarlo en CMake más adelante. El archivo `readme` para `vcpkg` en GitHub también indica dónde debe usarlo con su conjunto de herramientas.
- Es posible que también tenga que anotar el tipo de configuración de compilación que ha instalado mediante `vcpkg`. El resultado de la consola muestra la configuración de compilación y la versión del SDK. El siguiente resultado de ejemplo indica que la configuración de compilación es "x86-windows" y que la versión del AWS SDK para C++ instalada es 1.8.

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

Después de instalar el AWS SDK para C++, puede desarrollar su propia aplicación con el SDK. En el ejemplo que se muestra en [Creación de una aplicación simple](#), aparecen los buckets de Amazon S3 cuyo propietario es usted.

Solución de problemas de compilación del AWS SDK para C++

Al compilar AWS SDK para C++ desde el código fuente, pueden surgir algunos de estos problemas de compilación habituales.

Temas

- [Error de CMake: Could not find a package configuration file provided by "AWSSDK"](#)
- [Error de CMake: Could not find load file \(and you're on SDK version 1.8\)](#)
- [Error de CMake: Could not find load file](#)

- [Runtime Error: cannot proceed because aws-*.dll was not found](#)

Error de CMake: Could not find a package configuration file provided by "AWSSDK"

CMake genera el siguiente error si no puede encontrar el SDK instalado.

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

Para resolver este error, debe indicarle a CMake dónde encontrar el SDK instalado (por ejemplo, la carpeta que se ha generado como resultado de la instalación del SDK ([Windows](#), [Linux/macOS](#))). Antes de la primera llamada a `find_package()`, inserte el siguiente comando en su archivo `CMakeLists.txt`. Consulte [???](#) para ver un ejemplo.

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

Error de CMake: Could not find load file (and you're on SDK version 1.8)

CMake genera el siguiente error si no puede encontrar las bibliotecas instaladas.

```
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
```

Para resolver este error, debe indicarle a CMake dónde encontrar el SDK instalado (por ejemplo, la carpeta que se ha generado como resultado de la instalación del SDK ([Windows](#), [Linux/macOS](#))). Antes de la primera llamada a `find_package()`, inserte los siguientes comandos en su archivo `CMakeLists.txt`. Consulte [???](#) para ver un ejemplo.

```
#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()
```

Esta solución solo es válida para la versión 1.8 del SDK, ya que estas dependencias se gestionan de forma diferente en las versiones posteriores. La versión 1.9 resuelve estos problemas al introducir una capa intermedia entre las bibliotecas `aws-sdk-cpp` y `aws-c-*`. Esta nueva capa se denomina `aws-crt-cpp` y es un submódulo de git del SDK para C++. `aws-crt-cpp` también tiene las bibliotecas `aws-c-*` (como `aws-c-common`, `aws-checksums`, `aws-c-event-stream`, etc.) como sus propios submódulos de git. Esto permite que el SDK para C++ obtenga todas las bibliotecas CRT de forma recursiva y mejora el proceso de compilación.

Error de CMake: Could not find load file

CMake genera el siguiente error si no puede encontrar las bibliotecas instaladas.

```
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
```

Para corregir este error, indíquele a CMake que cree bibliotecas compartidas. Antes de la primera llamada a `find_package()`, inserte el siguiente comando en su archivo `CMakeLists.txt`. Consulte [???](#) para ver un ejemplo.

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

Runtime Error: cannot proceed because **aws-*.dll** was not found

CMake genera un error similar al siguiente si no puede encontrar un archivo DLL obligatorio.

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

Este error se produce porque las bibliotecas o los ejecutables necesarios para el SDK para C++ no están disponibles en la misma carpeta que los ejecutables de la aplicación. Para resolver este error, copie el resultado de la compilación del SDK en la ubicación del ejecutable. El nombre de archivo DLL específico del error variará en función de los servicios de AWS que utilice. Realice una de las siguientes acciones siguientes:

- Copie el contenido de la carpeta `/bin` de instalación del AWS SDK para C++ en la carpeta de compilación de la aplicación.
- En el archivo `CMakeLists.txt`, utilice la macro `AWSSDK_CPY_DYN_LIBS` para copiarlos de manera automática.

Añada una llamada a `AWSSDK_CPY_DYN_LIBS(SERVICE_LIST ""`
 `${CMAKE_CURRENT_BINARY_DIR})` o a `CMakeLists.txt` a su archivo
`AWSSDK_CPY_DYN_LIBS(SERVICE_LIST "" ${CMAKE_CURRENT_BINARY_DIR}/`
 `${CMAKE_BUILD_TYPE})` para utilizar esta macro para que realice la copia por usted. Consulte [???](#) para ver un ejemplo.

Elija la ruta de copia correcta para su entorno de compilación. La compilación mediante la línea de comandos suele colocar el resultado de la compilación en una subcarpeta (`/Debug`), pero Visual Studio y otros IDE no suelen hacerlo. Compruebe dónde están los ejecutables de salida y asegúrese de que la macro se esté copiando en esa ubicación. Al realizar este tipo de cambios, se recomienda eliminar el contenido del directorio de salida de la compilación para disponer de un punto de partida limpio para la siguiente compilación.

Configuración de clientes de servicio en el AWS SDK para C++

Para acceder mediante programación a los Servicios de AWS, el AWS SDK para C++ utiliza una clase de cliente para cada Servicio de AWS. Por ejemplo, si su aplicación necesita acceder a Amazon EC2, su aplicación crearía un objeto de cliente de Amazon EC2 para interactuar con ese servicio. A continuación, utiliza el cliente de servicio para realizar solicitudes al mismo Servicio de AWS.

Para realizar una solicitud a un Servicio de AWS, debe crear primero un cliente de servicio. Para cada Servicio de AWS que utilice su código, tiene su propia biblioteca y su propio tipo específico para interactuar con él. El cliente expone un método para cada operación de API expuesta por el servicio.

Hay muchas maneras alternativas de configurar el comportamiento del SDK, pero, en última instancia, todo está relacionado con el comportamiento de los clientes de servicio. Una configuración no tiene efecto hasta que se utiliza un cliente de servicio creado a partir de ella.

Debe establecer cómo se autentica el código con AWS cuando desarrolla con Servicios de AWS. También debe configurar la Región de AWS en la que lo desea usar.

La [Guía de referencia de las herramientas y los SDK de AWS](#) también contiene configuraciones, características y otros conceptos fundamentales comunes a muchos de los SDK de AWS.

Temas

- [Configuración general con el uso de `Aws::SDKOptions` en el AWS SDK para C++](#)
- [Configuración de clientes de servicio de AWS SDK para C++ de forma externa](#)
- [Configuración de clientes de servicio en código de AWS SDK para C++](#)
- [Configuración de la Región de AWS para el AWS SDK para C++](#)
- [Uso AWS del SDK para proveedores de credenciales de C++](#)
- [Parámetros de CMake para compilar el AWS SDK para C++](#)
- [Configuración y uso de registros en el AWS SDK para C++](#)
- [Anulación del cliente HTTP en el AWS SDK para C++](#)
- [Control de los `iostreams` utilizados por el `HttpClient` y el `AWSCClient` en el AWS SDK para C++](#)
- [Uso de una biblioteca `libcrypto` personalizada en el AWS SDK para C++](#)

Configuración general con el uso de `Aws::SDKOptions` en el AWS SDK para C++

La estructura [Aws::SDKOptions](#) contiene las opciones de configuración del SDK.

`Aws::SDKOptions` se centra en la configuración general del SDK, mientras que la estructura [ClientConfiguration](#) se centra en la configuración de la comunicación con Servicios de AWS.

Se transfiere una instancia de [Aws::SDKOptions](#) a los [métodos `Aws::InitAPI` y `Aws::ShutdownAPI`](#). Se debe enviar la misma instancia a ambos métodos.

En los siguientes ejemplos se muestran algunas de las opciones disponibles.

- Activación del registro con el registrador predeterminado

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

- Anulación de la fábrica predeterminada de clientes HTTP

```
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` utiliza un cierre (también denominado función anónima o expresión lambda) en lugar de un `std::shared_ptr`. Cada una de las funciones de la fábrica del SDK funciona de esta manera porque, en el momento en que se realiza la asignación de

memoria de fábrica, el administrador de memoria no se ha instalado todavía. Al transferir un cierre al método, se llamará al administrador de memoria para que realice la asignación de memoria cuando sea seguro hacerlo. Una técnica sencilla para llevar a cabo este procedimiento consiste en utilizar una expresión lambda.

- Uso de un controlador SIGPIPE global

Si compila el SDK para C++ con curl y OpenSSL, debe especificar un controlador de señales. Si no utiliza su propio controlador de señales personalizado, establezca `installSigPipeHandler` en `true`.

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

Cuando `installSigPipeHandler` es `true`, el SDK para C++ usa un controlador que ignora las señales SIGPIPE. Para obtener más información sobre SIGPIPE, consulte [Señales de error de operación](#) en el sitio web del sistema operativo GNU. Para obtener más información sobre el controlador curl, consulte [CURLOPT_NOSIGNAL](#) en el sitio web de curl.

Las bibliotecas subyacentes de curl y OpenSSL pueden enviar una señal SIGPIPE para notificar cuándo cierra el lado remoto una conexión. La aplicación debe gestionar estas señales. Para obtener más información sobre esta funcionalidad de curl, consulte [libcurl thread safety](#) en el sitio web de curl. Este comportamiento no está integrado automáticamente en el SDK porque los controladores de señales son globales para cada aplicación y la biblioteca es una dependencia del SDK.

Configuración de clientes de servicio de AWS SDK para C++ de forma externa

Muchas opciones de configuración se pueden gestionar fuera del código. Cuando la configuración se gestiona de manera externa, se aplica a todas las aplicaciones. La mayoría de las opciones de configuración se pueden establecer como variables de entorno o en un archivo `config` de AWS

independiente compartido. El archivo de `config` compartido puede mantener conjuntos de opciones independientes, denominados perfiles, para proporcionar diferentes configuraciones para distintos entornos o pruebas.

Las variables de entorno y la configuración del archivo `config` compartido están estandarizadas y se comparten entre los SDK y las herramientas de AWS para garantizar una funcionalidad coherente en los diferentes lenguajes de programación y aplicaciones.

Consulte la Guía de referencia de las herramientas y los SDK de AWS para obtener información sobre cómo configurar una aplicación con estos métodos, además de obtener información sobre cada ajuste de `cross-sdk`. Para ver toda la configuración que el SDK puede resolver a partir de las variables de entorno o los archivos de configuración, consulte [Referencia de configuración](#) en la Guía de referencia de los SDK y las herramientas de AWS.

Para realizar una solicitud a un Servicio de AWS, primero debe crear una instancia de un cliente para ese servicio. Puede configurar los ajustes comunes para los clientes del servicio, como los tiempos de espera, el cliente HTTP y la configuración de reintentos.

Cada cliente de servicio requiere una Región de AWS y un proveedor de credenciales. El SDK usa estos valores para enviar solicitudes a la región correcta de sus recursos y para firmar las solicitudes con las credenciales correctas. Puede especificar estos valores mediante programación en el código o hacer que se carguen automáticamente desde el entorno.

El SDK tiene una serie de lugares (o fuentes) que comprueba para encontrar un valor para la configuración global.

1. Cualquier ajuste explícito establecido en el código o en el propio cliente de un servicio tiene prioridad sobre cualquier otra cosa.
2. Variables de entorno
 - Para obtener más información sobre cómo establecer variables de entorno, consulte [variables de entorno](#) en la Guía de referencia de las herramientas y los SDK de AWS.
 - Tenga en cuenta que puede configurar variables de entorno para un intérprete de comandos en diferentes niveles de ámbito: todo el sistema, todos los usuarios y una sesión de terminal específica.
3. Archivos `config` y `credentials` compartidos
 - Para obtener más información sobre cómo configurar estos archivos, consulte [Archivos `config` y `credentials` compartidos](#) y en la Guía de referencia de los SDK y las herramientas de AWS.

4. Los valores predeterminados proporcionados por el código fuente del SDK se utilizan en último lugar.
 - Algunas propiedades, como `Region`, no tienen un valor predeterminado. Debe especificarlas de manera explícita en el código, en la configuración del entorno o en el archivo `config` compartido. Si el SDK no puede resolver la configuración requerida, las solicitudes de la API pueden generar errores en tiempo de ejecución.

Note

Para ver toda la configuración que el SDK puede resolver a partir de las variables de entorno o los archivos de configuración, consulte [Referencia de configuración](#) en la Guía de referencia de los SDK y las herramientas de AWS.

Configuración de clientes de servicio en código de AWS SDK para C++

Cuando la configuración se gestiona directamente en el código, el alcance de la configuración se limita a la aplicación que utiliza ese código. Dentro de esa aplicación, hay opciones para la configuración global de todos los clientes de servicio, la configuración para todos los clientes de un determinado tipo de Servicio de AWS o la configuración para una instancia de cliente de servicio específica.

El AWS SDK para C++ incluye clases de clientes de Servicio de AWS que proporcionan funciones para interactuar con los Servicios de AWS que se utilizan en la aplicación. En el SDK para C++, puede cambiar la configuración de cliente predeterminada. Esto le resultará útil si desea:

- Conectarse a Internet a través del proxy
- Cambiar la configuración del transporte HTTP, como el tiempo de espera y los reintentos de solicitud de conexión
- Especificar sugerencias del tamaño del búfer del socket TCP

`ClientConfiguration` es una estructura del SDK para C++ para la que puede crear una instancia y utilizar en el código. El siguiente fragmento de código ilustra el uso de esta clase para acceder a Amazon S3 a través de un proxy.


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

Declaraciones de variables de configuración

La estructura `ClientConfiguration` declara las siguientes variables de miembro:

```
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);
```

Descripciones de las variables de configuración

En la siguiente lista se describen las variables de miembro de `ClientConfiguration` que puede utilizar para personalizar el comportamiento de clientes.

`accountId`

Especifica el ID de la Cuenta de AWS para el enrutamiento de puntos de conexión basado en cuentas. Utilice el formato 11122223333. El enrutamiento de puntos de conexión basado en cuentas mejora el rendimiento de las solicitudes para algunos servicios.

`accountIdEndpointMode`

Controla el comportamiento del enrutamiento de los puntos de conexión basado en cuentas. Los valores válidos son "required", "disabled" o "preferred". El valor predeterminado es "preferred". Configúrelo como "disabled" para desactivar el enrutamiento de puntos de conexión basado en cuentas cuando sea necesario.

`allowSystemProxy`

Controla si el cliente HTTP descubre la configuración del proxy del sistema. El valor predeterminado es false. Establézcalo en true para habilitar la detección automática del proxy.

IdAplicación

Indica un identificador opcional específico de la aplicación. Cuando se establece, este valor se añade al encabezado `User-Agent` con el formato `App/{appId}`. Puede establecer este valor mediante la variable de entorno `AWS_SDK_UA_APP_ID` o el atributo de perfil de configuración compartido `sdk_ua_app_id`.

caPath, caFile

Indica al cliente HTTP dónde encontrar el almacén de confianza de certificados SSL. Un ejemplo de almacén de confianza podría ser un directorio preparado con la utilidad `c_rehash` de OpenSSL. No es necesario que configure estas variables a menos que su entorno utilice enlaces simbólicos. Estas variables no tienen ningún efecto en los sistemas Windows y macOS.

checksumConfig

Contiene la configuración de cálculo y validación de la suma de verificación. Incluye `requestChecksumCalculation` y `responseChecksumValidation` con el valor predeterminado `WHEN_SUPPORTED`.

configFactories

Especifica los métodos de fábrica para inicializar las clases de utilidades de cliente, como `Executor` y `RetryStrategy`. Utiliza las fábricas predeterminadas a menos que se anulen.

requestTimeoutMs y connectTimeoutMs

Especifica la cantidad de tiempo en milisegundos que se debe esperar para que el tiempo de espera de una solicitud HTTP se agote. Por ejemplo, considere la posibilidad de aumentar estos tiempos cuando transfiera archivos de gran tamaño.

credentialProviderConfig

Contiene las opciones de configuración de los proveedores de credenciales. Utilice esta estructura para personalizar la forma en que el SDK obtiene credenciales de AWS.

disableExpectHeader

Aplicable solo a clientes HTTP CURL. De forma predeterminada, CURL añade el encabezado "Expect: 100-Continue" en una solicitud HTTP para que no se envíe la carga útil HTTP en los casos en los que el servidor responde con un error inmediatamente después de recibir el encabezado. Este comportamiento puede ahorrar un viaje de ida y vuelta y resulta útil en situaciones en las que la carga útil es pequeña y la latencia de la red es relevante. La

configuración predeterminada de la variable es `false`. Si se establece en `true`, `CURL` recibe la instrucción de enviar juntos el encabezado y la carga útil del cuerpo de la solicitud HTTP.

`disableIMDS`

Controla si las llamadas al servicio de metadatos de instancias (IMDS) están deshabilitadas. El valor predeterminado es `false`. Establézcalo en `true` para deshabilitar las llamadas IMDS cuando se ejecutan fuera de las instancias EC2.

`disableImdsV1`

Controla si las llamadas de IMDSv1 están deshabilitadas y, al mismo tiempo, se permiten las llamadas de IMDSv2. El valor predeterminado es `false`. Si se establece en `true`, solo se deshabilitan las llamadas de IMDSv1 para mejorar la seguridad.

`enableClockSkewAdjustment`

Controla si el sesgo del reloj se ajusta después de cada intento HTTP. El valor predeterminado es `false`.

`enableEndpointDiscovery`

Controla si se utiliza la detección de puntos de conexión. De forma predeterminada, se utilizan puntos de conexión regionales o anulados. Para habilitar la detección de puntos de conexión, establezca la variable en `true`.

`enableHostPrefixInjection`

Controla si el host HTTP añade un prefijo "data-" a las solicitudes de `DiscoverInstances`. De manera predeterminada, el comportamiento está habilitado. Para deshabilitarlo, establezca la variable en `false`.

`enableHttpClientTrace`

Controla si el rastreo de clientes HTTP está habilitado con fines de depuración. El valor predeterminado es `false`. Establézcalo en `true` para habilitar el registro detallado de solicitudes y respuestas HTTP.

`enableTcpKeepAlive`

Controla si se envían paquetes TCP keep-alive. El valor predeterminado es `true`. Utilícelo junto con la variable `tcpKeepAliveIntervalMs`. Esta variable no se aplica a `WinInet` ni al cliente `IXMLHttpRequest2`.

`endpointOverride`

Especifica un punto de conexión HTTP de anulación con el que comunicarse con un servicio.

executor

Hace referencia a la implementación del controlador Executor asíncrono. El comportamiento predeterminado es crear y separar un subproceso para cada llamada asíncrona. Para cambiar este comportamiento, implemente una subclase de la clase `Executor` y asigne una instancia a esta variable.

followRedirects

Controla el comportamiento al gestionar los códigos de redireccionamiento HTTP 300.

httpLibOverride

Especifica la implementación HTTP devuelta por la fábrica HTTP predeterminada. El cliente HTTP predeterminado para Windows es `WinHTTP`. El cliente HTTP predeterminado para todas las demás plataformas es `CURL`.

httpLibPerfMode

Especifica el modo de rendimiento de la biblioteca HTTP. El valor predeterminado es `LOW_LATENCY`. Puede ajustar esta configuración para optimizarla en función de distintas características de rendimiento.

httpRequestTimeoutMs

Especifica el tiempo de espera de la solicitud HTTP en milisegundos. El valor predeterminado del tiempo de espera es 0 (sin tiempo de espera). Considere la posibilidad de aumentar este valor al transferir archivos grandes.

lowSpeedLimit

Especifica la velocidad de transferencia mínima permitida en bytes por segundo. Si la velocidad de transferencia es inferior a la velocidad especificada, la operación de transferencia se interrumpe. La configuración predeterminada es 1 byte/segundo. Esta variable solo se aplica a los clientes `CURL`.

maxConnections

Especifica el número máximo de conexiones HTTP con un único servidor. El valor predeterminado es 25. No existe un valor máximo permitido que no sea el que pueda soportar razonablemente el ancho de banda.

nonProxyHosts

Especifica un conjunto de nombres de host que deben omitir la configuración del proxy. Utilice esta configuración para excluir hosts específicos de la configuración del proxy.

profileName

Especifica el nombre de perfil de AWS para usar en la configuración. El SDK carga los ajustes del perfil especificado en el archivo de configuración de AWS.

proxyCaFile

Especifica la ruta al archivo de la entidad de certificación para las conexiones de proxy cuando es diferente de la predeterminada.

proxyCaPath

Especifica la ruta al almacén de confianza de la autoridad de certificación para las conexiones de proxy cuando es diferente de la predeterminada.

proxyScheme, proxyHost, proxyPort, proxyUserName y proxyPassword

Se utilizan para instalar y configurar un proxy para todas las comunicaciones con AWS. Algunos ejemplos de casos en los que esta funcionalidad puede resultar útil son la depuración junto con la suite Burp o el uso de un proxy para conectarse a Internet.

proxySSLCertPath

Especifica la ruta de acceso al archivo de certificado SSL para las conexiones de proxy que requieren certificados de cliente.

proxySSLCertType

Especifica el tipo de certificado SSL para las conexiones de proxy. Los tipos más frecuentes incluyen PEM y DER.

proxySSLKeyPassword

Especifica la contraseña de la clave privada SSL utilizada en las conexiones de proxy cuando la clave está protegida con contraseña.

proxySSLKeyPath

Especifica la ruta de acceso al archivo de clave privada para las conexiones de proxy que requieren certificados de cliente.

proxySSLKeyType

Especifica el tipo de clave privada SSL para las conexiones de proxy. Los tipos más frecuentes incluyen PEM y DER.

`writeRateLimiter` y `readRateLimiter`

Referencias a las implementaciones de limitadores de velocidad de lectura y escritura que se utilizan para reducir el ancho de banda utilizado por la capa de transporte. De forma predeterminada, las velocidades de lectura y escritura no están limitadas. Para introducir la limitación, implemente una subclase de `RateLimiterInterface` y asigne una instancia a estas variables.

`region`

Especifica la región de AWS que se va a utilizar; por ejemplo, `us-east-1`. De forma predeterminada, la región utilizada es la región predeterminada configurada en las credenciales de AWS aplicables.

`requestCompressionConfig`

Contiene las opciones de configuración de la compresión de solicitudes. Utilice esta estructura para controlar cuándo y cómo se comprimen las solicitudes antes de la transmisión.

`retryStrategy`

Hace referencia a la implementación de la estrategia de reintentos. La estrategia predeterminada implementa una política de retroceso exponencial. Para llevar a cabo una estrategia diferente, implemente una subclase de la clase `RetryStrategy` y asigne una instancia a esta variable.

`scheme`

Especifica el esquema de direccionamiento URI: `HTTP` o `HTTPS`. El esquema predeterminado es `HTTPS`.

`tcpKeepAliveIntervalMs`

Especifica el intervalo de tiempo en milisegundos en el que se debe enviar un paquete `keep-alive` a través de una conexión TCP. El intervalo `keep-alive` predeterminado es de 30 segundos. La configuración mínima es de 15 segundos. Esta variable no se aplica a `WinInet` ni al cliente `IXMLHttpRequest2`.

`telemetryProvider`

Hace referencia a la implementación del proveedor de telemetría para recopilar métricas y rastrear datos. Configure este ajuste para habilitar las características de observabilidad.

`userAgent`

Para uso interno únicamente. No cambie la configuración de esta variable.

useDualStack

Controla si se deben utilizar puntos de conexión IPv4 e IPv6 de doble pila. Tenga en cuenta que no todos los servicios de AWS admiten IPv6 en todas las regiones.

useFIPS

Controla si se deben utilizar módulos criptográficos validados por los Estándares de procesamiento de la información federal (FIPS) 140-2. El valor predeterminado es false. Se establece en true cuando se requiere conformidad con FIPS.

verifySSL

Controla si se deben verificar los certificados SSL. De forma predeterminada, los certificados SSL están verificados. Para deshabilitar la verificación, establezca la variable en false.

versión

Especifica la versión HTTP que se va a utilizar con las solicitudes. La configuración predeterminada es HTTP_VERSION_2TLS (HTTP/2 sobre TLS).

winHTTPOptions

Contiene opciones de configuración HTTP específicas de Windows. Incluye useAnonymousAuth con la configuración predeterminada false.

Configuración de la Región de AWS para el AWS SDK para C++

Puede acceder a los Servicios de AWS que operan en un área geográfica específica mediante las Regiones de AWS. Esto puede ser útil para evitar redundancias y para que sus datos y aplicaciones se ejecuten cerca del lugar desde donde usted y los usuarios obtendrán acceso a ellos.

Important

La mayoría de los recursos residen en una Región de AWS específica y debe proporcionar la región correcta para el recurso cuando utilice el SDK.

Para ver ejemplos sobre cómo configurar la región predeterminada mediante el archivo config de AWS compartido o las variables de entorno, consulte [Región de AWS](#) en la Guía de referencia de las herramientas y los SDK de AWS.

Debe establecer una Región de AWS predeterminada para que el AWS SDK para C++ la utilice en las solicitudes de AWS. Este valor predeterminado se usa con las llamadas al método de servicio del SDK que no tengan especificada una región. En el SDK para C++, también puede configurar la región predeterminada mediante la [Configuración de clientes en código](#).

Uso AWS del SDK para proveedores de credenciales de C++

Todas las solicitudes AWS deben estar firmadas criptográficamente con las credenciales emitidas por AWS. En tiempo de ejecución, el SDK recupera los valores de configuración para las credenciales comprobando varias ubicaciones.

La autenticación con se AWS puede gestionar fuera de su base de código. El SDK puede detectar, utilizar y actualizar automáticamente muchos métodos de autenticación mediante la cadena de proveedores de credenciales.

Para ver las opciones guiadas para empezar a AWS autenticar tu proyecto, consulta [Autenticación y acceso](#) en la Guía de referencia de herramientas AWS SDKs y herramientas.

La cadena de proveedores de credenciales

Si no especifica de manera explícita un proveedor de credenciales al crear un cliente, el SDK para C++ usa una cadena de proveedores de credenciales que comprueba una serie de lugares donde usted puede proporcionar credenciales. Cuando el SDK encuentra credenciales en una de esas ubicaciones, la búsqueda se detiene.

Orden de recuperación de credenciales

Todos SDKs tienen una serie de sitios (o fuentes) que consultan para obtener credenciales válidas y utilizarlas para realizar una solicitud a un Servicio de AWS. Una vez que se encuentran las credenciales válidas, se detiene la búsqueda. Esta búsqueda sistemática se denomina cadena de proveedores de credenciales.

Para cada paso de la cadena, hay diferentes maneras de establecer los valores. La configuración de los valores directamente en el código siempre tiene prioridad, seguida de la configuración como variables de entorno y, por último, en el AWS config archivo compartido. Para obtener más información, consulte [Prioridad de los ajustes](#) en la Guía de referencia de herramientas AWS SDKs y herramientas.

El SDK intenta cargar las credenciales del [default] perfil en los `credentials` archivos AWS config AND compartidos. Puede usar la variable de entorno `AWS_PROFILE` para elegir un perfil con

nombre que desee que cargue el SDK en lugar de usar [default]. Los `credentials` archivos `config` y los comparten AWS SDKs las herramientas. La guía de referencia de AWS SDKs and Tools contiene información sobre los ajustes de configuración del SDK que utilizan todos AWS SDKs y los AWS CLI. Para obtener más información sobre cómo configurar el SDK a través del AWS `config` archivo compartido, consulte [Archivos de credenciales y configuración compartidos](#). Para obtener más información sobre cómo configurar el SDK mediante la configuración de variables de entorno, consulte [Compatibilidad con variables de entorno](#).

Para autenticarse AWS, el SDK para C++ comprueba los proveedores de credenciales en el siguiente orden.

1. AWS claves de acceso (credenciales temporales y de larga duración)

El SDK intenta cargar las credenciales desde las variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY` y de `AWS_SESSION_TOKEN` entorno, o desde el AWS `credentials` archivo compartido.

- Para obtener instrucciones sobre la configuración de este proveedor, consulta [las claves de AWS acceso](#) en la Guía de referencia de herramientas AWS SDKs y herramientas.
- Para obtener más información sobre las propiedades de configuración del SDK de este proveedor, consulte [las claves de AWS acceso](#) en la Guía de referencia de herramientas AWS SDKs y herramientas.

2. AWS STS identidad web

Al crear aplicaciones móviles o aplicaciones web basadas en clientes que requieren acceso a AWS, AWS Security Token Service (AWS STS) devuelve un conjunto de credenciales de seguridad temporales para los usuarios federados que se autentican a través de un proveedor de identidad público (IdP).

- Cuando lo especificas en un perfil, el SDK o la herramienta intentarán recuperar las credenciales temporales mediante el método de la API. `AWS STS AssumeRoleWithWebIdentity` Para obtener más información sobre este método, consulta [AssumeRoleWithWebIdentity](#) la referencia de la AWS Security Token Service API.
- Para obtener orientación sobre la configuración de este proveedor, consulte [Federate with web identity u OpenID Connect](#) AWS SDKs en la Guía de referencia de and Tools.
- Para obtener más información sobre las propiedades de configuración del SDK de este proveedor, consulte [Asumir el rol de proveedor de credenciales](#) en la Guía de referencia de herramientas AWS SDKs y herramientas.

3. Centro de identidades de IAM

Si utiliza el Centro de identidad de IAM para autenticarse, el SDK para C++ utiliza el token de inicio de sesión único que se configuró mediante la ejecución del comando CLI. `AWS aws sso login` El SDK usa las credenciales temporales que IAM Identity Center ha intercambiado por un token válido. A continuación, el SDK utiliza las credenciales temporales cuando llama a los Servicios de AWS. Para obtener información detallada sobre este proceso, consulte [Comprender la resolución de credenciales del SDK Servicios de AWS en la Guía de referencia de herramientas y herramientas.AWS SDKs](#)

- Para obtener información sobre la configuración de este proveedor, consulte la [autenticación del Centro de Identidad de IAM](#) en la Guía de referencia de herramientas AWS SDKs y herramientas.
- Para obtener más información sobre las propiedades de configuración del SDK de este proveedor, consulte el proveedor de [credenciales del IAM Identity Center en la Guía](#) de referencia de herramientas AWS SDKs y herramientas.

4. Inicie sesión en el solucionador de identidades de credenciales con Signin AWS

Si utiliza las credenciales de inicio de sesión y de consola de AWS para autenticarse, el SDK para C++ utiliza las credenciales de consola configuradas en ejecución `aws login` o `aws login --profile` en la CLI. El SDK usa estas credenciales cuando llama a AWS los servicios.

- Para obtener información detallada sobre este proceso, consulte [Iniciar sesión para el desarrollo AWS local con las credenciales de la consola](#) en la Guía de referencia de AWS SDKs and Tools.

5. Proveedor de procesos externo

Este proveedor se puede utilizar para proporcionar implementaciones personalizadas, como recuperar credenciales de un almacén de credenciales en las instalaciones o integrarlas con su proveedor de identificación en las instalaciones.

- Para obtener información sobre una forma de configurar este proveedor, consulte [IAM Roles Anywhere](#) en la guía de referencia AWS SDKs y herramientas.
- Para obtener más información sobre las propiedades de configuración del SDK de este proveedor, consulte [Procesar el proveedor de credenciales](#) en la Guía de referencia de herramientas AWS SDKs y herramientas.

6. Credenciales de contenedores de Amazon ECS y Amazon EKS

Sus tareas de Amazon Elastic Container Service y sus cuentas de servicio de Kubernetes pueden tener un rol de IAM asociado a ellas. Los contenedores que se ejecutan en la tarea o los contenedores del pod asumen los permisos concedidos en el rol de IAM. Este rol permite que su código de la aplicación de SDK for C++ (en el contenedor) utilice otros servicios de Servicios de AWS.

El SDK intenta recuperar las credenciales de las variables de entorno

`AWS_CONTAINER_CREDENTIALS_RELATIVE_URI` o

`AWS_CONTAINER_CREDENTIALS_FULL_URI`, que Amazon ECS y Amazon EKS pueden configurar automáticamente.

- Para obtener más información sobre cómo configurar este rol para Amazon ECS, consulte [Rol de IAM de tarea de Amazon ECS](#) en la Guía para desarrolladores de Amazon Elastic Container Service.
- Para obtener información sobre la configuración de Amazon EKS, consulte [Configuración del agente de Pod Identity de Amazon EKS](#) en la Guía del usuario de Amazon EKS.
- Para obtener más información sobre las propiedades de configuración del SDK de este proveedor, consulte el [proveedor de credenciales de contenedor](#) en la Guía de referencia de herramientas AWS SDKs y herramientas.

7. Servicio de metadatos de EC2 instancias de Amazon

Cree un rol de IAM y asíelo a su instancia. La aplicación SDK para C++ de la instancia intenta recuperar las credenciales que proporciona el rol en los metadatos de la instancia.

- Para obtener más información sobre la configuración de esta función y el uso de metadatos, [consulta las funciones de IAM para Amazon EC2](#) y [Work with instance metadata](#) en la Guía del EC2 usuario de Amazon.
- Para obtener más información sobre las propiedades de configuración del SDK de este proveedor, consulta la sección sobre el [proveedor de credenciales del IMDS](#) en la AWS SDKs Guía de referencia de herramientas.

La cadena de proveedores de credenciales se puede consultar

[AWSCredentialsProviderChain](#) en el AWS SDK para C++ código fuente de GitHub

Si ha seguido el enfoque recomendado para los nuevos usuarios para empezar, configure la autenticación con credenciales de inicio [Autenticación AWS mediante AWS SDK para C++](#) de AWS sesión en el tema Primeros pasos. Otros métodos de autenticación son útiles en diferentes

situaciones. Para evitar riesgos de seguridad, recomendamos utilizar siempre credenciales a corto plazo. Para conocer otros procedimientos de métodos de autenticación, consulte [Autenticación y acceso](#) en la Guía de referencia de herramientas AWS SDKs y herramientas.

Proveedor de credenciales explícito

En lugar de confiar en la cadena de proveedores de credenciales para detectar su método de autenticación, puede especificar un proveedor de credenciales específico para que lo utilice el SDK. Para ello, debe proporcionar las credenciales en el constructor del cliente de servicio.

El siguiente ejemplo crea un cliente de Amazon Simple Storage Service proporcionando directamente las credenciales de acceso temporal en lugar de utilizar la cadena.

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

Almacenamiento en caché de la identidad

El SDK almacenará en caché las credenciales y otros tipos de identidad, como los tokens de SSO. De forma predeterminada, el SDK usa una implementación de caché diferida que carga las credenciales cuando se solicitan por primera vez, las almacena en la memoria caché y, a continuación, intenta actualizarlas durante otra solicitud cuando están a punto de caducar. Los clientes creados a partir de la misma [Aws::Client::ClientConfiguration](#) comparten una memoria caché.

Parámetros de CMake para compilar el AWS SDK para C++

Utilice los parámetros de [CMake](#) que se enumeran en esta sección para personalizar la compilación de su SDK.

Puede configurar estas opciones con las herramientas GUI de CMake o con la línea de comandos utilizando `-D`. Por ejemplo:

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

Variables y opciones generales de CMake

A continuación se indican las variables y opciones de **cmake** generales que afectan al proceso de compilación del código fuente del SDK.

Note

Utilice estos parámetros al compilar el código fuente del SDK para el propio SDK para C++.

Temas

- [ADD_CUSTOM_CLIENTS](#)
- [AUTORUN_UNIT_TESTS](#)
- [AWS_AUTORUN_LD_LIBRARY_PATH](#)
- [AWS_SDK_WARNINGS_ARE_ERRORS](#)
- [AWS_USE_CRYPT_SHARED_LIBS](#)
- [AWS_TEST_REGION](#)
- [BUILD_BENCHMARKS](#)
- [BUILD_DEPS](#)
- [BUILD_ONLY](#)
- [BUILD_OPTEL](#)
- [BUILD_SHARED_LIBS](#)
- [BYPASS_DEFAULT_PROXY](#)
- [CPP_STANDARD](#)
- [CURL_INCLUDE_DIR](#)
- [CURL_LIBRARY](#)
- [CUSTOM_MEMORY_MANAGEMENT](#)
- [DISABLE_INTERNAL_IMDSV1_CALLS](#)

- [ENABLE_ADDRESS_SANITIZER](#)
- [ENABLE_CURL_LOGGING](#)
- [ENABLE_HTTP_CLIENT_TESTING](#)
- [ENABLE_RTTI](#)
- [ENABLE_TESTING](#)
- [ENABLE_UNITY_BUILD](#)
- [ENABLE_VIRTUAL_OPERATIONS](#)
- [ENABLE_ZLIB_REQUEST_COMPRESSION](#)
- [FORCE_CURL](#)
- [FORCE_SHARED_CRT](#)
- [G](#)
- [MINIMIZE_SIZE](#)
- [NO_ENCRYPTION](#)
- [NO_HTTP_CLIENT](#)
- [REGENERATE_CLIENTS](#)
- [REGENERATE_DEFAULTS](#)
- [SIMPLE_INSTALL](#)
- [TARGET_ARCH](#)
- [USE_CRT_HTTP_CLIENT](#)
- [USE_IXML_HTTP_REQUEST_2](#)
- [USE_OPENSSL](#)
- [USE_TLS_V1_2](#)
- [USE_TLS_V1_3](#)

ADD_CUSTOM_CLIENTS

Crea un cliente de manera arbitraria en función de la definición de la API. Coloque la definición en la carpeta `code-generation/api-definitions` y, a continuación, transfiera este argumento a **cmake**. El paso de configuración de **cmake** genera el cliente y lo incluye como un subdirectorio en la compilación. Esto resulta especialmente útil para generar un cliente C++ para usar uno de los servicios de [API Gateway](#). Por ejemplo:


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

Note

Para usar el parámetro `ADD_CUSTOM_CLIENTS`, debe tener [Python 2.7](#), Java ([JDK 1.8+](#)) y [Maven](#) instalados y en su `PATH`.

AUTORUN_UNIT_TESTS

Si se establece en `ON`, ejecute las pruebas unitarias automáticamente después de la compilación.

Valores

`ON` | `OFF`

Predeterminado/a

`ON`

AWS_AUTORUN_LD_LIBRARY_PATH

La ruta que se añadirá a `LD_LIBRARY_PATH` para las pruebas unitarias que CMake ejecute automáticamente. Establezca esta ruta si se necesitan bibliotecas de tiempo de ejecución personalizadas para las dependencias anuladas.

Valores

cadena.

Predeterminado/a

`N/A`

AWS_SDK_WARNINGS_ARE_ERRORS

Si se establece en `ON`, trata las advertencias del compilador como errores. Pruebe a cambiar esta opción a `OFF` si observa errores en un compilador nuevo o poco común.

Valores

ON | OFF

Predeterminado/a

ON

AWS_USE_CRYPTO_SHARED_LIBS

Obliga a FindCrypto a usar una biblioteca criptográfica compartida si la encuentra. Cámbielo a OFF para usar la configuración de [BUILD_SHARED_LIBS](#) en su lugar.

Valores

ON | OFF

Predeterminado/a

OFF

AWS_TEST_REGION

La Región de AWS que se usará en las pruebas de integración.

Valores

cadena.

Predeterminado/a

N/A

BUILD_BENCHMARKS

Si se establece en ON, cree el ejecutable de referencia.

Valores

ON | OFF

Predeterminado/a

OFF

BUILD_DEPS

Si se establece en ON, cree dependencias de terceros.

Valores

ON | OFF

Predeterminado/a

ON

BUILD_ONLY

Cree solo los clientes que quiera usar. Si se configura en un SDK de alto nivel, como `aws-cpp-sdk-transfer`, `BUILD_ONLY` resuelve cualquier dependencia de cliente de bajo nivel. También crea pruebas unitarias y de integración relacionadas con los proyectos que seleccione, si existen. Se trata de un argumento de lista, con valores separados por puntos y comas (;). Por ejemplo:

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

Note

El módulo principal del SDK, `aws-sdk-cpp-core`, siempre se crea, independientemente del valor del parámetro `BUILD_ONLY`.

BUILD_OPTEL

Si se establece en ON, crea la implementación de OpenTelemetry del rastreo.

Valores

ON | OFF

Predeterminado/a

OFF

BUILD_SHARED_LIBS

Una opción CMake integrada, que se vuelve a exponer aquí para mayor visibilidad. Si se establece en ON, crea bibliotecas compartidas; de lo contrario, solo crea bibliotecas estáticas.

Note

Para la vinculación dinámica con el SDK, debe definir el símbolo `USE_IMPORT_EXPORT` para todos los objetivos de compilación que utilicen el SDK.

Valores

ON | OFF

Predeterminado/a

ON

BYPASS_DEFAULT_PROXY

Si se establece en ON, omite la configuración de proxy predeterminada de la máquina cuando utilice `IXMLHTTPRequest2`.

Valores

ON | OFF

Predeterminado/a

ON

CPP_STANDARD

Especifica un estándar de C++ personalizado para usarlo con las bases de código C++ 14 y 17.

Valores

11 | 14 | 17

Predeterminado/a

11

CURL_INCLUDE_DIR

La ruta a curl incluye el directorio que contiene los encabezados `libcurl`.

Valores

Ruta de la cadena al directorio *include* seleccionado. Por ejemplo, *D:/path/to/dir/with/curl/include*.

Predeterminado/a

N/A

CURL_LIBRARY

Ruta al archivo de la biblioteca de curl con el que establecer el enlace. Esta biblioteca puede ser una biblioteca estática o una biblioteca de importación, según las necesidades de la aplicación.

Valores

Ruta de la cadena al archivo de la biblioteca de curl. Por ejemplo, *D:/path/to/static/libcurl/file/ie/libcurl.lib.a*.

Predeterminado/a

N/A

CUSTOM_MEMORY_MANAGEMENT

Para usar un administrador de memoria personalizado, establezca el valor en 1. Puede instalar un asignador personalizado para que todos los tipos de STL utilicen la interfaz de asignación personalizada. Si establece el valor 0, es posible que desee seguir utilizando los tipos de plantilla STL para garantizar la seguridad de las DLL en Windows.

Si el enlace estático se establece en ON, la administración de memoria personalizada se establecerá de forma predeterminada en off (0). Si la vinculación dinámica se establece en ON, la administración de memoria personalizada se establece de forma predeterminada en on (1) y evita la asignación y desasignación entre archivos DLL.

Note

Para evitar errores de discordancia en los enlazadores, debe usar el mismo valor (0 o 1) en la totalidad del sistema de compilación.

Para instalar su propio administrador de memoria para que gestione las asignaciones realizadas por el SDK, debe configurar `-DCUSTOM_MEMORY_MANAGEMENT` y definir `USE_AWS_MEMORY_MANAGEMENT` para todos los objetivos de compilación que dependan del SDK.

DISABLE_INTERNAL_IMDSV1_CALLS

Si se establece en ON, no se realiza ninguna llamada interna a la API V1 del [servicio de metadatos de instancias](#). Si se establece en OFF, las llamadas de IMDSv2 utilizarán como alternativa IMDSv1 si se produce un error en la llamada de IMDSv2. Para obtener más información sobre IMDSv1 e IMDSv2, consulte [Uso del servicio de metadatos de instancia para acceder a los metadatos de instancia](#) en la Guía del usuario de Amazon EC2.

Valores

ON | OFF

Predeterminado/a

OFF

ENABLE_ADDRESS_SANITIZER

Si se establece en ON, activa Address Sanitizer para gcc o clang.

Valores

ON | OFF

Predeterminado/a

OFF

ENABLE_CURL_LOGGING

Si se establece en ON, canaliza el registro interno de curl al registrador del SDK.

Valores

ON | OFF

Predeterminado/a

OFF

ENABLE_HTTP_CLIENT_TESTING

Si se establece en ON, cree y ejecute los conjuntos de pruebas de clientes HTTP correspondientes.

Valores

ON | OFF

Predeterminado/a

OFF

ENABLE_RTTI

Controla si el SDK está diseñado para habilitar la información de tipos en tiempo de ejecución (RTTI).

Valores

ON | OFF

Predeterminado/a

ON

ENABLE_TESTING

Controla si los proyectos de pruebas unitarias y de integración se crean durante la creación del SDK.

Valores

ON | OFF

Predeterminado/a

ON

ENABLE_UNITY_BUILD

Si está establecido en ON, la mayoría de las bibliotecas del SDK se crean como un único archivo .cpp generado. Esto puede reducir de manera considerable el tamaño de la biblioteca estática y acelerar el tiempo de compilación.

Valores

ON | OFF

Predeterminado/a

OFF

ENABLE_VIRTUAL_OPERATIONS

Este parámetro suele funcionar junto con REGENERATE_CLIENTS para generar código.

Si ENABLE_VIRTUAL_OPERATIONS se establece en ON y REGENERATE_CLIENTS en ON, las funciones relacionadas con las operaciones en los clientes de servicio se marcarán como `virtual`.

Si ENABLE_VIRTUAL_OPERATIONS se establece en OFF y REGENERATE_CLIENTS en ON, `virtual` no se añadirá a las funciones de operación y las clases de clientes de servicio se marcarán como `final`.

Si ENABLE_VIRTUAL_OPERATIONS se establece en OFF, el SDK también añadirá los indicadores de compilación `-ffunction-sections` y `-fdata-sections` para gcc y clang al compilar.

Para obtener más información, consulte [CMake Parameters](#) en GitHub.

Valores

ON | OFF

Predeterminado/a

ON

ENABLE_ZLIB_REQUEST_COMPRESSION

En los servicios que lo admitan, el contenido de la solicitud se comprimirá. Está habilitado de forma predeterminada si la dependencia está disponible.

Valores

ON | OFF

Predeterminado/a

ON

FORCE_CURL

solo Windows. Si se establece en ON, fuerza el uso del cliente curl en lugar del proveedor de transferencia de datos [WinHTTP](#) predeterminado.

Valores

ON | OFF

Predeterminado/a

OFF

FORCE_SHARED_CRT

Si se establece en ON, el SDK se vincula al motor de ejecución de C de forma dinámica; de lo contrario, utiliza la configuración BUILD_SHARED_LIBS (que a veces es necesaria para mantener la compatibilidad con versiones anteriores del SDK).

Valores

ON | OFF

Predeterminado/a

ON

G

Genera artefactos de compilación, como soluciones de Visual Studio y proyectos de Xcode.

Por ejemplo, en Windows:

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

Para obtener más información, consulte la documentación de CMake que corresponda a su plataforma de aplicaciones.

MINIMIZE_SIZE

Un superconjunto de [ENABLE_UNITY_BUILD](#). Si se establece en ON, esta opción activa `ENABLE_UNITY_BUILD` y otros ajustes de reducción de tamaño binario.

Valores

ON | OFF

Predeterminado/a

OFF

NO_ENCRYPTION

Si se establece en ON, impide que la implementación de criptografía predeterminada específica de la plataforma se incorpore a la biblioteca. Establézcala en ON para incluir propia implementación de criptografía.

Valores

ON | OFF

Predeterminado/a

OFF

NO_HTTP_CLIENT

Si se establece en ON, impide que el cliente HTTP predeterminado específico de la plataforma se integre en la biblioteca. Si se establece en ON, tendrá que proporcionar su propia implementación de cliente HTTP específica de la plataforma.

Valores

ON | OFF

Predeterminado/a

OFF

REGENERATE_CLIENTS

Si se establece en ON, este parámetro elimina todo el código generado y genera los directorios de cliente a partir de la carpeta `code-generation/api-definitions`. Por ejemplo:

```
-DREGENERATE_CLIENTS=1
```

Note

Para usar el parámetro `REGENERATE_CLIENTS`, debe tener [Python 2.7](#), Java ([JDK 1.8+](#)) y [Maven](#) instalados y en su PATH.

REGENERATE_DEFAULTS

Si se establece en ON, este parámetro elimina todo el código predeterminado generado y lo vuelve a generar a partir de la carpeta `code-generation/defaults`. Por ejemplo:

```
-DREGENERATE_DEFAULTS=1
```

Note

Para usar el parámetro `REGENERATE_DEFAULTS`, debe tener [Python 2.7](#), Java ([JDK 1.8+](#)) y [Maven](#) instalados y en su PATH.

SIMPLE_INSTALL

Si se establece en ON, el proceso de instalación no inserta directorios intermedios específicos de la plataforma debajo de `bin/` y `lib/`. Establézcalo en OFF si necesita crear versiones multiplataforma en un único directorio de instalación.

Valores

ON | OFF

Predeterminado/a

ON

TARGET_ARCH

Para realizar una compilación cruzada o una compilación para una plataforma móvil, debe especificar la plataforma de destino. De forma predeterminada, la compilación detecta el sistema operativo del host y compilaciones del sistema operativo detectado.

Note

Cuando TARGET_ARCH es ANDROID, hay opciones adicionales disponibles. Consulte [Variables y opciones de CMake para Android](#).

Valores

WINDOWS | LINUX | APPLE | ANDROID

USE_CRT_HTTP_CLIENT

Si se establece en ON, utilice el cliente HTTP de tiempo de ejecución común, y los sistemas heredados, como WinHTTP y libcurl, no se compilan ni se incluyen.

Valores

ON | OFF

Predeterminado/a

OFF

USE_IXML_HTTP_REQUEST_2

solo Windows. Si se establece en ON, utilice el objeto com de IXMLHttpRequest2 para la pila HTTP.

Valores

ON | OFF

Predeterminado/a

OFF

USE_OPENSSL

Si se establece en ON, el SDK se compila con OpenSSL; de lo contrario, utiliza [awslabs/aws-lc](https://github.com/aws-lc). AWS-LC es una biblioteca criptográfica de uso general de cuyo mantenimiento se encarga el equipo de AWS Cryptography para AWS y sus clientes. Si se establece en OFF, el parámetro instala AWS-LC como reemplazo de OpenSSL en el directorio predeterminado del sistema. No use este parámetro si ya tiene una instalación de OpenSSL en su sistema.

Valores

ON | OFF

Predeterminado/a

ON

USE_TLS_V1_2

Si se establece en ON, el cliente HTTP aplica TLS 1.2.

Valores

ON | OFF

Predeterminado/a

ON

USE_TLS_V1_3

Si se establece en ON, el cliente HTTP aplica TLS 1.3.

Valores

ON | OFF

Predeterminado/a

OFF

Variables y opciones de CMake para Android

Utilice las siguientes variables cuando cree una compilación del SDK para Android (cuando [TARGET_ARCH](#) esté configurado en ANDROID).

Temas

- [ANDROID_ABI](#)
- [ANDROID_BUILD_CURL](#)
- [ANDROID_BUILD_OPENSSL](#)
- [ANDROID_BUILD_ZLIB](#)
- [ANDROID_NATIVE_API_LEVEL](#)
- [ANDROID_STL](#)
- [ANDROID_TOOLCHAIN_NAME](#)
- [DISABLE_ANDROID_STANDALONE_BUILD](#)
- [NDK_DIR](#)

ANDROID_ABI

Solo Android. Controla la interfaz binaria de aplicaciones (ABI) para la que se generará el código.

Note

Actualmente, no se admiten todos los valores de ABI para Android válidos.

Valores

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

Predeterminado/a

armeabi-v7a

ANDROID_BUILD_CURL

Solo Android. Si se establece en ON, compile también curl.

Valores

ON | OFF

Predeterminado/a

ON

ANDROID_BUILD_OPENSSL

Solo Android. Si se establece en ON, compile también Openssl.

Valores

ON | OFF

Predeterminado/a

ON

ANDROID_BUILD_ZLIB

Solo Android. Si se establece en ON, compile también Zlib.

Valores

ON | OFF

Predeterminado/a

ON

ANDROID_NATIVE_API_LEVEL

Solo Android. Controla el nivel de API en relación con el cual que se compila el SDK. Si configura [ANDROID_STL](#) como gnuSTL, puede elegir cualquier nivel de API. Si usa libc++, debe usar un nivel de API de 21 como mínimo.

Predeterminado/a

Varía según la elección de STL.

ANDROID_STL

Solo Android. Controla la versión de la biblioteca estándar de C++ que utiliza el SDK.

Important

Se pueden producir problemas de rendimiento en el SDK si se utilizan las opciones de `gnustl`; se recomienda encarecidamente utilizar `libc++_shared` o `libc++_static`.

Valores

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

Predeterminado/a

`libc++_shared`

ANDROID_TOOLCHAIN_NAME

Solo Android. Controla qué compilador se usa para compilar el SDK.

Note

Dado que el NDK de Android ha dejado de usar GCC, le recomendamos usar el valor predeterminado.

Predeterminado/a

`standalone-clang`

DISABLE_ANDROID_STANDALONE_BUILD

Solo Android. De forma predeterminada, las compilaciones de Android utilizan una cadena de herramientas independiente basada en clang creada mediante scripts del NDK. Para usar su propia cadena de herramientas, establezca esta opción en ON.

Valores

`ON` | `OFF`

Predeterminado/a

OFF

NDK_DIR

Solo Android. Especifica una ruta de anulación en la que el sistema de compilación debe encontrar el NDK de Android. De forma predeterminada, el sistema de compilación comprueba las variables de entorno (ANDROID_NDK) si esta variable no está configurada.

Configuración y uso de registros en el AWS SDK para C++

AWS SDK para C++ incluye un registro configurable que genera un registro de las acciones realizadas por el SDK durante la ejecución. Para habilitar el registro, defina el `SDKOptions` de `LogLevel` con la nivel de detalle adecuado para su aplicación.

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

Hay siete niveles de detalle entre los que puede elegir. El valor predeterminado es `Off` y no se generará ningún registro. `Trace` generará el mayor nivel de detalle y `Fatal` generará la menor cantidad de mensajes que informen solo de situaciones de error graves.

Una vez activado el registro en la aplicación, el SDK generará los archivos de registro en el directorio del ejecutable siguiendo el patrón de nomenclatura predeterminado de `aws_sdk_<date>.log`. El archivo de registro generado por la opción de nomenclatura de prefijos se transfiere una vez por hora para poder archivar o eliminar los archivos de registro.

Las versiones posteriores del SDK dependen cada vez más de las bibliotecas AWS Common Runtime (CRT) subyacentes. Estas bibliotecas proporcionan funciones comunes y operaciones básicas entre los SDK. Todos los mensajes de registro de las bibliotecas CRT se redirigirán al SDK para C++ de forma predeterminada. El nivel de registro y el sistema de registro que especifique para el SDK para C++ se aplicarán también al CRT.

En el ejemplo anterior, el CRT heredará `LogLevel::Info` y registrará también los mensajes del nivel `Info` en el mismo archivo.

Puede controlar de forma independiente el registro de las bibliotecas CRT, ya sea redirigiendo su salida a un archivo de registro independiente o configurando un nivel de registro diferente para los

mensajes del CRT. A menudo, puede resultar práctico reducir el nivel de detalle de las bibliotecas CRT para que no sobrecarguen los registros. Por ejemplo, el nivel de registro solo para la salida CRT se puede establecer en Warn de la siguiente manera:

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

Si opta por utilizar el método `InitializeAWSLogging`, puede controlar el nivel de detalle y la salida de registro del `DefaultLogSystem`. Puede configurar el prefijo del nombre del archivo de registro o redireccionar la salida a un flujo en lugar de a un archivo.

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

O también, en lugar de usar el `DefaultLogSystem`, puede usar este método para proporcionar su propia implementación de registro.

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

Si llama al método `InitializeAWSLogging`, libere los recursos al final de su programa llamando a `ShutdownAWSLogging`.

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

Ejemplo de prueba de integración con registro

```
#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;
}

```

Ejemplo de subclase de **Aws::Utils::Logging::DefaultLogSystem** para un registro personalizado

El siguiente código muestra cómo subclasificar la clase

Aws::Utils::Logging::DefaultLogSystem, que forma parte del AWS SDK para C++. En este ejemplo, se anula la función `ProcessFormattedStatement` virtual para personalizar el registro.

Aws::Utils::Logging::DefaultLogSystem es una de las diversas clases de AWS SDK para C++ que subclasifica **Aws::Utils::Logging::LogSystemInterface** para un registro personalizado.

```

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;
}
```

Consulte el [ejemplo completo](#) en GitHub.

Anulación del cliente HTTP en el AWS SDK para C++

El cliente HTTP predeterminado para Windows es [WinHTTP](#). El cliente HTTP predeterminado para todas las demás plataformas es [curl](#).

Si lo desea, puede anular el valor predeterminado del cliente HTTP creando una configuración de `HttpClientFactory` personalizada para transferirla al constructor de cualquier cliente de servicio. Para anular el cliente HTTP, el SDK debe compilarse con compatibilidad con curl. La compilación con compatibilidad con curl se realiza de forma predeterminada en Linux y macOS, pero se requieren pasos adicionales para hacerlo en Windows. Para obtener más información sobre cómo crear el SDK en Windows con compatibilidad con curl, consulte [Compilación del AWS SDK para C++ en Windows](#).

Control de los iostreams utilizados por el `HttpClient` y el `AWSCli` en el AWS SDK para C++

De forma predeterminada, todas las respuestas utilizan un flujo de entrada respaldado por una función `stringbuf`. Si es necesario, puede anular el comportamiento predeterminado. Por ejemplo, si utiliza un `GetObject` de Amazon S3 y no quiere cargar todo el archivo en la memoria, puede utilizar `IOStreamFactory` en `AmazonWebServiceRequest` para pasar una función lambda para crear un flujo de archivos.

Ejemplo de solicitud de flujo de archivos

```
//! 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

Hay más en GitHub. Busque el ejemplo completo en el [Repositorio de ejemplos de código de AWS](#).

Uso de una biblioteca libcrypto personalizada en el AWS SDK para C++

De forma predeterminada, el AWS SDK para C++ utiliza la biblioteca criptográfica del sistema predeterminada para la seguridad de la capa de transporte. Sin embargo, el SDK para C++ se puede configurar opcionalmente para usar una biblioteca libcrypto diferente al compilar el SDK desde el código fuente. Funcionalmente, esto significa que todas las operaciones criptográficas se desviarán a una implementación personalizada de OpenSSL. Por ejemplo, es posible que desee utilizar la biblioteca [AWS-LC](#) en [modo FIPS](#) para lograr un estándar FIPS en su aplicación.

Cómo compilar una libcrypto personalizada en el SDK para C++

Paso 1: Compile u obtenga su biblioteca libcrypto

[AWS-LC](#) es un ejemplo de una biblioteca libcrypto alternativa, pero cualquier distribución de OpenSSL o equivalente a OpenSSL funcionaría.

Tanto el SDK para C++ como su dependencia, el CRT, utilizan libcrypto para sus funciones criptográficas y ambos necesitan gestionar las dependencias de la misma manera. El SDK para C++ depende de dos clientes HTTP diferentes en función de si la solicitud utiliza la funcionalidad CRT S3 del SDK. El CRT depende específicamente de [s2n](#), una implementación de TLS que se inicializa en el momento del inicio. Tanto el SDK como el equipo de s2n tienen un parámetro cmake para forzar el uso de una biblioteca libcrypto compartida, independientemente del valor de [BUILD_SHARED_LIBS](#). Lo normal es que el cliente HTTP de CRT y el cliente HTTP normal usen la misma libcrypto. En este caso, eso significaría que ambos hacen referencia a OpenSSL en el árbol de dependencias. El SDK proporciona esto a través de [AWS_USE_CRYPTO_SHARED_LIBS](#) y s2n (para llamadas basadas en CRT) a través de [S2N_USE_CRYPTO_SHARED_LIBS](#). La resolución de las dependencias es la misma entre estas dos bibliotecas y, por lo general, están configuradas para que coincidan, aunque puede configurarlas de forma explícita para que sean diferentes.

Por ejemplo, para usar AWS-LC como biblioteca libcrypto, debería compilarla de la siguiente manera:

```
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_CXX_COMPILER=g++ \  
    -DCMAKE_CXX_FLAGS=-std=c++11
```

```
-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 .
```

Paso 2: Compile curl desde el código fuente o utilice una distribución de curl con su biblioteca libcrypto

El SDK para C++ requiere que se instale un cliente HTTP en el sistema que se utilizará para realizar solicitudes HTTP. El cliente HTTP debe crearse con la libcrypto que vaya a utilizar. El cliente HTTP se encarga de las operaciones de TLS y, por lo tanto, utiliza su biblioteca libcrypto.

En el ejemplo siguiente, la biblioteca curl se reconstruye con una versión instalada de 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
```

Paso 3: Compile el SDK con las bibliotecas libcrypto y curl

El SDK para C++ ahora se puede compilar con los artefactos libcrypto y curl creados anteriormente. Esta compilación del SDK utilizará la biblioteca libcrypto personalizada para todas las funciones criptográficas.

```
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 .
```

Veámoslo todo en una imagen de Docker

El siguiente ejemplo de archivo Docker muestra cómo implementar estos pasos en el entorno Amazon Linux 2023.

```
# 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 && \  
  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 .
```

```
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 .
```

Utilización del AWS SDK para C++

En esta sección, se proporciona información sobre el uso general del AWS SDK para C++ e información adicional respecto de [Introducción al uso del AWS SDK para C++](#).

Para ver ejemplos de programación específicos de servicios, consulte [Ejemplos de código de AWS SDK para C++](#).

Temas

- [Inicialización y cierre del AWS SDK para C++](#)
- [Realización de solicitudes de Servicio de AWS con el AWS SDK para C++](#)
- [Programación asíncrona con el AWS SDK para C++](#)
- [Módulos de utilidades disponibles en el AWS SDK para C++](#)
- [Administración de la memoria en el AWS SDK para C++](#)
- [Gestión de errores en el AWS SDK para C++](#)

Inicialización y cierre del AWS SDK para C++

Las aplicaciones que utilicen el AWS SDK para C++ deben inicializarlo. Del mismo modo, antes de que finalice la aplicación, se debe cerrar el SDK. Ambas operaciones aceptan opciones de configuración que afectan a los procesos de inicialización y cierre y a las llamadas posteriores al SDK.

Todas las aplicaciones que utilicen el AWS SDK para C++ deben incluir el archivo `aws/core/Aws.h`.

El AWS SDK para C++ se debe inicializar mediante una llamada a `Aws::InitAPI`. Antes de que la aplicación finalice, el SDK se debe cerrar llamando a `Aws::ShutdownAPI`. Cada método acepta un argumento de [Aws::SDKOptions](#). Todas las demás llamadas al SDK se pueden realizar entre estas dos llamadas de métodos.

Todas las llamadas del AWS SDK para C++ realizadas entre **`Aws::InitAPI`** y **`Aws::ShutdownAPI`** deben estar incluidas entre corchetes o deben invocarse mediante funciones a las que se llama entre los dos métodos.

A continuación, se muestra una aplicación esqueleto básica.

```
#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;
}
```

El SDK para C++ y sus dependencias utilizan objetos estáticos de C++, y el estándar de C++ no determina el orden de destrucción de los objetos estáticos. Para evitar problemas de memoria causados por el orden no determinista de destrucción de las variables estáticas, no agrupe las llamadas a la **Aws::InitAPI** y la **Aws::ShutdownAPI** dentro de otro objeto estático.

Realización de solicitudes de Servicio de AWS con el AWS SDK para C++

Para acceder a los Servicios de AWS mediante programación, los SDK utilizan una clase de cliente u objeto para cada Servicio de AWS. Por ejemplo, si su aplicación necesita acceder a Amazon EC2, su aplicación crearía un objeto de cliente de Amazon EC2 para interactuar con ese servicio. A continuación, utiliza el cliente de servicio para realizar solicitudes al mismo Servicio de AWS.

Para realizar una solicitud a un Servicio de AWS, primero debe crear y [configurar](#) un cliente de servicio. Para cada Servicio de AWS que utilice su código, tiene su propia biblioteca y su propio tipo específico para interactuar con él. El cliente expone un método para cada operación de API expuesta por el servicio.

El espacio de nombres de una clase de cliente sigue la convención `Aws::Service::ServiceClient`. Por ejemplo, la clase de cliente de AWS Identity and Access Management (IAM) es `Aws::IAM::IAMClient` y la clase de cliente de Amazon S3 es `Aws::S3::S3Client`.

Todas las clases de clientes para todos los servicios de AWS son seguros para subprocesos.

Al crear una instancia de una clase de cliente, se deben proporcionar las credenciales de AWS. Las credenciales se pueden proporcionar desde el código, el entorno o el archivo `config` de AWS y el

archivo `credentials` compartido. Para obtener más información sobre las credenciales, consulte [las instrucciones para configurar la autenticación recomendada en IAM Identity Center](#) o utilice [otro proveedor de credenciales que esté disponible](#).

Programación asíncrona con el AWS SDK para C++

Métodos asíncronos para el SDK

El SDK para C++ proporciona para muchos métodos versiones síncronas y asíncronas. Un método es asíncrono si incluye el sufijo `Async` en su nombre. Por ejemplo, el método `PutObject` de Amazon S3 es síncrono y `PutObjectAsync` es asíncrono.

Como todas las operaciones asíncronas, un método asíncrono del SDK devuelve resultados antes de que finalice su tarea principal. Por ejemplo, el método `PutObjectAsync` devuelve resultados antes de que termine de cargar el archivo en el bucket de Amazon S3. Mientras continúa la operación de carga, la aplicación puede realizar otras operaciones, incluido llamar a otros métodos asíncronos. La aplicación recibe una notificación de que una operación asíncrona ha finalizado cuando se invoca una función de devolución de llamada asociada.

En las siguientes secciones se describe un ejemplo de código que muestra la llamada al método asíncrono `PutObjectAsync`. Cada sección se centra en partes individuales del [archivo de origen completo](#) del ejemplo.

Llamada a métodos asíncronos del SDK

Por lo general, la versión asíncrona de un método del SDK acepta los siguientes argumentos.

- Una referencia al mismo objeto de tipo `Request` que su homólogo síncrono.
- Una referencia a una función de devolución de llamada del controlador de respuestas. Esta función de devolución de llamada se invoca cuando finaliza la operación asíncrona. Uno de los argumentos contiene el resultado de la operación.
- Un `shared_ptr` opcional para un objeto `AsyncCallerContext`. El objeto se transfiere a la devolución de llamada del controlador de respuestas. Incluye una propiedad `UUID` que se puede usar para transferir información de texto a la devolución de llamada.

El método `uploadFileAsync` que se muestra a continuación configura y llama al método `PutObjectAsync` de Amazon S3 del SDK para cargar de forma asíncrona un archivo en un bucket de Amazon S3.

La función recibe referencias a un objeto `S3Client` y a un objeto `PutObjectRequest`. Las recibe de la función principal porque tenemos que asegurarnos de que estos objetos existan durante todas las llamadas asíncronas.

Se asigna un `shared_ptr` a un objeto `AsyncCallerContext`. Su propiedad `UUID` se establece en el nombre del objeto de Amazon S3. Solo para fines de demostración, la devolución de llamada del controlador de respuestas accede a la propiedad y genera su valor.

La llamada a `PutObjectAsync` incluye un argumento de referencia a la función de devolución de llamada del controlador de respuestas `uploadFileAsyncFinished`. Esta función de devolución de llamada se analiza con más detalle en la siguiente sección.

```
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;
}
```

```
}
```

Los recursos de una operación asíncrona deben existir hasta que finalice la operación. Por ejemplo, los objetos de cliente y de solicitud deben existir hasta que la aplicación reciba la notificación de que se ha completado la operación. La aplicación en sí misma no puede finalizar hasta que se complete la operación asíncrona.

Por este motivo, el método `uploadFileAsync` acepta referencias a objetos `S3Client` y `PutObjectRequest` en lugar de crearlos en el método `uploadFileAsync` y almacenarlos en una variable local.

En el ejemplo, el método `PutObjectAsync` devuelve a la persona que llama inmediatamente después de iniciar la operación asíncrona, lo que permite a la cadena de llamada realizar tareas adicionales mientras la operación de carga está en curso.

Si el cliente estuviera almacenado en una variable local del método `uploadFileAsync`, quedaría fuera del alcance cuando el método devolviera un resultado. Sin embargo, el objeto de cliente debe seguir existiendo hasta finalizar la operación asíncrona.

Notificación de la finalización de una operación asíncrona

Cuando finaliza una operación asíncrona, se invoca una función de devolución de llamada del controlador de respuestas de la aplicación. Esta notificación incluye el resultado de la operación. El resultado está contenido en la misma clase de tipo `Resultado` que ha devuelto el método equivalente síncrono. En el ejemplo de código, el resultado está en un objeto `PutObjectOutcome`.

La función de devolución de llamada del controlador de respuestas del ejemplo `uploadFileAsyncFinished` se muestra a continuación. Comprueba si la operación asíncrona se realizó correctamente o no. Utiliza `std::condition_variable` para notificar al subproceso de la aplicación que la operación asíncrona ha finalizado.

```
// 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();
}

```

Una vez finalizada la operación asíncrona, se pueden liberar los recursos asociados a ella. La aplicación también puede finalizar si lo desea.

El siguiente código muestra cómo utiliza una aplicación los métodos `uploadFileAsync` y `uploadFileAsyncFinished`.

La aplicación asigna los objetos `S3Client` y `PutObjectRequest` para que sigan existiendo hasta que finalice la operación asíncrona. Tras la llamada `uploadFileAsync`, la aplicación puede realizar las operaciones que desee. Por motivos de simplificación, en el ejemplo se utilizan `std::mutex` y `std::condition_variable` para esperar a que la devolución de llamada del controlador de respuestas notifique que la operación de carga ha finalizado.

```

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;
}
```

Consulte el [ejemplo completo](#) en Github.

Módulos de utilidades disponibles en el AWS SDK para C++

El AWS SDK para C++ incluye muchos [módulos de utilidades](#) para reducir la complejidad del desarrollo de las aplicaciones de AWS en C++.

Pila HTTP

Una pila HTTP que proporciona una agrupación de conexiones, es segura para subprocessos y se puede reutilizar según sea necesario. Para obtener más información, consulte [Configuración de clientes de AWS](#).

Encabezados	/aws/core/http/
Documentación de la API	Aws::Http

Utilidades de cadenas

Funciones principales de cadenas, como `trim`, `lowercase` y conversiones numéricas.

Encabezado	aws/core/utils/StringUtils.h
Documentación de la API	Aws::Utils::StringUtils

Utilidades de hash

Funciones de hash, como `SHA256`, `MD5`, `Base64` y `SHA256_HMAC`.

Encabezado	/aws/core/utils/HashingUtils.h
------------	--

[Documentación de la API](#)[Aws::Utils::HashingUtils](#)

Analizador JSON

Un analizador JSON totalmente funcional pero ligero (un encapsulador ligero alrededor de *cJSON*).

[Encabezado](#)[/aws/core/utils/json/JsonSerializer.h](#)[Documentación de la API](#)[Aws::Utils::Json::JsonValue](#)

Analizador XML

Un analizador XML ligero (un encapsulador ligero alrededor de *tinyxml2*). Se ha añadido el [patrón RAII](#) a la interfaz.

[Encabezado](#)[/aws/core/utils/xml/XmlSerializer.h](#)[Documentación de la API](#)[Aws::Utils::Xml](#)

Administración de la memoria en el AWS SDK para C++

El AWS SDK para C++ ofrece una manera de controlar la asignación y desasignación de memoria en una biblioteca.

Note

La administración de memoria personalizada solo está disponible si utiliza una versión de la biblioteca creada con la constante de tiempo de compilación definida `USE_AWS_MEMORY_MANAGEMENT`.

Si utiliza una versión de la biblioteca creada sin la constante de tiempo de compilación, las funciones del sistema de memoria global, como `InitializeAWSMemorySystem`, no funcionarán; en su lugar, se utilizarán las funciones globales `new` y `delete`.

Para obtener más información sobre la constante de tiempo de compilación, consulte [STL y cadenas y vectores de AWS](#).

Asignación y desasignación de memoria

Para asignar o desasignar memoria

1. SubclaseMemorySystemInterface: `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

Puede cambiar la firma de tipo de `AllocateMemory` si es necesario.

2. Utilice la estructura `Aws::SDKOptions` para configurar el uso del administrador de memoria personalizado. Transfiera la instancia de la estructura a `Aws::InitAPI`. Antes de que la aplicación finalice, se debe cerrar el SDK llamando a `Aws::ShutdownAPI` con la misma instancia.

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

    // ... do stuff

    Aws::ShutdownAPI(options);

    return 0;
}
```

```
}
```

STL y cadenas y vectores de AWS

Cuando se inicializa con un administrador de memoria, el AWS SDK para C++ desvía todas las asignaciones y desasignaciones al administrador de memoria. Si no existe un administrador de memoria, el SDK usa `global new` y `delete`.

Si utiliza asignadores STL personalizados, debe modificar las firmas de tipos de todos los objetos STL para que coincidan con la política de asignación. Como el STL ocupa un lugar destacado en la implementación y la interfaz del SDK, un enfoque único en el SDK impediría la transferencia directa de los objetos STL predeterminados al SDK o el control de la asignación de STL. El uso de una alternativa, como un enfoque híbrido (utilizar asignadores personalizados internamente y permitir la presencia de objetos STL estándares y personalizados en la interfaz), podría dificultar la investigación de los problemas de memoria.

La solución consiste en utilizar la constante de tiempo de compilación `USE_AWS_MEMORY_MANAGEMENT` del sistema de memoria para controlar los tipos de STL que utiliza el SDK.

Si la constante de tiempo de compilación está habilitada (on), los tipos se convierten en tipos STL con un asignador personalizado conectado al sistema de memoria de AWS.

Si la constante de tiempo de compilación está deshabilitada (off), todos los tipos `Aws::*` se resuelven con el tipo `std::*` predeterminado correspondiente.

Código de ejemplo del archivo **`AWSAllocator.h`** del SDK

```
#ifndef 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
```

En el código de ejemplo, `AwsAllocator` puede ser un asignador personalizado o un asignador predeterminado, en función de la constante de tiempo de compilación.

Código de ejemplo del archivo **AWSVector.h** del SDK

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

En el código de ejemplo, definimos los tipos `Aws::*`.

Si la constante de tiempo de compilación está habilitada (on), el tipo se asigna a un vector mediante la asignación de memoria personalizada y el sistema de memoria de AWS.

Si la constante de tiempo de compilación está desactivada (off), el tipo se asigna a un `std::vector` normal con los parámetros de tipo predeterminados.

El alias de tipos se usa con todos los tipos `std::` del SDK que realizan la asignación de memoria; por ejemplo, los contenedores, los flujos de cadenas y los búferes de cadenas. El AWS SDK para C++ utiliza estos tipos.

Problemas pendientes

Puede controlar la asignación de memoria en el SDK; sin embargo, los tipos de STL siguen dominando la interfaz pública a través de los parámetros `initialize` del objeto y los métodos `set` del modelo. Si no utiliza STL y usa cadenas y contenedores en su lugar, tendrá que crear muchos archivos temporales cada vez que desee realizar una llamada de servicio.

Para eliminar la mayoría de los archivos temporales y la asignación al realizar llamadas de servicio que no utilizan STL, hemos implementado lo siguiente:

- Cada función `Init/Set` que toma una cadena tiene una sobrecarga que requiere un `const char*`.
- Cada función `Init/Set` que toma un contenedor (mapa/vector) tiene una variante de adición que usa una sola entrada.
- Cada función `Init/Set` que toma datos binarios tiene una sobrecarga que dirige un puntero a los datos y a un valor `length`.
- (Opcional) Cada función `Init/Set` que toma una cadena tiene una sobrecarga que toma un valor `const char*` con una terminación distinta de cero y un `length`

Desarrolladores de SDK nativos y controles de memoria

En el código del SDK, siga estas reglas:

- No use `new` y `delete`; utilice `Aws::New<>` y `Aws::Delete<>` en su lugar.
- No use `new[]` y `delete[]`; utilice `Aws::NewArray<>` y `Aws::DeleteArray<>`.
- No use `std::make_shared`; utilice `Aws::MakeShared`.
- Utilice `Aws::UniquePtr` para tener punteros únicos a un solo objeto. Utilice la función `Aws::MakeUnique` para crear el puntero único.
- Utilice `Aws::UniqueArray` para tener punteros únicos a una matriz de objetos. Utilice la función `Aws::MakeUniqueArray` para crear el puntero único.
- No utilice contenedores STL directamente; use uno de los typedefs de `Aws::` o añada un typedef para el contenedor que desee. Por ejemplo:

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

- Utilice `shared_ptr` con cualquier puntero externo transferido al SDK y administrado por él. Debe inicializar el puntero compartido con una política de destrucción que coincida con la manera en la que se asignó el objeto. Puede usar un puntero sin procesar si no se espera que el SDK limpie el puntero.

Gestión de errores en el AWS SDK para C++

El AWS SDK para C++ no utiliza excepciones; sin embargo, usted puede utilizarlas en su código. Cada cliente de servicio devuelve un objeto `outcome` que incluye el resultado y un código de error.

Ejemplo de manejo de condiciones de error

```
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;
}
```


Llamada a Servicios de AWS desde el AWS SDK para C++

Las siguientes secciones contienen ejemplos, tutoriales, tareas y guías que muestran cómo utilizar AWS SDK para C++ para trabajar con servicios de AWS.

Si está empezando con el AWS SDK para C++, le recomendamos que lea primero el tema [Introducción](#).

Tiene más ejemplos de código en la [carpeta de ejemplos de C++](#) en GitHub.

Y hay más ejemplos en el capítulo [Ejemplos de código](#) de esta guía o en el [Repositorio de ejemplos de código de AWS](#) en GitHub.

Temas

- [Introducción a los ejemplos de código del AWS SDK para C++](#)
- [Introducción a la resolución de errores de tiempo de ejecución en el AWS SDK para C++](#)
- [Ejemplos guiados de llamadas a los Servicios de AWS mediante el AWS SDK para C++](#)

Introducción a los ejemplos de código del AWS SDK para C++

Estructura de los ejemplos de código

La [carpeta de ejemplos de C++](#) en Github contiene carpetas de proyectos para cada servicio de AWS. Por lo general, los archivos de código fuente .cpp individuales en las carpetas muestran una función o acción específica para ese servicio. Por ejemplo, en Amazon DynamoDB, obtener un elemento de la base de datos y cargar un elemento en la base de datos son dos tipos de acción diferentes, por lo que hay un archivo independiente para cada uno de ellos en la carpeta de DynamoDB: `get_item.cpp` y `put_item.cpp`. Cada archivo.cpp contiene una función `main()` como punto de entrada a un ejecutable independiente. Los ejecutables del proyecto se generan en una carpeta designada por el sistema de compilación y hay un archivo ejecutable para cada archivo de origen de ejemplo. El nombre del ejecutable sigue las convenciones de la plataforma (`{name}.exe` o simplemente `{name}`) y se aplica cualquier prefijo personalizado `CMakeLists.txt`, como `run_`.

Para ejecutar una funcionalidad de ejemplo

1. Descargue el ejemplo de código deseado en el [Repositorio de ejemplos de código de AWS](#) en GitHub.
2. Abra un archivo.cpp para explorar su función `main()` y cualquier método al que se haya llamado.
3. Cree el proyecto, siguiendo el ejemplo para empezar en [Primeros pasos con el AWS SDK para C++](#). Tenga en cuenta que la creación del proyecto genera un ejecutable por cada archivo de origen del proyecto.
4. Ejecute el archivo ejecutable para la funcionalidad seleccionada.
 - En una línea de comandos, ejecute ese programa mediante el archivo ejecutable en función del nombre del archivo `*.cpp`.
 - Si trabaja en un IDE, elija el archivo `.cpp` de la funcionalidad que desee mostrar y selecciónelo como opción de inicio (u objeto de inicio).

Pruebas unitarias

Las pruebas de los ejemplos se escriben con el marco GoogleTest. Para obtener más información, consulte [GoogleTest Primer](#) en el sitio web de GoogleTest.

Las pruebas unitarias de cada ejemplo se encuentran en una `tests` subcarpeta que contiene su propio archivo `CMakeLists.txt`. Para cada archivo de origen del ejemplo, hay un archivo de prueba correspondiente denominado `gtest_<source file>`. El ejecutable de prueba de la subcarpeta se llama `<Servicio de AWS>_gtests`.

Archivo CMakeLists.txt

La carpeta de cada servicio contiene un archivo denominado `CMakeLists.txt`. Muchos de estos archivos contienen una construcción similar a esta:

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

Para cada archivo `.cpp` de la carpeta, el archivo `CMakeLists.txt` crea un ejecutable (`cmake: add_executable`) con un nombre basado en el nombre del archivo de código fuente sin la extensión del archivo.

Ejemplos de código de compilación y depuración en Visual Studio

Creación y ejecución del ejemplo de código de Amazon S3

1. Obtenga el código fuente de ejemplo de Amazon S3. En este procedimiento se utiliza el ejemplo de código [Ejemplos de código de Amazon S3 con el AWS SDK para C++](#) para empezar a utilizar Visual Studio.
2. En el Explorador de Windows, vaya a la carpeta de s3 (por ejemplo, `\aws-doc-sdk-examples\cpp\example_code\s3`).
3. Haga clic con el botón derecho en la carpeta del ejemplo de s3 y seleccione Abrir con Visual Studio. Los proyectos de Visual Studio para CMake no tienen un archivo de "file", sino que es la carpeta completa.
4. En el menú desplegable Configuration Selector del menú superior de Visual Studio, compruebe que la configuración seleccionada coincida con el tipo de compilación que seleccionó al compilar el SDK con el código fuente. Por ejemplo, debe seleccionar una configuración Debug si realizó la compilación con el código fuente mediante debug (`-DCMAKE_BUILD_TYPE=Debug` en la línea de comandos de CMake de las instrucciones de instalación del SDK).
5. Abra el archivo `CMakeLists.txt`.
6. Haga clic en Guardar. Cada vez que hace clic en Save en el archivo `CMakeLists.txt`, Visual Studio actualiza los archivos generados por CMake. Si se muestra la pestaña Output, podrá ver los mensajes de registro resultantes de esta generación.
 - Debe haber un cuadro desplegable en la pestaña Output que dice: "Show output from:" donde CMake debería ser la opción seleccionada de forma predeterminada.
 - La salida del último mensaje debería ser: "CMake generation finished."
 - Si el último mensaje no es este, el archivo CMake tiene errores. No siga adelante hasta resolverlos. Consulte [Solución de problemas de compilación del AWS SDK para C++](#).
 - Tenga en cuenta que CMake utiliza la caché de CMake para aumentar la velocidad. Si está solucionando problemas con CMake, asegúrese de hacer una limpieza total para que los mensajes de error que reciba reflejen realmente sus cambios más recientes. En Solution Explorer, haga clic con el botón derecho en `CMakeLists.txt` y seleccione CMake Cache y, a continuación, seleccione Delete Cache Hágalo con frecuencia a medida que vaya resolviendo problemas de CMake.
7. Para compilar y ejecutar ejemplos desde Visual Studio, Visual Studio coloca los ejecutables en una estructura de carpetas diferente a la de la línea de comandos. Para ejecutar el

- código, los ejecutables del SDK deben copiarse en el lugar correcto. Busque la línea "TODO" del archivo CMakeLists (aprox. la línea 40) y elija la línea que hemos comentado con anterioridad para usarla en Visual Studio. Visual Studio no usa una subcarpeta dedicada para el tipo de compilación, por lo que no está incluida. Cambie la línea comentada del archivo CMakeLists.txt para usarla en Visual Studio.
8. Elimine la caché de CMake (como se ha explicado anteriormente), haga clic en el archivo CMakeLists.txt para seleccionar/activar la pestaña y vuelva a seleccionar Save en el CMakeLists.txt archivo para empezar a generar los archivos de compilación de CMake.
 9. Abra el archivo de origen del "programa" que desee ejecutar.
 - Por ejemplo, abra `list_buckets.cpp`.
 - La carpeta de ejemplo de Amazon S3 está codificada para que cada "característica" de Amazon S3 que aparezca se muestre en un ejecutable dedicado exclusivamente a esa función. Por ejemplo, `list_buckets.cpp` se convertirá en un ejecutable que solo muestra la lista de buckets.
 10. En el menú de la parte superior, seleccione Build y, a continuación, elija Build All.
 - La opción Show output from de la pestaña Output debería mostrar Build como la opción seleccionada y todos los mensajes de compilación y vinculación.
 - El último resultado debería ser: Build All succeeded."
 - Ahora se generan los ejecutables para cada uno de los archivos de origen individuales. Puede confirmarlo consultando el directorio de salida de la compilación (por ejemplo, `\aws-doc-sdk-examples\cpp\example_code\s3\out\build\x64-Debug`).
 - Tenga en cuenta que los ejecutables llevan el prefijo "run"» porque así lo establece el archivo CMakeLists.txt.
 11. En el menú superior, hay una flecha verde y un selector desplegable para Debug Target. Elegir `run_list_buckets.exe`.
 12. Haga clic en el botón de flecha verde para seleccionar el elemento de inicio.
 13. Se abrirá una ventana de Visual Studio Debug Console donde se mostrará el resultado del código.
 14. Pulse una tecla para cerrar la ventana o ciérrela manualmente para finalizar el programa. También puede establecer puntos de interrupción en el código y, cuando vuelva a hacer clic en ejecutar, se alcanzarán los puntos de interrupción.

Introducción a la resolución de errores de tiempo de ejecución en el AWS SDK para C++

A medida que aprenda a desarrollar aplicaciones con el AWS SDK para C++, también es útil que se familiarice con el uso de la Consola de administración de AWS y de la AWS CLI. Estas herramientas se pueden utilizar indistintamente para diversas tareas de diagnóstico y resolución de problemas cuando se producen errores de tiempo de ejecución.

En el siguiente tutorial, se muestra un ejemplo de estas tareas de diagnóstico y solución de problemas. Se centra en el error `Access denied`, que puede producirse por diversos motivos. El tutorial muestra un ejemplo de cómo se puede determinar la causa real del error. Se centra en dos de las posibles causas: permisos incorrectos del usuario actual y un recurso que no está disponible para el usuario actual.

Para obtener el código fuente y los ejecutables del proyecto

1. Descargue la carpeta de ejemplos de código de Amazon S3 del [Repositorio de ejemplos de código de AWS](#) en GitHub.
2. Abra `delete_bucket.cpp` y observe que hay dos métodos: `main()` y `DeleteBucket()`. `DeleteBucket()` usa el SDK para eliminar el bucket.
3. Cree el ejemplo de Amazon S3 siguiendo los mismos pasos de compilación que se explican en [Introducción al uso del AWS SDK para C++](#). El proceso de compilación genera un ejecutable para cada archivo de origen.
4. Abra una línea de comandos en la carpeta en la que el sistema de compilación ha generado los ejecutables de la compilación. Ejecute el archivo ejecutable `run_create_bucket` (el nombre del archivo ejecutable real variará en función del sistema operativo). Se creará un bucket en su cuenta (para que tenga uno para eliminar).
5. En el símbolo del sistema, ejecute el archivo ejecutable `run_delete_bucket`. En este ejemplo, el valor previsto es el parámetro del bucket que desea eliminar. Introduzca un nombre de depósito incorrecto; escriba mal de manera intencionada el nombre de este bucket para que podamos analizar la solución de problemas.
6. Confirme que recibe el mensaje de error `Access Denied`. El mensaje de error `Access Denied` le preguntará si ha creado un usuario con todos los permisos para Amazon S3, algo que debe verificar a continuación.

Para instalar la AWS CLI y buscar el nombre de usuario que realiza las llamadas a AWS

1. Para instalar la AWS CLI más reciente en el equipo local, consulte [Instalación de la AWS CLI](#) en la Guía del usuario de AWS Command Line Interface.
2. Para comprobar que la AWS CLI funciona, abra una línea de comandos y ejecute el comando `aws -\-version`.

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

3. Para obtener el nombre de usuario que está realizando las llamadas realmente a AWS, ejecute el comando `aws sts get-caller-identity` de la AWS CLI. En el siguiente resultado de ejemplo, ese nombre de usuario es `userX`.

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

Hay muchas formas de especificar las credenciales, pero si ha seguido el procedimiento incluido en [Autenticación AWS mediante AWS SDK para C++](#), este nombre de usuario proviene de su archivo de credenciales de AWS compartido. Durante ese procedimiento, concedió a su usuario los permisos de `AmazonS3FullAccess`.

Note

Por lo general, la mayoría de los comandos de la AWS CLI siguen esta estructura sintáctica:

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

donde el *comando* es el servicio y el *subcomando* es el método al que se llama en ese servicio. Para obtener más información, consulte [Estructura de comandos en la AWS CLI](#) en la Guía del usuario de la AWS Command Line Interface.

Para comprobar si un usuario tiene permiso para eliminar un bucket

1. Abra la [Consola de administración de AWS](#) e inicie sesión. Para obtener más información, consulte [Introducción a la Consola de administración de AWS](#).
2. En la barra de navegación principal, en Buscar servicios..., introduzca **IAM** y seleccione el servicio de IAM en los resultados.
3. En la barra lateral del panel de control o en Recursos de IAM, seleccione Usuarios.
4. En la tabla de usuarios disponibles para su cuenta, seleccione el nombre de usuario obtenido en el procedimiento anterior.
5. Seleccione la pestaña Permisos en la página de Resumen y en la tabla de Nombre de la política, seleccione AmazonS3FullAccess.
6. Consulte el resumen de la política y los datos de JSON. Compruebe que este usuario tiene acceso completo al servicio Amazon S3.

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

Este proceso de eliminación es habitual para descartar dónde podría estar el problema. En este caso, ha comprobado que el usuario tiene los permisos correctos, por lo que el problema debe ser de otra índole. Es decir, dado que tiene los permisos correctos para acceder a los buckets, el error `Access Denied` puede significar que está intentando acceder a un bucket que no es suyo. Al solucionar el problema, debe revisar el nombre del bucket que ha proporcionado al programa y comprobar que en su cuenta no exista ningún depósito con ese nombre y que, por lo tanto, le impida "acceder" a él.

Para actualizar el ejemplo de código para que se ejecute correctamente

1. Vuelva a la función `main()` de `delete_bucket.cpp`, y utilice la enumeración para cambiar la región por la región de su cuenta. Para encontrar la región de su cuenta, inicie sesión en la Consola de administración de AWS y localice la región en la esquina superior derecha. También en `main()`, puede cambiar el nombre del bucket por uno que ya exista en su cuenta. Existen varias maneras de encontrar los nombres de los buckets actuales:
 - Puede usar el ejecutable `run_list_buckets` que también existe en la carpeta de este ejemplo de código para obtener mediante programación los nombres de sus buckets.

- También puede utilizar el siguiente comando de la AWS CLI para obtener una lista de buckets de Amazon S3.

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

- O bien, puede utilizar el alias [Consola de administración de AWS](#). En la barra de navegación principal, en Buscar servicios..., introduzca **S3**. En la página Buckets, se muestran los buckets de su cuenta.
2. Vuelva a compilar el código y ejecute el archivo ejecutable `run_delete_bucket` modificado.
 3. Utilice la Consola de administración de AWS o la AWS CLI, para comprobar que el bucket de Amazon S3 que ha creado anteriormente se ha eliminado.

Ejemplos guiados de llamadas a los Servicios de AWS mediante el AWS SDK para C++

Si está empezando con AWS o los ejemplos de código de AWS, le recomendamos que empiece con [Introducción a los ejemplos de código](#).

El código fuente que muestra cómo trabajar con los servicios de AWS mediante el AWS SDK para C++ está disponible en el capítulo [Ejemplos de código](#) de esta guía o directamente en el [Repositorio de ejemplos de código de AWS](#) en GitHub.

En esta sección, se seleccionan varios servicios de AWS y se guía a través de ejemplos en los que se utilizan esos servicios. Los siguientes ejemplos guiados son un subconjunto de lo que está disponible en Github.

Ejemplos de servicios con una explicación adicional (consulte el [Repositorio de ejemplos de código de AWS](#) para ver la lista completa)

Servicio	Resumen de lo que el servicio proporciona al programa
Amazon CloudWatch	Recopila y supervisa las métricas de los recursos de AWS que está utilizando
Amazon DynamoDB	Un servicio de base de datos NoSQL

Servicio	Resumen de lo que el servicio proporciona al programa
Amazon Elastic Compute Cloud (Amazon EC2)	Capacidad de computación segura y de tamaño ajustable
Amazon Simple Storage Service (Amazon S3)	Almacenamiento y recuperación de datos (objetos en buckets)
Amazon Simple Queue Service (Amazon SQS)	Servicio de cola de mensajes para enviar, almacenar y recibir mensajes entre componentes de software

También hay ejemplos que muestran cómo utilizar los [métodos asíncronos](#).

Para proponer un nuevo ejemplo de código al equipo de documentación de AWS, consulte las [directrices para contribuir](#) en GitHub para crear una nueva solicitud. El equipo prefiere crear ejemplos de código que muestren escenarios amplios en lugar de llamadas individuales a la API.

Uso de los ejemplos de código en Windows

Si está compilando los ejemplos en Windows con la versión 1.9 del SDK, consulte [Solución de problemas de compilación del AWS SDK para C++](#).

Ejemplos de Amazon CloudWatch con el AWS SDK para C++

Amazon CloudWatch (CloudWatch) es un servicio de monitorización de recursos de la nube de AWS y las aplicaciones que se ejecutan en AWS. Puede utilizar los siguientes ejemplos para programar [CloudWatch](#) con el AWS SDK para C++.

Amazon CloudWatch monitorea los recursos de AWS y las aplicaciones que ejecuta en AWS en tiempo real. Puede utilizar CloudWatch para recopilar y hacer un seguimiento de métricas, que son las variables que puede medir en los recursos y aplicaciones. Las alarmas de CloudWatch envían notificaciones o efectúan cambios automáticamente en los recursos que está supervisando basándose en las reglas que defina.

Para obtener más información acerca de CloudWatch, consulte la [Guía del usuario de Amazon CloudWatch](#).

Note

En esta Guía solo se proporciona el código necesario para demostrar algunas técnicas, pero el [código de ejemplo completo está disponible en GitHub](#). En GitHub, puede descargar un único archivo de origen o clonar el repositorio localmente para obtener todos los ejemplos para compilarlos y ejecutarlos.

Temas

- [Obtención de métricas de CloudWatch](#)
- [Publicación de datos de métricas personalizadas](#)
- [Uso de alarmas de CloudWatch](#)
- [Uso de acciones de alarma en CloudWatch](#)
- [Envío de eventos a CloudWatch](#)

Obtención de métricas de CloudWatch

Requisitos previos

Antes de comenzar, le recomendamos que lea [Primeros pasos con el AWS SDK para C++](#).

Descargue el código de ejemplo y compile la solución tal y como se describe en [Introducción a los ejemplos de código](#).

Para ejecutar los ejemplos, el perfil de usuario que su código utilice para realizar las solicitudes debe tener los permisos adecuados en AWS (para el servicio y la acción). Para obtener más información, consulte [Proporcionar credenciales de AWS](#).

Mostrar métricas

Para enumerar las métricas de CloudWatch, cree un objeto [ListMetricsRequest](#) y llame a la función `ListMetrics` de `CloudWatchClient`. Puede utilizar el objeto `ListMetricsRequest` para filtrar las métricas devueltas por espacio de nombres, nombre de métrica o dimensiones.

Note

Puede encontrar una lista de métricas y dimensiones publicadas por los servicios de AWS en la [Referencia de dimensiones y métricas de Amazon CloudWatch](#) en la Guía del usuario de Amazon CloudWatch.

Incluye

```
#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>
```

Código de

```
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();
}

```

Las métricas se devuelven en un objeto [ListMetricsResult](#) llamando a su función `GetMetrics`. Los resultados puede que estén paginados. Para recuperar el siguiente lote de resultados, llame a `SetNextToken` en el objeto de la solicitud original con el valor devuelto de la función `GetNextToken` del objeto `ListMetricsResult` y transfiera el objeto de la solicitud modificado a otra llamada a `ListMetrics`.

Consulte el [ejemplo completo](#)

Más información

- [ListMetrics](#) en la Referencia de la API de Amazon CloudWatch.

Publicación de datos de métricas personalizadas

Algunos servicios de AWS publican [sus propias métricas](#) en espacios de nombres que empiezan por AWS/. También puede publicar datos de métricas personalizadas usando su propio espacio de nombres (siempre y cuando no empiece por AWS/).

Requisitos previos

Antes de comenzar, le recomendamos que lea [Primeros pasos con el AWS SDK para C++](#).

Descargue el código de ejemplo y compile la solución tal y como se describe en [Introducción a los ejemplos de código](#).

Para ejecutar los ejemplos, el perfil de usuario que su código utilice para realizar las solicitudes debe tener los permisos adecuados en AWS (para el servicio y la acción). Para obtener más información, consulte [Proporcionar credenciales de AWS](#).

Publicación de datos de métricas personalizadas

Para publicar sus propios datos de métricas, llame a la función `PutMetricData` de `CloudWatchClient` con un objeto [PutMetricDataRequest](#). El `PutMetricDataRequest` debe incluir el espacio de nombres personalizado que se va a usar para los datos e información sobre el propio punto de datos en un objeto [MetricDatum](#).

Note

No puede especificar un espacio de nombres que comience por AWS/. Los espacios de nombres que empiezan por AWS/ están reservados para su uso por los productos de Amazon Web Services.

Incluye

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

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

Código de

```
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;
}
```

Consulte el [ejemplo completo](#)

Más información

- [Uso de las métricas de Amazon CloudWatch](#) en la Guía del usuario de Amazon CloudWatch.
- [AWS Espacios de nombres](#) en la guía del usuario de Amazon CloudWatch.
- [PutMetricData](#) en la Referencia de la API de Amazon CloudWatch.

Uso de alarmas de CloudWatch

Requisitos previos

Antes de comenzar, le recomendamos que lea [Primeros pasos con el AWS SDK para C++](#).

Descargue el código de ejemplo y compile la solución tal y como se describe en [Introducción a los ejemplos de código](#).

Para ejecutar los ejemplos, el perfil de usuario que su código utilice para realizar las solicitudes debe tener los permisos adecuados en AWS (para el servicio y la acción). Para obtener más información, consulte [Proporcionar credenciales de AWS](#).

Crear una alarma

Para crear una alarma basada en una métrica de CloudWatch, llame a la función `PutMetricAlarm` de `AmazonCloudWatchClient` con un objeto [PutMetricAlarmRequest](#) en el que se especifiquen las condiciones de la alarma.

Incluye

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

Código de

```
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;
}
```

Consulte el [ejemplo completo](#)

Mostrar alarmas

Para mostrar las alarmas de CloudWatch que ha creado, llame a la función `DescribeAlarms` de `AmazonCloudWatchClient` con un objeto [DescribeAlarmsRequest](#) que puede utilizar para establecer opciones para el resultado.

Incluye

```
#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>
```

Código de

```
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();
}
```

La lista de alarmas se puede obtener llamando a `getMetricAlarms` en el objeto [DescribeAlarmsResult](#) que devuelve `DescribeAlarms`.

Los resultados puede que estén paginados. Para recuperar el siguiente lote de resultados, llame a `SetNextToken` en el objeto de la solicitud original con el valor devuelto de la función `GetNextToken` del objeto `DescribeAlarmsResult` y transfiera el objeto de la solicitud modificado a otra llamada a `DescribeAlarms`.

Note

También puede recuperar alarmas para una métrica específica mediante la función `DescribeAlarmsForMetric` de `CloudWatchClient`. Su uso es similar a `DescribeAlarms`.

Consulte el [ejemplo completo](#)

Eliminar alarmas

Para eliminar alarmas de CloudWatch, llame a la función `DeleteAlarms` de `AmazonCloudWatchClient` con un objeto [DeleteAlarmsRequest](#) que contenga uno o más nombres de alarma que desee eliminar.

Incluye

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

Código de

```
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;
}
```

Consulte el [ejemplo completo](#)

Más información

- [Creación de alarmas de Amazon CloudWatch](#) en la Guía del usuario de Amazon CloudWatch
- [PutMetricAlarm](#) en la Referencia de la API de Amazon CloudWatch
- [DescribeAlarms](#) en la Referencia de la API de Amazon CloudWatch
- [DeleteAlarms](#) en la Referencia de la API de Amazon CloudWatch

Uso de acciones de alarma en CloudWatch

Mediante las acciones de alarma de CloudWatch, puede crear alarmas que realicen acciones como detener, terminar, reiniciar, o recuperar automáticamente instancias de Amazon EC2.

Las acciones de alarma se pueden añadir a una alarma mediante la función `SetAlarmActions` de [PutMetricAlarmRequest](#) al [crear una alarma](#).

Requisitos previos

Antes de comenzar, le recomendamos que lea [Primeros pasos con el AWS SDK para C++](#).

Descargue el código de ejemplo y compile la solución tal y como se describe en [Introducción a los ejemplos de código](#).

Para ejecutar los ejemplos, el perfil de usuario que su código utilice para realizar las solicitudes debe tener los permisos adecuados en AWS (para el servicio y la acción). Para obtener más información, consulte [Proporcionar credenciales de AWS](#).

Habilitar acciones de alarma

Para habilitar acciones de alarma para una alarma de CloudWatch, llame a la función `EnableAlarmActions` de `CloudWatchClient` con un objeto [EnableAlarmActionsRequest](#) que contenga uno o varios nombres de alarma cuyas acciones desee habilitar.

Incluye

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

Código de

```
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;
```

Consulte el [ejemplo completo](#)

Deshabilitar acciones de alarma

Para deshabilitar acciones de alarma para una alarma de CloudWatch, llame a la función `DisableAlarmActions` de `CloudWatchClient` con un objeto [DisableAlarmActionsRequest](#) que contenga uno o varios nombres de alarma cuyas acciones desee deshabilitar.

Incluye

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

Código de

```
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;
```

```
}
```

Consulte el [ejemplo completo](#)

Más información

- [Creación de alarmas para parar, terminar, reiniciar o recuperar una instancia](#) en la Guía del usuario de Amazon CloudWatch.
- [PutMetricAlarm](#) en la Referencia de la API de Amazon CloudWatch
- [EnableAlarmActions](#) en la Referencia de la API de Amazon CloudWatch
- [DisableAlarmActions](#) en la Referencia de la API de Amazon CloudWatch

Envío de eventos a CloudWatch

Eventos CloudWatch permite la transmisión casi en tiempo real de eventos del sistema que describen cambios en los recursos de AWS para instancias de Amazon EC2, funciones de Lambda, secuencias de Kinesis, tareas de Amazon ECS, máquinas de estado de Step Functions, temas de Amazon SNS, colas de Amazon SQS o destinos integrados. Mediante reglas sencillas, puede asignar los eventos y dirigirlos a una o más secuencias o funciones de destino.

Note

En estos fragmentos de código, se presupone que conoce la información que se describe en [Introducción al uso del AWS SDK para C++](#) y que ha configurado credenciales de AWS predeterminadas utilizando la información incluida en [Proporcionar credenciales de AWS](#).

Añadir eventos

Para añadir eventos de CloudWatch personalizados, llame a la función `PutEvents` de `CloudWatchEventsClient` con un objeto [PutEventsRequest](#) que contenga uno o varios objetos [PutEventsRequestEntry](#) que proporcionen detalles sobre cada evento. Puede especificar varios parámetros para la entrada como el origen y el tipo del evento, los recursos asociados con el evento, etc.

Note

Puede especificar un máximo de 10 eventos para cada llamada a `putEvents`.

Incluye

```
#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>
```

Código de

```
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;
}
```

Añadir reglas

Para crear o actualizar una regla, llame a la función `PutRule` de `CloudWatchEventsClient` con un objeto [PutRuleRequest](#) con el nombre de la regla y parámetros opcionales como el [patrón del evento](#), el rol de IAM que se va a asociar a la regla y una [expresión de programación](#) que describa con qué frecuencia se ejecuta la regla.

Incluye

```
#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>
```

Código de

```
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;
}
```

Añadir destinos

Los destinos son los recursos que se invocan cuando se activa una regla. Algunos ejemplos de destinos son las instancias de Amazon ECS, funciones de Lambda, flujos de Kinesis, tareas de Amazon EC2, máquinas de estado de Step Functions y destinos integrados.

Para añadir un destino a una regla, llame a la función `PutTargets` de `CloudWatchEventsClient` con un objeto [PutTargetsRequest](#) que contenga la regla para actualizar y la lista de destinos que se van a añadir a la regla.

Incluye


```
#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>
```

Código de

```
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;
}
}
```

Consulte el [ejemplo completo](#)

Más información

- [Agregar eventos con PutEvents](#) en la Guía del usuario de Eventos de Amazon CloudWatch
- [Programar expresiones para reglas](#) en la Guía del usuario de Eventos de Amazon CloudWatch
- [Tipos de evento para Eventos de CloudWatch](#) en la Guía del usuario de Eventos de Amazon CloudWatch

- [Eventos y patrones de eventos](#) en la Guía del usuario de Eventos de Amazon CloudWatch
- [PutEvents](#) en la Referencia de la API de Eventos de Amazon CloudWatch
- [PutTargets](#) en la Referencia de la API de Eventos de Amazon CloudWatch
- [PutRule](#) en la Referencia de la API de Eventos de Amazon CloudWatch

Ejemplos de uso del AWS SDK para C++ con Amazon DynamoDB

Amazon DynamoDB es un servicio de base de datos NoSQL totalmente administrado que ofrece un rendimiento rápido y predecible, así como una perfecta escalabilidad. En los ejemplos siguientes se muestra cómo puede programar [Amazon DynamoDB](#) con el AWS SDK para C++.

Note

En esta Guía solo se proporciona el código necesario para demostrar algunas técnicas, pero el [código de ejemplo completo está disponible en GitHub](#). En GitHub, puede descargar un único archivo de origen o clonar el repositorio localmente para obtener todos los ejemplos para compilarlos y ejecutarlos.

Temas

- [Uso de tablas en DynamoDB](#)
- [Uso de elementos en DynamoDB](#)

Uso de tablas en DynamoDB

Las tablas son los contenedores de todos los elementos de una base de datos de DynamoDB. Para poder añadir o eliminar datos de DynamoDB, debe crear una tabla.

Para cada tabla, debe definir:

- Un nombre de tabla que sea único para su Cuenta de AWS y Región de AWS.
- Una clave principal para la que cada valor debe ser único. No hay dos elementos de la tabla que puedan tener el mismo valor de clave principal.

Una clave principal puede ser simple, formada por una sola clave de partición (HASH) o compuesta, formada por una clave de partición y una clave de ordenación (RANGE).

Cada valor de clave tiene un tipo de datos asociado enumerado por la clase [ScalarAttributeType](#). El valor de clave puede ser binario (B), numérico (N) o una cadena (S). Para obtener más información, consulte [Reglas de nomenclatura y tipos de datos](#) en la Guía para desarrolladores de Amazon DynamoDB.

- Valores de rendimiento aprovisionado que definan el número de unidades de capacidad de lectura/escritura reservadas para la tabla.

Note

[Los precios de Amazon DynamoDB](#) se basan en los valores de desempeño aprovisionado que puede definir en sus tablas para que solo se reserve la capacidad que piensa que va a necesitar para la tabla.

El desempeño aprovisionado para una tabla se puede modificar en cualquier momento, por lo que puede ajustar la capacidad si cambian sus necesidades.

Crear una tabla

Use el método `CreateTable` del [cliente de DynamoDB](#) para crear una tabla de DynamoDB. Debe crear los atributos de la tabla y un esquema de tabla, que se pueden usar para identificar la clave principal de la tabla. También debe proporcionar los valores iniciales de rendimiento aprovisionados y un nombre de tabla. `CreateTable` es una operación asíncrona. `GetTableStatus` devolverá `CREATING` hasta que la tabla esté `ACTIVE` y lista para su uso.

Creación de una tabla con una clave principal simple

Este código crea una tabla con una clave principal simple ("Name").

Incluye

```
#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>
```

Código

```

//! 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);
}

```

Consulte el [ejemplo completo](#)

Creación de una tabla con una clave primaria compuesta

Añada otro objeto [AttributeDefinition](#) y [KeySchemaElement](#) a [CreateTableRequest](#).

Incluye

```

#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>

```

Código de

```

//! 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);

```

```
}
```

Consulte el [ejemplo completo](#) en GitHub.

Mostrar tablas

Puede mostrar las tablas de una región determinada llamando al método `ListTables` del [cliente de DynamoDB](#).

Incluye

```
#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>
```

Código

```
//! 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;
}
```

De forma predeterminada, se devuelven hasta 100 tablas por llamada. Utilice `GetExclusiveStartTableName` en el objeto [ListTablesOutcome](#) devuelto para obtener la última tabla que se ha evaluado. Puede utilizar este valor para iniciar el listado después del último valor devuelto de la enumeración anterior.

Consulte el [ejemplo completo](#)

Recuperación de información sobre una tabla

Puede obtener más información sobre una tabla llamando al método `DescribeTable` del [cliente de DynamoDB](#).

Incluye

```
#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();
}

```

Consulte el [ejemplo completo](#) en GitHub.

Modificación de una tabla

Puede modificar los valores de rendimiento aprovisionados de la tabla en cualquier momento llamando al método `UpdateTable` del [cliente de DynamoDB](#).

Incluye

```
#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>
```

Código de

```
//! 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);
}

```

Consulte el [ejemplo completo](#)

Eliminar una tabla

Llame al método `DeleteTable` del [cliente de DynamoDB](#) y transfírela el nombre de la tabla.

Incluye

```

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

```

Código

```

//! 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();
}
```

Consulte el [ejemplo completo](#) en GitHub.

Más información

- [Prácticas recomendadas para trabajar con tablas](#) en la Guía para desarrolladores de Amazon DynamoDB
- [Uso de tablas en DynamoDB](#) en la Guía para desarrolladores de Amazon DynamoDB

Uso de elementos en DynamoDB

En DynamoDB, un elemento es una colección de atributos, cada uno de los cuales tiene un nombre y un valor. Los valores de los atributos pueden ser escalares, conjuntos o tipos de documentos.

Para obtener más información, consulte [Reglas de nomenclatura y tipos de datos](#) en la Guía para desarrolladores de Amazon DynamoDB.

Recuperación de un elemento de una tabla

Llame al método `GetItem` del [cliente de DynamoDB](#). Transfíerale un objeto [GetItemRequest](#) con el nombre de la tabla y el valor de clave principal del elemento que desee. Este método devuelve un objeto [GetItemResult](#).

Puede utilizar el método `GetItem()` del objeto `GetItemResult` para recuperar un `Aws::Map` de los pares de clave `Aws::String` y valor [AttributeValue](#) asociados al elemento.

Incluye

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

Código de

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

Consulte el [ejemplo completo](#) en GitHub.

Adición de un nuevo elemento a una tabla

Cree pares de clave `Aws::String` y valor [AttributeValue](#) que representen cada elemento. Estos deben incluir valores para los campos de la clave principal de la tabla. Si el elemento identificado por la clave principal ya existe, la solicitud actualiza sus campos. Añádalos a [PutItemRequest](#) con el método `AddItem`.

Incluye

```
#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>
```

Código

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

Consulte el [ejemplo completo](#) en GitHub.

Actualizar un elemento existente en una tabla

Puede actualizar un atributo de un elemento que ya exista en una tabla con el método `UpdateItem` de `DynamoDBClient`, proporcionando un nombre de tabla, un valor de clave principal y campos para actualizar con su valor correspondiente.

Importaciones

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

Código de

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

Consulte el [ejemplo completo](#)

Más información

- [Prácticas recomendadas para el uso de elementos](#) en la Guía para desarrolladores de Amazon DynamoDB
- [Uso de elementos en DynamoDB](#) en la Guía para desarrolladores de Amazon DynamoDB

Ejemplos de Amazon EC2 que utilizan el AWS SDK para C++

Amazon Elastic Compute Cloud (Amazon EC2) es un servicio web que proporciona capacidad de computación de tamaño variable, literalmente, en servidores de los centros de datos de Amazon, que puede usar para crear y alojar sus sistemas de software. Puede utilizar los siguientes ejemplos para programar [Amazon EC2](#) con el AWS SDK para C++.

Note

En esta Guía solo se proporciona el código necesario para demostrar algunas técnicas, pero el [código de ejemplo completo está disponible en GitHub](#). En GitHub, puede descargar un único archivo de origen o clonar el repositorio localmente para obtener todos los ejemplos para compilarlos y ejecutarlos.

Temas

- [Administración de instancias Amazon EC2](#)
- [Uso de direcciones IP elásticas en Amazon EC2](#)
- [Uso de las regiones y las zonas de disponibilidad de Amazon EC2](#)

- [Uso de pares de claves de Amazon EC2](#)
- [Trabajar con grupos de seguridad en Amazon EC2](#)

Administración de instancias Amazon EC2

Requisitos previos

Antes de comenzar, le recomendamos que lea [Primeros pasos con el AWS SDK para C++](#).

Descargue el código de ejemplo y compile la solución tal y como se describe en [Introducción a los ejemplos de código](#).

Para ejecutar los ejemplos, el perfil de usuario que su código utilice para realizar las solicitudes debe tener los permisos adecuados en AWS (para el servicio y la acción). Para obtener más información, consulte [Proporcionar credenciales de AWS](#).

Creación de una instancia

Cree una nueva instancia de Amazon EC2 llamando a la función `RunInstances` del `EC2Client`, proporcionando un objeto [RunInstancesRequest](#) que contenga la [Imagen de máquina de Amazon \(AMI\)](#) que se va a usar y un [tipo de instancia](#).

Incluye

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

Código de

```
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;
    }
}
```

Consulte el [ejemplo completo](#)

Inicio de una instancia

Para iniciar una instancia de Amazon EC2, llame a la función `StartInstances` del `EC2Client`, proporcionando un objeto [StartInstancesRequest](#) que contenga el ID de la instancia que se va a iniciar.

Incluye

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

Código de

```
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;
    }
}

```

Consulte el [ejemplo completo](#)

Detención de una instancia

Para detener una instancia de Amazon EC2, llame a la función `StopInstances` del `EC2Client`, proporcionando un objeto [StopInstancesRequest](#) que contenga el ID de la instancia que se va a detener.

Incluye

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

Código de

```

    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;
    }
}

```

Consulte el [ejemplo completo](#)

Reiniciar una instancia

Para reiniciar una instancia de Amazon EC2, llame a la función `RebootInstances` del `EC2Client`, proporcionando un objeto [RebootInstancesRequest](#) que contenga el ID de la instancia que se va a reiniciar.

Incluye

```

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

```

Código de

```

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;
}
}
```

Consulte el [ejemplo completo](#)

Describir instancias

Para mostrar sus instancias, cree un objeto [DescribeInstancesRequest](#) y llame a la función `DescribeInstances` del `EC2Client`. Devolverá un objeto [DescribeInstancesResponse](#) que puede utilizar para mostrar las instancias de Amazon EC2 para su Cuenta de AWS y Región de AWS.

Las instancias se agrupan por reserva. Cada reserva se corresponde con la llamada a `StartInstances` que lanzó la instancia. Para mostrar sus instancias, primero debe llamar a la función `GetReservations` de la clase `DescribeInstancesResponse` y, a continuación, llamar a `getInstances` en cada objeto `Reservation` devuelto.

Incluye

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

Código

```
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;
}
}
}

```

Los resultados se pagan; puede obtener más resultados transfiriendo el valor devuelto de la función `GetNextToken` del objeto resultante a la función `SetNextToken` del objeto de la solicitud original, y después usando el mismo objeto de solicitud en la siguiente llamada a `DescribeInstances`.

Consulte el [ejemplo completo](#)

Habilitación de la monitorización de instancias

Puede monitorizar distintos aspectos de las instancias de Amazon EC2, como el uso de la CPU y la red, la memoria disponible y el espacio en disco restante. Para obtener más información sobre la monitorización de instancias, consulte [Monitoreo de Amazon EC2](#) en la Guía del usuario de Amazon EC2.

Para iniciar la monitorización de una instancia, debe crear un objeto [MonitorInstancesRequest](#) con el ID de la instancia que se va a monitorizar y transferirlo a la función `MonitorInstances` del `EC2Client`.

Incluye

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

Código

```
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;
    }
}

```

Consulte el [ejemplo completo](#)

Deshabilitación de la monitorización de instancias

Para detener la monitorización de una instancia, cree un objeto [UnmonitorInstancesRequest](#) con el ID de la instancia cuya monitorización se va a detener y transfíralo a la función `UnmonitorInstances` del `EC2Client`.

Incluye

```

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

```

Código

```

    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;
}
}
```

Consulte el [ejemplo completo](#)

Más información

- [RunInstances](#) en la Referencia de la API de Amazon EC2
- [DescribeInstances](#) en la Referencia de la API de Amazon EC2
- [StartInstances](#) en la Referencia de la API de Amazon EC2
- [StopInstances](#) en la Referencia de la API de Amazon EC2
- [RebootInstances](#) en la Referencia de la API de Amazon EC2
- [DescribeInstances](#) en la Referencia de la API de Amazon EC2
- [MonitorInstances](#) en la Referencia de la API de Amazon EC2
- [UnmonitorInstances](#) en la Referencia de la API de Amazon EC2

Uso de direcciones IP elásticas en Amazon EC2

Requisitos previos

Antes de comenzar, le recomendamos que lea [Primeros pasos con el AWS SDK para C++](#).

Descargue el código de ejemplo y compile la solución tal y como se describe en [Introducción a los ejemplos de código](#).

Para ejecutar los ejemplos, el perfil de usuario que su código utilice para realizar las solicitudes debe tener los permisos adecuados en AWS (para el servicio y la acción). Para obtener más información, consulte [Proporcionar credenciales de AWS](#).

Asignación de una dirección IP elástica

Para utilizar una dirección IP elástica, primero asigne una a su cuenta y, a continuación, asóciela a su instancia o a una interfaz de red.

Para asignar una dirección IP elástica, llame a la función `AllocateAddress` del `EC2Client` con un objeto [AllocateAddressRequest](#) que contenga el tipo de red (EC2 clásico o VPC).

Warning

Vamos a retirar EC2-Classic el 15 de agosto de 2022. Le recomendamos que migre de EC2-Classic a una VPC. Para obtener más información, consulte [Migración de EC2-Classic a una VPC](#) en la [Guía del usuario de Amazon EC2 para instancias de Linux](#) o en la [Guía del usuario de Amazon EC2 para instancias de Windows](#). Consulte también la entrada de blog [EC2-Classic Networking is Retiring – Here's How to Prepare](#) (Se va a retirar la red EC2-Classic: cómo prepararse).

La clase [AllocateAddressResponse](#) devuelta en el objeto de respuesta contiene un ID de asignación que puede utilizar para asociar la dirección a una instancia, transfiriendo el ID de asignación y el ID de instancia en un objeto [AssociateAddressRequest](#) a la función `AssociateAddress` del `EC2Client`.

Incluye

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

Código de

```
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;

```

Consulte el [ejemplo completo](#)

Describir direcciones IP elásticas

Para listar las direcciones IP elásticas asignadas a su cuenta, llame a la función `DescribeAddresses` del `EC2Client`. Devuelve como resultado un objeto que contiene [DescribeAddressesResponse](#) que puede utilizar para obtener una lista de objetos [Address](#) que representen las direcciones IP elásticas de su cuenta.

Incluye

```

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

```

```
#include <iostream>
```

Código

```
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;
}
```

Consulte el [ejemplo completo](#)

Liberación de una dirección IP elástica

Para liberar una dirección IP elástica, llame a la función `ReleaseAddress` del `EC2Client`; transfiriéndole un objeto [ReleaseAddressRequest](#) que contenga el ID de asignación de la dirección IP elástica que quiere liberar.

Incluye

```
#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;
}
```

Tras liberar una dirección IP elástica, esta se libera del grupo de direcciones IP de AWS y puede que no esté disponible para usarla más adelante. Asegúrese de actualizar sus registros DNS y los servidores o dispositivos que se comunican con la dirección. Si intenta liberar una dirección IP elástica que ya ha liberado, obtendrá un error `AuthFailure` si la dirección ya se ha asignado a otra cuenta de AWS.

Si utiliza una VPC predeterminada, al liberar una dirección IP elástica, esta se desvincula automáticamente de cualquier instancia a la que esté asociada. Para desvincular una dirección IP elástica sin liberarla, utilice la función `DisassociateAddress` del `EC2Client`.

Si utiliza una VPC distinta de la predeterminada, debe usar `DisassociateAddress` para desvincular la dirección IP elástica antes de intentar liberarla. De lo contrario, Amazon EC2; devuelve un error (`InvalidIPAddress.InUse`).

Consulte el [ejemplo completo](#)

Más información

- [Direcciones IP elásticas](#) en la Guía del usuario de Amazon EC2
- [AllocateAddress](#) en la Guía del usuario de Amazon EC2
- [DescribeAddresses](#) en la Guía del usuario de Amazon EC2

- [ReleaseAddress](#) en la Guía del usuario de Amazon EC2

Uso de las regiones y las zonas de disponibilidad de Amazon EC2

Requisitos previos

Antes de comenzar, le recomendamos que lea [Primeros pasos con el AWS SDK para C++](#).

Descargue el código de ejemplo y compile la solución tal y como se describe en [Introducción a los ejemplos de código](#).

Para ejecutar los ejemplos, el perfil de usuario que su código utilice para realizar las solicitudes debe tener los permisos adecuados en AWS (para el servicio y la acción). Para obtener más información, consulte [Proporcionar credenciales de AWS](#).

Descripción de regiones

[Para ver las Regiones de AWS disponibles para su Cuenta de AWS, llame a la función DescribeRegions del EC2Client con un objeto DescribeRegionsRequest.](#)

Recibirá el objeto de resultado [DescribeRegionsResponse](#). Llame a su función `GetRegions` para obtener una lista de objetos [Region](#) que representen a cada región.

Incluye

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

Código de

```
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;
}

```

Consulte el [ejemplo completo](#)

Descripción de zonas de disponibilidad

Para mostrar las zonas de disponibilidad disponibles para su cuenta, llame a la función `DescribeAvailabilityZones` del `EC2Client` con un objeto [DescribeAvailabilityZonesRequest](#).

Recibirá el objeto de resultado [DescribeAvailabilityZonesResponse](#) en el resultado. Llame a su función `GetAvailabilityZones` para obtener una lista de objetos [AvailabilityZone](#) que representen cada zona de disponibilidad.

Incluye

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

Código

```

    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;
}
}
```

Consulte el [ejemplo completo](#)

Más información

- [Regiones y zonas de disponibilidad](#) en la Guía del usuario de Amazon EC2
- [DescribeRegions](#) en la referencia de la API de Amazon EC2
- [DescribeAvailabilityZones](#) en la referencia de la API de Amazon EC2

Uso de pares de claves de Amazon EC2

Requisitos previos

Antes de comenzar, le recomendamos que lea [Primeros pasos con el AWS SDK para C++](#).

Descargue el código de ejemplo y compile la solución tal y como se describe en [Introducción a los ejemplos de código](#).

Para ejecutar los ejemplos, el perfil de usuario que su código utilice para realizar las solicitudes debe tener los permisos adecuados en AWS (para el servicio y la acción). Para obtener más información, consulte [Proporcionar credenciales de AWS](#).

Creación de un par de claves

Para crear un par de claves, llame a la función `CreateKeyPair` del `EC2Client` con un objeto [CreateKeyPairRequest](#) que contenga el nombre de la clave.

Incluye

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

```
#include <iostream>
#include <fstream>
```

Código de

```
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;
    }
}
}
```

Consulte el [ejemplo completo](#)

Describir pares de claves

Para mostrar sus pares de claves o para obtener información sobre ellas, llame a la función `DescribeKeyPairs` del `EC2Client` con un objeto [describeKeyPairs](#).

Este método devuelve [DescribeKeyPairsResult](#), que puede utilizar para obtener acceso a la lista de pares de claves llamando a su función `GetKeyPairs`, que devuelve una lista de objetos [KeyPairInfo](#).

Incluye

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

Código de

```
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;
}
```

Consulte el [ejemplo completo](#)

Eliminar un par de claves

Para eliminar un par de claves, llame a la función `DeleteKeyPair` de `EC2Client`, transfiriéndole un objeto [DeleteKeyPairRequest](#) que contenga el nombre del par de claves que desea eliminar.

Incluye

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

Código de

```
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;
}
```

Consulte el [ejemplo completo](#)

Más información

- [Pares de claves de Amazon EC2](#) en la Guía del usuario de Amazon EC2
- [CreateKeyPairs](#) en la Referencia de la API de Amazon EC2
- [DescribeKeyPairs](#) en la Referencia de la API de Amazon EC2
- [DeleteKeyPair](#) en la Referencia de la API de Amazon EC2

Trabajar con grupos de seguridad en Amazon EC2

Requisitos previos

Antes de comenzar, le recomendamos que lea [Primeros pasos con el AWS SDK para C++](#).

Descargue el código de ejemplo y compile la solución tal y como se describe en [Introducción a los ejemplos de código](#).

Para ejecutar los ejemplos, el perfil de usuario que su código utilice para realizar las solicitudes debe tener los permisos adecuados en AWS (para el servicio y la acción). Para obtener más información, consulte [Proporcionar credenciales de AWS](#).

Creación de un grupo de seguridad

Para crear un grupo de seguridad, llame a la función `CreateSecurityGroup` de `EC2Client` con un objeto [CreateSecurityGroupRequest](#) que contenga el nombre de la clave.

Incluye

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

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

Código de

```
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;
```

Consulte el [ejemplo completo](#)

Configuración de un grupo de seguridad

Un grupo de seguridad puede controlar el tráfico de entrada y salida a sus instancias de Amazon EC2.

Para añadir reglas de entrada al grupo de seguridad, utilice la función `AuthorizeSecurityGroupIngress` de `EC2Client`, proporcionando el nombre del grupo de seguridad y las reglas de acceso ([IpPermission](#)) que desea asignar al grupo dentro de un objeto [AuthorizeSecurityGroupIngressRequest](#). El siguiente ejemplo muestra cómo añadir permisos de IP a un grupo de seguridad.

Incluye

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

Código de

```

    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;
    }

```

Para añadir una regla de salida al grupo de seguridad, proporcione datos similares en un objeto [AuthorizeSecurityGroupEgressRequest](#) a la función `AuthorizeSecurityGroupEgress` del `EC2Client`.

Consulte el [ejemplo completo](#)

Describir grupos de seguridad

Para describir los grupos de seguridad o para obtener información sobre ellos, llame a la función `DescribeSecurityGroups` del `EC2Client` con un objeto [DescribeSecurityGroupsRequest](#).

Recibirá un objeto [DescribeSecurityGroupsResponse](#) en el resultado, que puede usar para obtener acceso a la lista de grupos de seguridad llamando a su función `GetSecurityGroups`, que devuelve una lista de objetos [SecurityGroup](#).

Incluye

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

Código

```
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());
```

Consulte el [ejemplo completo](#)

Eliminación de un grupo de seguridad

Para eliminar un grupo de seguridad, llame a la `DeleteSecurityGroup` del `EC2Client` transfiriéndole un objeto [DeleteSecurityGroupRequest](#) que contenga el ID del grupo de seguridad que desea eliminar.

Incluye

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

Código

```
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;
}
```

Consulte el [ejemplo completo](#)

Más información

- [Grupos de seguridad de Amazon EC2](#) en la Guía del usuario de Amazon EC2
- [Autorización del tráfico de entrada para sus instancias de Linux](#) en la Guía del usuario de Amazon EC2
- [CreateSecurityGroup](#) en la Referencia de la API de Amazon EC2
- [DescribeSecurityGroups](#) en la Referencia de la API de Amazon EC2
- [DeleteSecurityGroup](#) en la Referencia de la API de Amazon EC2
- [AuthorizeSecurityGroupIngress](#) en la Referencia de la API de Amazon EC2

Ejemplos de código de Amazon S3 con el AWS SDK para C++

[Amazon S3](#) es un servicio de almacenamiento de objetos creado para almacenar y recuperar cualquier volumen de datos desde cualquier ubicación. El AWS SDK para C++ proporciona varias clases para interactuar con Amazon S3.

Note

En esta Guía solo se proporciona el código necesario para demostrar algunas técnicas, pero el [código de ejemplo completo está disponible en GitHub](#). En GitHub, puede descargar un único archivo de origen o clonar el repositorio localmente para obtener todos los ejemplos para compilarlos y ejecutarlos.

- [S3Client](#) Clase

La biblioteca `S3Client` es una interfaz de Amazon S3 con todas las características.

El ejemplo `list_buckets_disabling_dns_cache.cpp` está diseñado específicamente para funcionar con CURL en Linux/Mac (aunque se puede modificar para que funcione en Windows). Si utiliza Windows, elimine el archivo `list_buckets_disabling_dns_cache.cpp` antes de compilar el proyecto porque se basa en el `HttpClient curl` de Linux.

El código de ejemplo que utiliza `S3Client` está en la [carpeta s3](#) de Github. Consulte el archivo [Readme](#) en Github para ver una lista completa de las funciones que se muestran en este ejemplo.

Algunas partes de los ejemplos de `s3` se tratan con más detalle en esta guía:

- [Creación, listado y eliminación de buckets](#)
- [Operaciones en objetos](#): carga y descarga de objetos de datos
- [Administración de permisos de acceso de Amazon S3](#)
- [Administración del acceso a los buckets de Amazon S3 mediante políticas de buckets](#)
- [Configuración de un bucket de Amazon S3 como un sitio web](#)
- [S3CrtClient](#) Clase

`S3CrtClient` se ha agregado en la versión 1.9 del SDK. `S3CrtClient` proporciona un rendimiento alto para las operaciones GET (descarga) y PUT (carga) de Amazon S3. `S3CrtClient` se implementa sobre las bibliotecas Common Runtime (CRT) de AWS.

El código de ejemplo que utiliza la clase `S3CrtClient` está en la [carpeta s3-crt](#) de Github. Consulte el archivo [Readme](#) en Github para ver una lista completa de las funciones que se muestran en este ejemplo.

- [Uso de S3CrtClient para operaciones de Amazon S3](#)
- [TransferManager](#) Clase

`TransferManager` es un servicio totalmente administrado que permite transferir archivos a través del Protocolo de File Transfer (FTP), el Protocolo de File Transfer a través de SSL (FTPS) o el Protocolo de File Transfer (SFTP) de Secure Shell (SSH) directamente hacia o desde Amazon S3.

El código de ejemplo que utiliza `TransferManager` está en la [carpeta transfer-manager](#) de Github. Consulte el archivo [Readme](#) en Github para ver una lista completa de las funciones que se muestran en este ejemplo.

- [Uso de TransferManager para operaciones de Amazon S3](#)

Creación, listado y eliminación de buckets

Todos los objetos o archivos de Amazon Simple Storage Service (Amazon S3) están contenidos en un bucket, que representa una carpeta de objetos. Cada bucket tiene un nombre único a nivel mundial dentro de AWS. Para obtener más información, consulte [Uso de buckets de Amazon S3](#) en la Guía del usuario de Amazon Simple Storage Service.

Requisitos previos

Antes de comenzar, le recomendamos que lea [Primeros pasos con el AWS SDK para C++](#).

Descargue el código de ejemplo y compile la solución tal y como se describe en [Introducción a los ejemplos de código](#).

Para ejecutar los ejemplos, el perfil de usuario que su código utilice para realizar las solicitudes debe tener los permisos adecuados en AWS (para el servicio y la acción). Para obtener más información, consulte [Proporcionar credenciales de AWS](#).

Obtener una lista de buckets

Para ejecutar el ejemplo `list_buckets`, en el símbolo del sistema, vaya hasta la carpeta en la que el sistema de compilación crea los ejecutables de compilación. Ejecute el ejecutable con un nombre similar a `run_list_buckets` (el nombre completo del archivo ejecutable variará en función de su sistema operativo). El resultado muestra una lista de los buckets de su cuenta, si tiene alguno, o una lista vacía si no tienes ningún bucket.

En `list_buckets.cpp`, hay dos métodos:

- `main()` llama a `ListBuckets()`.
- `ListBuckets()` usa el SDK para consultar sus depósitos.

El objeto `S3Client` llama al método `ListBuckets()` del SDK. Si se ejecuta correctamente, el método devuelve un objeto `ListBucketOutcome`, que contiene un objeto `ListBucketResult`. El objeto `ListBucketResult` llama al método `GetBuckets()` para obtener una lista de objetos `Bucket` que contienen información sobre cada bucket de Amazon S3 de su cuenta.

Código de

```
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;
}
```

Consulte el [ejemplo de list_buckets](#) completo en Github.

Crear un bucket

Para ejecutar el ejemplo `create_bucket`, en el símbolo del sistema, vaya hasta la carpeta en la que el sistema de compilación crea los ejecutables de compilación. Ejecute el ejecutable con un nombre similar a `run_create_bucket` (el nombre completo del archivo ejecutable variará en función de su sistema operativo). El código crea un depósito vacío en su cuenta y, a continuación, muestra si la solicitud se ha realizado correctamente o no.

En `create_bucket.cpp`, hay dos métodos:

- `main()` llama a `CreateBucket()`. En `main()`, tiene que cambiar Región de AWS por la región de su cuenta con `enum`. Para ver la región de su cuenta, inicie sesión en la [Consola de administración de AWS](#) y localice la región en la esquina superior derecha.
- `CreateBucket()` utiliza el SDK para crear un bucket.

El objeto `S3Client` llama al método `CreateBucket()` del SDK, transfiriéndole un objeto `CreateBucketRequest` con el nombre del bucket. De forma predeterminada, los buckets se crean en la región `us-east-1` (Norte de Virginia). Si su región no es `us-east-1`, el código establece una restricción de bucket para garantizar que el bucket se cree en su región.

Código de

```
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();
}
```

Consulte el [ejemplo de create_buckets](#) completo en Github.

Eliminar un bucket

Para ejecutar el ejemplo `delete_bucket`, en el símbolo del sistema, vaya hasta la carpeta en la que el sistema de compilación crea los ejecutables de compilación. Ejecute el ejecutable con un nombre similar a `run_delete_bucket` (el nombre completo del archivo ejecutable variará en función de su sistema operativo). El código elimina el bucket especificado de su cuenta y, a continuación, muestra si la solicitud se ha realizado correctamente o no.

En `delete_bucket.cpp`, hay dos métodos.

- `main()` llama a `DeleteBucket()`. En `main()`, tiene que cambiar Región de AWS por la región de su cuenta con `enum`. También debe cambiar `bucket_name` por el nombre del bucket que desea eliminar.
- `DeleteBucket()` usa el SDK para eliminar el bucket.

El objeto `S3Client` usa el método `DeleteBucket()` del SDK, transfiriéndole un objeto `DeleteBucketRequest` con el nombre del bucket que se va a eliminar. El bucket debe estar vacío para que funcione correctamente.

Código

```
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();
}
```

Consulte el [ejemplo de delete_bucket](#) completo en Github.

Operaciones en objetos

Un objeto de Amazon S3 representa un archivo o un conjunto de datos. Cada objeto debe residir en un [bucket](#).

Requisitos previos

Antes de comenzar, le recomendamos que lea [Primeros pasos con el AWS SDK para C++](#).

Descargue el código de ejemplo y compile la solución tal y como se describe en [Introducción a los ejemplos de código](#).

Para ejecutar los ejemplos, el perfil de usuario que su código utilice para realizar las solicitudes debe tener los permisos adecuados en AWS (para el servicio y la acción). Para obtener más información, consulte [Proporcionar credenciales de AWS](#).

Para cargar un archivo en un bucket

Utilice la función `PutObject` del objeto `S3Client` y proporcione un nombre de bucket, un nombre de clave y el archivo que se va a cargar. `Aws::FStream` se utiliza para cargar el contenido del archivo local en el bucket. El bucket debe existir o se producirá un error.

Para ver un ejemplo sobre cómo cargar objetos de forma asíncrona, consulte [Programación asíncrona con el AWS SDK para C++](#).

Código de

```
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();
}
```

Consulte el ejemplo completo en [Github](#).

Carga de una cadena en un bucket

Utilice la función `PutObject` del objeto `S3Client` y proporcione un nombre de bucket, un nombre de clave y el archivo que se va a cargar. El bucket debe existir o se producirá un error. Este ejemplo se diferencia del anterior porque se utiliza `Aws::StringStream` para cargar un objeto de datos de cadena en memoria directamente en un bucket.

Para ver un ejemplo sobre cómo cargar objetos de forma asíncrona, consulte [Programación asíncrona con el AWS SDK para C++](#).

Código de

```
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();
}
```

Consulte el ejemplo completo en [Github](#).

List objects

Para obtener una lista de objetos dentro de un bucket, utilice la función `ListObjects` del objeto `S3Client`. Proporcione una solicitud `ListObjectsRequest` configurada con el nombre de un bucket para enumerar su contenido.

La función `ListObjects` devuelve un objeto `ListObjectsOutcome` que puede utilizar para obtener una lista de objetos en forma de instancias de `Object`.

Código de

```
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);
        }
    } while (true);
}
```

```
    }

    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;
}
```

Consulte el ejemplo completo en [Github](#).

Descargar un objeto

Utilice la función `GetObject` del objeto `S3Client`, transfiriéndole una solicitud `GetObjectRequest` que debe configurar con el nombre de un bucket y la clave del objeto para descargar. `GetObject` devuelve un objeto [GetObjectOutcome](#) que consta de un objeto [GetObjectResult](#) y un [S3Error](#). `GetObjectResult` se puede usar para acceder a los datos del objeto S3.

En el siguiente ejemplo se descarga un objeto de Amazon S3. El contenido del objeto se almacena en una variable local y la primera línea del contenido se envía a la consola.

Código de

```
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();
}

```

Consulte el ejemplo completo en [Github](#).

Elimine un objeto

Utilice la función `DeleteObject` del objeto `S3Client`, transfiriéndole una solicitud `DeleteObjectRequest` que debe configurar con el nombre de un bucket y el objeto para descargar. El bucket y la clave de objeto especificados deben existir o se producirá un error.

Código

```

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();
}
```

Consulte el ejemplo completo en [Github](#).

Administración de permisos de acceso de Amazon S3

Los permisos de acceso para un bucket o un objeto de Amazon S3 se definen en una lista de control de acceso (ACL). La ACL especifica el propietario del bucket o del objeto y una lista de concesiones. Cada concesión especifica un usuario (o beneficiario) y los permisos del usuario para acceder al bucket o al objeto; por ejemplo, acceso de lectura (READ) o escritura (WRITE).

Requisitos previos

Antes de comenzar, le recomendamos que lea [Primeros pasos con el AWS SDK para C++](#).

Descargue el código de ejemplo y compile la solución tal y como se describe en [Introducción a los ejemplos de código](#).

Para ejecutar los ejemplos, el perfil de usuario que su código utilice para realizar las solicitudes debe tener los permisos adecuados en AWS (para el servicio y la acción). Para obtener más información, consulte [Proporcionar credenciales de AWS](#).

Administración de la lista de control de acceso de un objeto

La lista de control de acceso de un objeto se puede recuperar llamando al método `GetObjectAcl` de `S3Client`. El método acepta los nombres del objeto y su bucket. El valor devuelto incluye el `Owner` de las ACL y una lista de `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";
    }
}

```

La ACL se puede modificar creando una nueva o cambiando las concesiones especificadas en la actual. La ACL actualizada se convierte en la nueva ACL actual al transferirla al método `PutObjectAcl`.

El siguiente código usa la ACL recuperada por `GetObjectAcl` y le agrega una nueva concesión. El usuario o beneficiario recibe el permiso `READ` para el objeto. La ACL modificada se pasa a `PutObjectAcl` y, de este modo, se convierte en la nueva ACL actual.

```

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;
}
```

Consulte el ejemplo completo en [Github](#).

Administración de la lista de control de acceso de un bucket

En la mayoría de los casos, el método preferido para configurar los permisos de acceso de un bucket es definir una política de bucket. Sin embargo, los buckets también admiten listas de control de acceso para los usuarios que deseen utilizarlas.

La administración de una lista de control de acceso para un bucket es idéntica a la que se usa para un objeto. El método `GetBucketAcl` recupera la ACL actual de un bucket y `PutBucketAcl` aplica una nueva ACL al bucket.

El siguiente código muestra cómo obtener y configurar la ACL de un bucket.

```

//! 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;
}
```

Consulte el ejemplo completo en [Github](#).

Administración del acceso a los buckets de Amazon S3 mediante políticas de buckets

Puede definir, obtener o eliminar una política de bucket para administrar el acceso a sus buckets de Amazon S3.

Requisitos previos

Antes de comenzar, le recomendamos que lea [Primeros pasos con el AWS SDK para C++](#).

Descargue el código de ejemplo y compile la solución tal y como se describe en [Introducción a los ejemplos de código](#).

Para ejecutar los ejemplos, el perfil de usuario que su código utilice para realizar las solicitudes debe tener los permisos adecuados en AWS (para el servicio y la acción). Para obtener más información, consulte [Proporcionar credenciales de AWS](#).

Definir una política de bucket

Para establecer la política de bucket para un bucket de S3 concreto, llame a la función `PutBucketPolicy` de `S3Client` y proporcione el nombre del bucket y la representación en JSON de la política en [PutBucketPolicyRequest](#).

Código de

```

//! 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

Puede utilizar la clase de utilidad [Aws::Utils::Json::JsonValue](#) para que le ayude a construir objetos JSON válidos para transferirlos a PutBucketPolicy.

Consulte el ejemplo completo en [Github](#).

Obtener una política de bucket

Para recuperar la política de un bucket de Amazon S3, llame a la función `GetBucketPolicy` del `S3Client`, transfiriéndole el nombre del bucket en un objeto [GetBucketPolicyRequest](#).

Código de

```
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();
}
```

Consulte el ejemplo completo en [Github](#).

Eliminar una política de bucket

Para eliminar una política de bucket, llame a la función `DeleteBucketPolicy` del `S3Client`, proporcionando el nombre de bucket en [deleteBucketPolicyRequest](#).

Código

```
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();
}
```

Esta función se ejecuta correctamente aunque el bucket no tenga una política todavía. Si especifica el nombre de un bucket que no existe o si no tiene acceso al bucket, se produce la excepción `AmazonServiceException`.

Consulte el ejemplo completo en [Github](#).

Más información

- [PutBucketPolicy](#) en la Referencia de la API de Amazon Simple Storage Service
- [GetBucketPolicy](#) en la Referencia de la API de Amazon Simple Storage Service
- [DeleteBucketPolicy](#) en la Referencia de la API de Amazon Simple Storage Service
- [Información general del lenguaje de la política de acceso](#) en la Guía del usuario de Amazon Simple Storage Service
- [Ejemplos de política de bucket](#) en la Guía del usuario de Amazon Simple Storage Service

Configuración de un bucket de Amazon S3 como un sitio web

Puede configurar un bucket de Amazon S3 para que se comporte como un sitio web. Para ello, debe establecer la configuración de su sitio web.

Requisitos previos

Antes de comenzar, le recomendamos que lea [Primeros pasos con el AWS SDK para C++](#).

Descargue el código de ejemplo y compile la solución tal y como se describe en [Introducción a los ejemplos de código](#).

Para ejecutar los ejemplos, el perfil de usuario que su código utilice para realizar las solicitudes debe tener los permisos adecuados en AWS (para el servicio y la acción). Para obtener más información, consulte [Proporcionar credenciales de AWS](#).

Establecimiento de la configuración de sitio web de un bucket

[Para establecer la configuración del sitio web de un bucket de Amazon S3, llame a la función `PutBucketWebsite` del `S3Client` con un objeto `PutBucketWebsiteRequest` que contenga el nombre del bucket y la configuración de su sitio web, proporcionados en un objeto `WebsiteConfiguration`.](#)

Es obligatorio establecer un documento de índice; todos los demás parámetros son opcionales.

Código de

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

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

    Aws::S3::Model::ErrorDocument errorDocument;
    errorDocument.SetKey(errorPage);

    Aws::S3::Model::WebsiteConfiguration websiteConfiguration;
    websiteConfiguration.SetIndexDocument(indexDocument);
    websiteConfiguration.SetErrorDocument(errorDocument);
```

```
Aws::S3::Model::PutBucketWebsiteRequest request;
request.SetBucket(bucketName);
request.SetWebsiteConfiguration(websiteConfiguration);

Aws::S3::Model::PutBucketWebsiteOutcome outcome =
    client.PutBucketWebsite(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Error: PutBucketWebsite: "
              << outcome.GetError().GetMessage() << std::endl;
} else {
    std::cout << "Success: Set website configuration for bucket '"
              << bucketName << "'." << std::endl;
}

return outcome.IsSuccess();
}
```

Note

El establecimiento de la configuración de sitio web no modifica los permisos de acceso del bucket. Para que los archivos estén visibles en la web, también deberá definir una política de bucket que permita el acceso de lectura pública a los archivos del bucket. Para obtener más información, consulte [Administración del acceso a buckets de Amazon S3 mediante políticas de buckets](#).

Consulte el ejemplo completo en [Github](#).

Obtener la configuración de sitio web de un bucket

Para obtener la configuración de sitio web de un bucket de Amazon S3, llame a la función `GetBucketWebsite` del `S3Client` con un objeto [GetBucketWebsiteRequest](#) que contenga el nombre del bucket para el que desea recuperar la configuración.

La configuración se devolverá como un objeto [getBucketWebsiteResult](#) dentro del objeto de resultado. Si no hay ninguna configuración de sitio web para el bucket, se devolverá `null`.

Código

```
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();
}

```

Consulte el ejemplo completo en [Github](#).

Eliminar la configuración de sitio web de un bucket

Para eliminar la configuración de sitio web de un bucket de Amazon S3, llame a la función `DeleteBucketWebsite` del `S3Client` con un objeto `DeleteBucketWebsiteRequest` que contenga el nombre del bucket del que se va a eliminar la configuración.

Código de

```
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();
}
```

Consulte el ejemplo completo en [Github](#).

Más información

- [PUT Bucket website](#) en la Referencia de la API de Amazon Simple Storage Service
- [GET Bucket website](#) en la Referencia de la API de Amazon Simple Storage Service
- [DELETE Bucket website](#) en la Referencia de la API de Amazon Simple Storage Service

Uso de TransferManager para operaciones de Amazon S3

Puede utilizar la clase AWS SDK para C++ del `TransferManager` para transferir archivos de forma fiable desde el entorno local a Amazon S3 y copiar objetos de una ubicación de S3 a otra. `TransferManager` puede obtener el progreso de la transferencia y detener o reanudar las cargas y descargas.

Note

Para evitar que se le cobre por cargas incompletas o parciales, le recomendamos que active la regla de ciclo de vida [AbortIncompleteMultipartUpload](#) en sus buckets de Amazon S3.

Esta regla le indica a Amazon S3 que anule las cargas multiparte que no se completen en un número especificado de días después de iniciarse. Cuando se supera el plazo establecido, Amazon S3 anula la carga y, a continuación, elimina los datos de la carga incompleta. Para obtener más información, consulte [Configuración del ciclo de vida de un bucket](#) en la Guía del usuario de Amazon S3.

Requisitos previos

Antes de comenzar, le recomendamos que lea [Primeros pasos con el AWS SDK para C++](#).

Descargue el código de ejemplo y compile la solución tal y como se describe en [Introducción a los ejemplos de código](#).

Para ejecutar los ejemplos, el perfil de usuario que su código utilice para realizar las solicitudes debe tener los permisos adecuados en AWS (para el servicio y la acción). Para obtener más información, consulte [Proporcionar credenciales de AWS](#).

Carga y descarga de un objeto mediante **TransferManager**

En este ejemplo, se muestra cómo [TransferManager](#) transfiere un objeto grande en la memoria. Se llama a los métodos `UploadFile` y `DownloadFile` de forma asíncrona que devuelven un `TransferHandle` para gestionar el estado de la solicitud. Si el objeto cargado es superior al `bufferSize`, se realiza una carga multiparte. El valor predeterminado del `bufferSize` es de 5 MB, pero se puede configurar mediante [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;
    }

```

Consulte el ejemplo completo en [Github](#).

Uso de **S3CrtClient** para operaciones de Amazon S3

La clase `S3CrtClient` está disponible en la versión 1.9 del AWS SDK para C++ y mejora el rendimiento de la carga y descarga de archivos de datos de gran tamaño desde y hacia Amazon S3. Para obtener más información sobre las mejoras de esta versión, consulte [Mejora del rendimiento de Amazon S3 con AWS SDK para C++ v1.9](#).

El `S3CrtClient` se implementa sobre las [bibliotecas Common Runtime \(CRT\) de AWS](#).

Note

Para evitar que se le cobre por cargas incompletas o parciales, le recomendamos que active la regla de ciclo de vida [AbortIncompleteMultipartUpload](#) en sus buckets de Amazon S3. Esta regla le indica a Amazon S3 que anule las cargas multiparte que no se completen en un número especificado de días después de iniciarse. Cuando se supera el plazo establecido, Amazon S3 anula la carga y, a continuación, elimina los datos de la carga incompleta. Para obtener más información, consulte [Configuración del ciclo de vida de un bucket](#) en la Guía del usuario de Amazon S3.

Requisitos previos

Antes de comenzar, le recomendamos que lea [Primeros pasos con el AWS SDK para C++](#).

Descargue el código de ejemplo y compile la solución tal y como se describe en [Introducción a los ejemplos de código](#).

Para ejecutar los ejemplos, el perfil de usuario que su código utilice para realizar las solicitudes debe tener los permisos adecuados en AWS (para el servicio y la acción). Para obtener más información, consulte [Proporcionar credenciales de AWS](#).

Carga y descarga de un objeto mediante `S3CrtClient`

En este ejemplo se muestra cómo usar [S3CrtClient](#). En el ejemplo se crea un bucket, se carga un objeto, se descarga el objeto y, a continuación, se eliminan el archivo y el bucket. Una operación PUT se convierte en una carga multiparte. Una operación GET se convierte en varias solicitudes GET "ordenadas". Para obtener más información sobre cargas multiparte, consulte [Carga y copia de objetos con carga multiparte](#) en la Guía del usuario de Amazon S3.

En este ejemplo, el archivo de datos proporcionado, `ny.json`, se carga como una carga multiparte. Puede confirmarlo consultando los registros de depuración después de ejecutar correctamente el programa.

Si se produce un error en la carga, `AbortMultipartUpload` se emite en una en la biblioteca CRT subyacente para limpiar las partes ya cargadas. Sin embargo, no todos los fallos se pueden solucionar internamente (por ejemplo, si se desconecta un cable de red). Le recomendamos crear una regla de ciclo de vida en su bucket de Amazon S3 para garantizar que los datos cargados

parcialmente no permanezcan en su cuenta (los datos cargados parcialmente siguen siendo facturables). Para obtener información sobre cómo configurar una regla de ciclo de vida, consulte [Descubrimiento y eliminación de cargas multiparte incompletas para reducir los costos de Amazon S3](#)

Uso del registro de depuración para explorar los detalles de las cargas multiparte

1. En `main()`, tenga en cuenta que hay comentarios "TODO" con instrucciones para actualizar el código.
 - a. Para `file_name`: desde el enlace proporcionado en el comentario del código, descargue un archivo de datos de muestra `ny.json` o utilice un archivo de datos de gran tamaño que sea suyo.
 - b. Para `region`: actualice la variable `region`, mediante la enumeración, de la Región de AWS de su cuenta. Para encontrar la región de su cuenta, inicie sesión en la Consola de administración de AWS y localice la región en la esquina superior derecha.
2. Compile el ejemplo.
3. Copie el archivo especificado en la variable `file_name` en su carpeta de ejecutables y ejecute el archivo ejecutable `s3-crt-demo`.
4. En la carpeta de ejecutables, busque el archivo `.log` más reciente.
5. Abra el archivo de registro, seleccione buscar y escriba **partNumber**.
6. El registro contiene entradas similares a las siguientes, donde se especifican `uploadId` y `partNumber` para cada parte del archivo cargado:

```
PUT /my-object
partNumber=1&uploadId=gsk8vDbmn1A5EseDo._LDEgq22Qmt0SeuszYxMsZ9ABt503VqDIFOP8
content-length:8388608 host:my-bucketasdfasdf.s3.us-
east-2.amazonaws.com x-amz-content-sha256:UNSIGNED-PAYLOAD

y
```

```
PUT /my-object
partNumber=2&uploadId=gsk8vDbmn1A5EseDo._LDEgq22Qmt0SeuszYxMsZ9ABt503VqDIFOP8
content-length:8388608 host:my-bucketasdfasdf.s3.us-
east-2.amazonaws.com x-amz-content-sha256:UNSIGNED-PAYLOAD
```

Consulte el ejemplo completo en [Github](#).

Ejemplos de código de Amazon SQS mediante el AWS SDK para C++

Amazon Simple Queue Service (Amazon SQS) es un servicio de colas de mensajes completamente administrado que facilita el desacople y el escalado de microservicios, sistemas distribuidos y aplicaciones sin servidor. Puede utilizar los siguientes ejemplos para programar [Amazon SQS](#) con el AWS SDK para C++.

Note

En esta Guía solo se proporciona el código necesario para demostrar algunas técnicas, pero el [código de ejemplo completo está disponible en GitHub](#). En GitHub, puede descargar un único archivo de origen o clonar el repositorio localmente para obtener todos los ejemplos para compilarlos y ejecutarlos.

Temas

- [Trabajo con colas de mensajes de Amazon SQS](#)
- [Envío, recepción y eliminación de mensajes de Amazon SQS](#)
- [Habilitar sondeos largos para las colas de mensajes de Amazon SQS](#)
- [Configuración del tiempo de espera de visibilidad en Amazon SQS](#)
- [Uso de colas de mensajes fallidos en Amazon SQS](#)

Trabajo con colas de mensajes de Amazon SQS

Una cola de mensajes es el contenedor lógico utilizado para enviar mensajes de forma fiable en Amazon SQS. Existen dos tipos de colas: estándar y primero en entrar, primero en salir (FIFO). Para obtener más información sobre las colas y las diferencias entre estos tipos, consulte la [Guía para desarrolladores de Amazon Simple Queue Service](#).

Estos ejemplos de C++ muestran cómo utilizar el AWS SDK para C++ para crear, enumerar, eliminar y obtener la URL de una cola de Amazon SQS.

Requisitos previos

Antes de comenzar, le recomendamos que lea [Primeros pasos con el AWS SDK para C++](#).

Descargue el código de ejemplo y compile la solución tal y como se describe en [Introducción a los ejemplos de código](#).

Para ejecutar los ejemplos, el perfil de usuario que su código utilice para realizar las solicitudes debe tener los permisos adecuados en AWS (para el servicio y la acción). Para obtener más información, consulte [Proporcionar credenciales de AWS](#).

Creación de una cola

Use el método `CreateQueue` de la clase `SQSClient` y proporciónale un objeto [CreateQueueRequest](#) que describa los parámetros de la cola.

Incluye

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

Código de

```
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;
}
```

Consulte el [ejemplo completo](#)

Mostrar colas

Para enumerar las colas de Amazon SQS de su cuenta, llame a la función miembro `ListQueues` de la clase `SQSClient` y transfíerale un objeto [ListQueuesRequest](#).

Incluye

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

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

Código

```
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;
}
```

Consulte el [ejemplo completo](#)

Cómo obtener la URL de la cola

Para obtener la URL de una cola de Amazon SQS existente, llame a la función miembro `GetQueueUrl` de la clase `SQSClient`.

Incluye

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

Código

```
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;
}
```

Consulte el [ejemplo completo](#)

Eliminar una cola

Proporcione la [URL](#) a la función miembro `DeleteQueue` de la clase `SQSClient`.

Incluye

```
#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;
}
```

Consulte el [ejemplo completo](#)

Más información

- [Cómo funcionan las colas de Amazon SQS](#) en la Guía para desarrolladores de Amazon Simple Queue Service
- [CreateQueue](#) en la Referencia de la API de Amazon Simple Queue Service
- [GetQueueUrl](#) en la Referencia de la API de Amazon Simple Queue Service
- [ListQueues](#) en la Referencia de la API de Amazon Simple Queue Service
- [DeleteQueues](#) en la Referencia de la API de Amazon Simple Queue Service

Envío, recepción y eliminación de mensajes de Amazon SQS

Los mensajes se envían siempre a través de una [cola de SQS](#). Estos ejemplos de C++ muestran cómo utilizar el AWS SDK para C++ para enviar, recibir y eliminar mensajes de Amazon SQS de las colas de SQS.

Requisitos previos

Antes de comenzar, le recomendamos que lea [Primeros pasos con el AWS SDK para C++](#).

Descargue el código de ejemplo y compile la solución tal y como se describe en [Introducción a los ejemplos de código](#).

Para ejecutar los ejemplos, el perfil de usuario que su código utilice para realizar las solicitudes debe tener los permisos adecuados en AWS (para el servicio y la acción). Para obtener más información, consulte [Proporcionar credenciales de AWS](#).

Enviar un mensaje

Puede añadir un único mensaje a una cola de Amazon SQS llamando a la función miembro `SendMessage` de la clase `SqsClient`. Proporcione a `SendMessage` un objeto [SendMessageRequest](#) que contenga la [URL](#) de la cola, el cuerpo del mensaje y un valor de retraso opcional (en segundos).

Incluye

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

Código de

```
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;
}
```

Consulte el [ejemplo completo](#)

Recibir mensajes

Recupere todos los mensajes que se encuentran actualmente en la cola llamando a la función miembro `ReceiveMessage` de la clase `SQSClient` y transfiriéndole la URL de la cola. Los mensajes se devuelven como una lista de objetos [Message](#).

Incluye

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

Código de

```
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;
}
}
```

Consulte el [ejemplo completo](#)

Eliminar mensajes después de su recepción

Tras recibir un mensaje y procesar su contenido, elimine el mensaje de la cola enviando el identificador de recepción y la URL de la cola del mensaje a la función miembro `DeleteMessage` de la clase `SQSClient`.

Incluye

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

Código de

```
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;
}
```

Consulte el [ejemplo completo](#)

Más información

- [Cómo funcionan las colas de Amazon SQS](#) en la Guía para desarrolladores de Amazon Simple Queue Service
- [SendMessage](#) en la Referencia de la API de Amazon Simple Queue Service
- [SendMessageBatch](#) en la Referencia de la API de Amazon Simple Queue Service
- [ReceiveMessage](#) en la Referencia de la API de Amazon Simple Queue Service
- [DeleteMessage](#) en la Referencia de la API de Amazon Simple Queue Service

Habilitar sondeos largos para las colas de mensajes de Amazon SQS

Amazon SQS utiliza el sondeo corto de forma predeterminada: consulta únicamente un subconjunto de los servidores (en función de una distribución aleatoria ponderada) para determinar si hay algún mensaje disponible para su inclusión en la respuesta.

El sondeo largo ayuda a reducir el costo de uso de Amazon SQS al reducir el número de respuestas vacías (cuando no hay ningún mensaje disponible para devolver como respuesta a una solicitud `ReceiveMessage` enviada a una cola de Amazon SQS y eliminar falsas respuestas vacías. Puede definir una frecuencia de sondeo largo de 1-20 segundos.

Requisitos previos

Antes de comenzar, le recomendamos que lea [Primeros pasos con el AWS SDK para C++](#).

Descargue el código de ejemplo y compile la solución tal y como se describe en [Introducción a los ejemplos de código](#).

Para ejecutar los ejemplos, el perfil de usuario que su código utilice para realizar las solicitudes debe tener los permisos adecuados en AWS (para el servicio y la acción). Para obtener más información, consulte [Proporcionar credenciales de AWS](#).

Cómo habilitar el sondeo largo al crear una cola

Para habilitar el sondeo largo al crear una cola de Amazon SQS, establezca el atributo `ReceiveMessageWaitTimeSeconds` en el objeto [CreateQueueRequest](#) antes de llamar a la función miembro `CreateQueue` de la clase `SQSClient`.

Incluye

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

Código de

```
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;
}
```

Consulte el [ejemplo completo](#)

Habilitar el sondeo largo en una cola existente

Además de habilitar el sondeo largo al crear una cola, también puede habilitarlo en una cola existente estableciendo `ReceiveMessageWaitTimeSeconds` en [SetQueueAttributesRequest](#) antes de llamar a la función miembro `SetQueueAttributes` de la clase `SQSClient`.

Incluye

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

Código de

```
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;
}
```

Consulte el [ejemplo completo](#)

Habilitar el sondeo largo al recibir un mensaje

Puede habilitar el sondeo largo al recibir un mensaje estableciendo el tiempo de espera en segundos en el objeto [ReceiveMessageRequest](#) que proporcione a la función miembro `ReceiveMessage` de la clase `SQSClient`.

Note

Debe asegurarse de que el tiempo de espera de la solicitud del cliente de AWS sea mayor que el tiempo del sondeo largo (20 segundos) para que no se agote el tiempo de espera de sus solicitudes `ReceiveMessage` mientras espera al siguiente evento de sondeo.

Incluye

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

Código de

```
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;
}
}
```

Consulte el [ejemplo completo](#)

Más información

- [Sondeo largo de Amazon SQS](#) en la Guía para desarrolladores de Amazon Simple Queue Service
- [CreateQueue](#) en la Referencia de la API de Amazon Simple Queue Service
- [ReceiveMessage](#) en la Referencia de la API de Amazon Simple Queue Service
- [SetQueueAttributes](#) en la Referencia de la API de Amazon Simple Queue Service

Configuración del tiempo de espera de visibilidad en Amazon SQS

Cuando se recibe un mensaje en Amazon SQS, este permanece en la cola hasta que se elimina a fin de garantizar su recepción. Un mensaje que se ha recibido, pero no se ha eliminado, estará disponible en las solicitudes posteriores después de un determinado tiempo de espera de visibilidad para ayudar a evitar que el mensaje se reciba más de una vez antes de que pueda procesarse y eliminarse.

Cuando se utilizan [colas estándar](#), el tiempo de espera de visibilidad no es una garantía de que un mensaje no se reciba dos veces. Si utiliza una cola estándar, asegúrese de que el código pueda tratar aquellas situaciones en las que el mismo mensaje se entrega más de una vez.

Requisitos previos

Antes de comenzar, le recomendamos que lea [Primeros pasos con el AWS SDK para C++](#).

Descargue el código de ejemplo y compile la solución tal y como se describe en [Introducción a los ejemplos de código](#).

Para ejecutar los ejemplos, el perfil de usuario que su código utilice para realizar las solicitudes debe tener los permisos adecuados en AWS (para el servicio y la acción). Para obtener más información, consulte [Proporcionar credenciales de AWS](#).

Establecimiento del tiempo de espera de visibilidad del mensaje al recibirlo

Cuando haya recibido un mensaje, puede modificar el tiempo de espera de su visibilidad transfiriendo su identificador de recepción en un objeto [ChangeMessageVisibilityRequest](#) a la función de miembro `ChangeMessageVisibility` de la clase de `SQSClient`.

Incluye

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

Código de

```
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;
}
```

Consulte el [ejemplo completo](#)

Más información

- [Tiempo de espera de visibilidad](#) en la Guía para desarrolladores de Amazon Simple Queue Service
- [SetQueueAttributes](#) en la Referencia de la API de Amazon Simple Queue Service
- [GetQueueAttributes](#) en la Referencia de la API de Amazon Simple Queue Service
- [ReceiveMessage](#) en la Referencia de la API de Amazon Simple Queue Service
- [ChangeMessageVisibility](#) en la Referencia de la API de Amazon Simple Queue Service
- [ChangeMessageVisibilityBatch](#) en la Referencia de la API de Amazon Simple Queue Service

Uso de colas de mensajes fallidos en Amazon SQS

Amazon SQS añade compatibilidad con las colas de mensajes fallidos. Una cola de mensajes fallidos es una cola a la que otras pueden enviar mensajes que no se pueden procesar correctamente. Puede apartar y aislar estos mensajes en la cola de mensajes fallidos para determinar por qué no se procesaron correctamente.

Para crear una cola de mensajes fallidos, primero debe crear una política de redireccionamiento y, a continuación, configurar la política en los atributos de la cola.

Important

Una cola de mensajes fallidos debe ser el mismo tipo de cola (FIFO o estándar) que la cola de origen. También se debe crear usando la misma Cuenta de AWS y Región de AWS que la cola de origen.

Requisitos previos

Antes de comenzar, le recomendamos que lea [Primeros pasos con el AWS SDK para C++](#).

Descargue el código de ejemplo y compile la solución tal y como se describe en [Introducción a los ejemplos de código](#).

Para ejecutar los ejemplos, el perfil de usuario que su código utilice para realizar las solicitudes debe tener los permisos adecuados en AWS (para el servicio y la acción). Para obtener más información, consulte [Proporcionar credenciales de AWS](#).

Creación de una política de redireccionamiento

La política de redireccionamiento se especifica en JSON. Para crearla, puede usar la clase de utilidad JSON que se proporciona con el AWS SDK para C++.

Este es un ejemplo de función que crea una política de redireccionamiento proporcionándole el ARN de la cola de mensajes fallidos y el número máximo de veces que se puede recibir y no procesar el mensaje antes de que se envíe a la cola de mensajes fallidos.

Incluye

```
#include <aws/core/Aws.h>
#include <aws/core/utils/json/JsonSerializer.h>
```

Código de

```
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();
}
```

Consulte el [ejemplo completo](#)

Definición de la política de redireccionamiento en la cola de origen

Para terminar de configurar la cola de mensajes fallidos, llame a la función miembro `SetQueueAttributes` de la clase de `SQSCClient` con un objeto [SetQueueAttributesRequest](#) para el que haya establecido el atributo `RedrivePolicy` con su política de redireccionamiento JSON.

Incluye

```
#include <aws/sqs/SQSCClient.h>
```

```
#include <aws/sqs/model/SetQueueAttributesRequest.h>
#include <iostream>
```

Código de

```
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;
}
```

Consulte el [ejemplo completo](#)

Más información

- [Uso de las colas de mensajes fallidos de Amazon SQS](#) en la Guía para desarrolladores de Amazon Simple Queue Service.
- [SetQueueAttributes](#) en la Referencia de la API de Amazon Simple Queue Service

Ejemplos de código de SDK para C++

En los ejemplos de código de este tema se muestra cómo utilizar el AWS SDK para C++ with AWS.

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

Algunos servicios contienen categorías de ejemplo adicionales que muestran cómo aprovechar las bibliotecas o funciones específicas del servicio.

Services

- [Ejemplos de ACM con SDK para C++](#)
- [Ejemplos de API Gateway con SDK para C++](#)
- [Ejemplos de Aurora usando SDK para C++](#)
- [Ejemplos de escalado automático con SDK para C++](#)
- [CloudTrail ejemplos de uso de SDK para C++](#)
- [CloudWatch ejemplos de uso de SDK para C++](#)
- [CloudWatch Ejemplos de registros con SDK for C++](#)
- [CodeBuild ejemplos de uso de SDK para C++](#)
- [Ejemplos de proveedor de identidad de Amazon Cognito usando SDK para C++](#)
- [Ejemplos de DynamoDB usando SDK para C++](#)
- [EC2 Ejemplos de Amazon que utilizan SDK para C++](#)
- [EventBridge ejemplos de uso de SDK para C++](#)
- [AWS Glue ejemplos de uso de SDK para C++](#)
- [HealthImaging ejemplos de uso de SDK para C++](#)
- [Ejemplos de IAM usando SDK para C++](#)
- [AWS IoT ejemplos de uso de SDK para C++](#)

- [AWS IoT data ejemplos de uso de SDK para C++](#)
- [Ejemplos de Lambda usando SDK para C++](#)
- [MediaConvert ejemplos de uso de SDK para C++](#)
- [Ejemplos de Amazon RDS usando SDK para C++](#)
- [Ejemplos de servicios de datos de Amazon RDS con SDK para C++](#)
- [Ejemplos de Amazon Rekognition con SDK para C++](#)
- [Ejemplos de Amazon S3 usando SDK para C++](#)
- [Ejemplos de Secrets Manager usando SDK para C++](#)
- [Ejemplos de Amazon SES usando SDK para C++](#)
- [Ejemplos de Amazon SNS usando SDK para C++](#)
- [Ejemplos de Amazon SQS usando SDK para C++](#)
- [AWS STS ejemplos de uso de SDK para C++](#)
- [Ejemplos de Amazon Transcribe Streaming con SDK para C++](#)

Ejemplos de ACM con SDK para C++

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso de AWS SDK para C++ ACM.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas


- [Acciones](#)

Acciones

AddTagsToCertificate

En el siguiente ejemplo de código, se muestra cómo utilizar `AddTagsToCertificate`.

SDK para C++

 Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
#!/ 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();
}

```

- Para obtener más información sobre la API, consulta [AddTagsToCertificate](#) la Referencia AWS SDK para C++ de la API.

DeleteCertificate

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteCertificate.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

//! 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();
}

```

- Para obtener más información sobre la API, consulta [DeleteCertificate](#) la Referencia AWS SDK para C++ de la API.

DescribeCertificate

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeCertificate.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

//! 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();
}

```

- Para obtener más información sobre la API, consulta [DescribeCertificate](#) la Referencia AWS SDK para C++ de la API.

ExportCertificate

En el siguiente ejemplo de código, se muestra cómo utilizar ExportCertificate.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

//! 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();
}
```

- Para obtener más información sobre la API, consulta [ExportCertificate](#) la Referencia AWS SDK para C++ de la API.

GetCertificate

En el siguiente ejemplo de código, se muestra cómo utilizar GetCertificate.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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



```
}
```

- Para obtener más información sobre la API, consulta [GetCertificate](#) la Referencia AWS SDK para C++ de la API.

ImportCertificate

En el siguiente ejemplo de código, se muestra cómo utilizar `ImportCertificate`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener más información sobre la API, consulta [ImportCertificate](#) la Referencia AWS SDK para C++ de la API.

ListCertificates

En el siguiente ejemplo de código, se muestra cómo utilizar `ListCertificates`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
/*! 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;
}
```

- Para obtener más información sobre la API, consulta [ListCertificates](#) la Referencia AWS SDK para C++ de la API.

ListTagsForCertificate

En el siguiente ejemplo de código, se muestra cómo utilizar `ListTagsForCertificate`.

SDK para C++

Note

Hay más información al respecto [en GitHub](#). Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener más información sobre la API, consulta [ListTagsForCertificate](#) la Referencia AWS SDK para C++ de la API.

RemoveTagsFromCertificate

En el siguiente ejemplo de código, se muestra cómo utilizar `RemoveTagsFromCertificate`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
#!/ 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;
    }
}
```

- Para obtener más información sobre la API, consulta [RemoveTagsFromCertificate](#) la Referencia AWS SDK para C++ de la API.

RenewCertificate

En el siguiente ejemplo de código, se muestra cómo utilizar `RenewCertificate`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
#!/ 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;
    }
}

```

- Para obtener más información sobre la API, consulta [RenewCertificate](#) la Referencia AWS SDK para C++ de la API.

RequestCertificate

En el siguiente ejemplo de código, se muestra cómo utilizar RequestCertificate.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

//! 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;
    }
}
```

- Para obtener más información sobre la API, consulta [RequestCertificate](#) la Referencia AWS SDK para C++ de la API.

ResendValidationEmail

En el siguiente ejemplo de código, se muestra cómo utilizar ResendValidationEmail.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener más información sobre la API, consulta [ResendValidationEmail](#) la Referencia AWS SDK para C++ de la API.

UpdateCertificateOptions

En el siguiente ejemplo de código, se muestra cómo utilizar UpdateCertificateOptions.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
#!/ 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;
    }
}
```

- Para obtener más información sobre la API, consulta [UpdateCertificateOptions](#) la Referencia AWS SDK para C++ de la API.

Ejemplos de API Gateway con SDK para C++

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK para C++ mediante API Gateway.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Escenarios](#)

Escenarios

Creación de una aplicación sin servidor para administrar fotos

En el siguiente ejemplo de código se muestra cómo crear una aplicación sin servidor que permita a los usuarios administrar fotos mediante etiquetas.

SDK para C++

Muestra cómo desarrollar una aplicación de administración de activos fotográficos que detecte las etiquetas de las imágenes mediante Amazon Rekognition y las almacene para su posterior recuperación.

Para obtener el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en [GitHub](#).

Para profundizar en el origen de este ejemplo, consulte la publicación en [Comunidad de AWS](#).

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Ejemplos de Aurora usando SDK para C++

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso AWS SDK para C++ de Aurora.

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Introducción](#)
- [Conceptos básicos](#)
- [Acciones](#)
- [Escenarios](#)

Introducción

Introducción a Aurora

En el siguiente ejemplo de código, se muestra cómo empezar a utilizar Aurora.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Código para el CMake archivo 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})
```

Código del archivo de origen `hello_aurora.cpp`.

```
#include <aws/core/Aws.h>
#include <aws/rds/RDSCClient.h>
#include <aws/rds/model/DescribeDBClustersRequest.h>
#include <iostream>

/*
```



```
* A "Hello Aurora" starter application which initializes an Amazon Relational
Database Service (Amazon RDS) client
* and describes the Amazon Aurora (Aurora) clusters.
*
* main function
*
* Usage: 'hello_aurora'
*
*/
int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
    // options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::RDS::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;
}
```

- Para obtener información detallada sobre la API, consulte la [sección Describir DBClusters](#) en la referencia de la AWS SDK para C++ API.

Conceptos básicos

Conceptos básicos

En el siguiente ejemplo de código, se muestra cómo:

- Cree un grupo de parámetros de clúster de base de datos de Aurora y defina los valores de los parámetros.
- Cree un clúster de base de datos que utilice el grupo de parámetros.
- Cree una instancia de base de datos que contenga una base de datos.
- Realice una instantánea del clúster de base de datos y luego limpie los recursos.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
Aws::Client::ClientConfiguration clientConfig;
// Optional: Set to the AWS Region (overrides config file).
// clientConfig.region = "us-east-1";
```

```

//! Routine which creates an Amazon Aurora DB cluster and demonstrates several
operations
//! on that cluster.
/*!
 \sa gettingStartedWithDBClusters()
 \param clientConfiguration: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::gettingStartedWithDBClusters(
    const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::RDS::RDSClient client(clientConfig);

    printAsterisksLine();
    std::cout << "Welcome to the Amazon Relational Database Service (Amazon Aurora)"
                << std::endl;
    std::cout << "get started with DB clusters demo." << std::endl;
    printAsterisksLine();

    std::cout << "Checking for an existing DB cluster parameter group named '" <<
                CLUSTER_PARAMETER_GROUP_NAME << "'." << std::endl;
    Aws::String dbParameterGroupFamily("Undefined");
    bool parameterGroupFound = true;
    {
        // 1. Check if the DB cluster parameter group already exists.
        Aws::RDS::Model::DescribeDBClusterParameterGroupsRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);

        Aws::RDS::Model::DescribeDBClusterParameterGroupsOutcome outcome =
            client.DescribeDBClusterParameterGroups(request);

        if (outcome.IsSuccess()) {
            std::cout << "DB cluster parameter group named '" <<
                CLUSTER_PARAMETER_GROUP_NAME << "' already exists." <<
std::endl;
            dbParameterGroupFamily =
outcome.GetResult().GetDBClusterParameterGroups()[0].GetDBParameterGroupFamily();
        }
        else if (outcome.GetError().GetErrorType() ==
            Aws::RDS::RDSErrors::D_B_PARAMETER_GROUP_NOT_FOUND_FAULT) {
            std::cout << "DB cluster parameter group named '" <<
                CLUSTER_PARAMETER_GROUP_NAME << "' does not exist." <<
std::endl;
            parameterGroupFound = false;
        }
    }
}

```

```
    else {
        std::cerr << "Error with Aurora::DescribeDBClusterParameterGroups. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        return false;
    }
}

if (!parameterGroupFound) {
    Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

    // 2. Get available parameter group families for the specified engine.
    if (!getDBEngineVersions(DB_ENGINE, NO_PARAMETER_GROUP_FAMILY,
                            engineVersions, client)) {
        return false;
    }

    std::cout << "Getting available parameter group families for " << DB_ENGINE
              << "."
              << std::endl;
    std::vector<Aws::String> families;
    for (const Aws::RDS::Model::DBEngineVersion &version: engineVersions) {
        Aws::String family = version.GetDBParameterGroupFamily();
        if (std::find(families.begin(), families.end(), family) ==
            families.end()) {
            families.push_back(family);
            std::cout << " " << families.size() << ": " << family << std::endl;
        }
    }

    int choice = askQuestionForIntRange("Which family do you want to use? ", 1,
                                       static_cast<int>(families.size()));
    dbParameterGroupFamily = families[choice - 1];
}

if (!parameterGroupFound) {
    // 3. Create a DB cluster parameter group.
    Aws::RDS::Model::CreateDBClusterParameterGroupRequest request;
    request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
    request.SetDBParameterGroupFamily(dbParameterGroupFamily);
    request.SetDescription("Example cluster parameter group.");

    Aws::RDS::Model::CreateDBClusterParameterGroupOutcome outcome =
        client.CreateDBClusterParameterGroup(request);
}
```

```

    if (outcome.IsSuccess()) {
        std::cout << "The DB cluster parameter group was successfully created."
                  << std::endl;
    }
    else {
        std::cerr << "Error with Aurora::CreateDBClusterParameterGroup. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        return false;
    }
}

printAsterisksLine();
std::cout << "Let's set some parameter values in your cluster parameter group."
          << std::endl;

Aws::Vector<Aws::RDS::Model::Parameter> autoIncrementParameters;
// 4. Get the parameters in the DB cluster parameter group.
if (!getDBClusterParameters(CLUSTER_PARAMETER_GROUP_NAME, AUTO_INCREMENT_PREFIX,
                            NO_SOURCE,
                            autoIncrementParameters,
                            client)) {
    cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
    return false;
}

Aws::Vector<Aws::RDS::Model::Parameter> updateParameters;

for (Aws::RDS::Model::Parameter &autoIncParameter: autoIncrementParameters) {
    if (autoIncParameter.GetIsModifiable() &&
        (autoIncParameter.GetDataType() == "integer")) {
        std::cout << "The " << autoIncParameter.GetParameterName()
                  << " is described as: " <<
                  autoIncParameter.GetDescription() << "." << std::endl;
        if (autoIncParameter.ParameterValueHasBeenSet()) {
            std::cout << "The current value is "
                      << autoIncParameter.GetParameterValue()
                      << "." << std::endl;
        }
        std::vector<int> splitValues = splitToInts(
            autoIncParameter.GetAllowedValues(), '-');
        if (splitValues.size() == 2) {
            int newValue = askQuestionForIntRange(
                Aws::String("Enter a new value between ") +

```

```

        autoIncParameter.GetAllowedValues() + ": ",
        splitValues[0], splitValues[1]);
    autoIncParameter.SetParameterValue(std::to_string(newValue));
    updateParameters.push_back(autoIncParameter);

    }
    else {
        std::cerr << "Error parsing " << autoIncParameter.GetAllowedValues()
        << std::endl;
    }
}
}

{
    // 5. Modify the auto increment parameters in the DB cluster parameter
group.
    Aws::RDS::Model::ModifyDBClusterParameterGroupRequest request;
    request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
    request.SetParameters(updateParameters);

    Aws::RDS::Model::ModifyDBClusterParameterGroupOutcome outcome =
        client.ModifyDBClusterParameterGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB cluster parameter group was successfully modified."
        << std::endl;
    }
    else {
        std::cerr << "Error with Aurora::ModifyDBClusterParameterGroup. "
        << outcome.GetError().GetMessage()
        << std::endl;
    }
}

std::cout
    << "You can get a list of parameters you've set by specifying a source
of 'user'."
    << std::endl;

    Aws::Vector<Aws::RDS::Model::Parameter> userParameters;
    // 6. Display the modified parameters in the DB cluster parameter group.
    if (!getDBClusterParameters(CLUSTER_PARAMETER_GROUP_NAME, NO_NAME_PREFIX,
"user",
                                userParameters,
```

```

        client)) {
    cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
    return false;
}

for (const auto &userParameter: userParameters) {
    std::cout << " " << userParameter.GetParameterName() << ", " <<
        userParameter.GetDescription() << ", parameter value - "
        << userParameter.GetParameterValue() << std::endl;
}

printAsterisksLine();
std::cout << "Checking for an existing DB Cluster." << std::endl;

Aws::RDS::Model::DBCluster dbCluster;
// 7. Check if the DB cluster already exists.
if (!describeDBCluster(DB_CLUSTER_IDENTIFIER, dbCluster, client)) {
    cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
    return false;
}

Aws::String engineVersionName;
Aws::String engineName;
if (dbCluster.DBClusterIdentifierHasBeenSet()) {
    std::cout << "The DB cluster already exists." << std::endl;
    engineVersionName = dbCluster.GetEngineVersion();
    engineName = dbCluster.GetEngine();
}
else {
    std::cout << "Let's create a DB cluster." << std::endl;
    const Aws::String administratorName = askQuestion(
        "Enter an administrator username for the database: ");
    const Aws::String administratorPassword = askQuestion(
        "Enter a password for the administrator (at least 8 characters): ");
    Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

    // 8. Get a list of engine versions for the parameter group family.
    if (!getDBEngineVersions(DB_ENGINE, dbParameterGroupFamily, engineVersions,
        client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
        return false;
    }
}

```

```

        std::cout << "The available engines for your parameter group family are:"
                << std::endl;

        int index = 1;
        for (const Aws::RDS::Model::DBEngineVersion &engineVersion: engineVersions)
        {
            std::cout << " " << index << ": " << engineVersion.GetEngineVersion()
                << std::endl;
            ++index;
        }
        int choice = askQuestionForIntRange("Which engine do you want to use? ", 1,
static_cast<int>(engineVersions.size()));
        const Aws::RDS::Model::DBEngineVersion engineVersion = engineVersions[choice
-
                                                                    1];

        engineName = engineVersion.GetEngine();
        engineVersionName = engineVersion.GetEngineVersion();
        std::cout << "Creating a DB cluster named '" << DB_CLUSTER_IDENTIFIER
                << "' and database '" << DB_NAME << "'.\n"
                << "The DB cluster is configured to use your custom cluster
parameter group '"
                << CLUSTER_PARAMETER_GROUP_NAME << "', and \n"
                << "selected engine version " << engineVersion.GetEngineVersion()
                << ".\nThis typically takes several minutes." << std::endl;

        Aws::RDS::Model::CreateDBClusterRequest request;
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        request.SetEngine(engineName);
        request.SetEngineVersion(engineVersionName);
        request.SetMasterUsername(administratorName);
        request.SetMasterUserPassword(administratorPassword);

        Aws::RDS::Model::CreateDBClusterOutcome outcome =
                client.CreateDBCluster(request);

        if (outcome.IsSuccess()) {
            std::cout << "The DB cluster creation has started."
                << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBCluster. "

```



```

        << outcome.GetError().GetMessage()
        << std::endl;
    cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, "", "", client);
    return false;
}
}

std::cout << "Waiting for the DB cluster to become available." << std::endl;

int counter = 0;
// 11. Wait for the DB cluster to become available.
do {
    std::this_thread::sleep_for(std::chrono::seconds(1));
    ++counter;
    if (counter > 900) {
        std::cerr << "Wait for cluster to become available timed out after "
            << counter
            << " seconds." << std::endl;
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
            DB_CLUSTER_IDENTIFIER, "", client);
        return false;
    }

    dbCluster = Aws::RDS::Model::DBCluster();
    if (!describeDBCluster(DB_CLUSTER_IDENTIFIER, dbCluster, client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
            DB_CLUSTER_IDENTIFIER, "", client);
        return false;
    }

    if ((counter % 20) == 0) {
        std::cout << "Current DB cluster status is '"
            << dbCluster.GetStatus()
            << "' after " << counter << " seconds." << std::endl;
    }
} while (dbCluster.GetStatus() != "available");

if (dbCluster.GetStatus() == "available") {
    std::cout << "The DB cluster has been created." << std::endl;
}

printAsterisksLine();
Aws::RDS::Model::DBInstance dbInstance;
// 11. Check if the DB instance already exists.

```

```

if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
    cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER, "",
                    client);
    return false;
}

if (dbInstance.DbInstancePortHasBeenSet()) {
    std::cout << "The DB instance already exists." << std::endl;
}
else {
    std::cout << "Let's create a DB instance." << std::endl;

    Aws::String dbInstanceClass;
    // 12. Get a list of instance classes.
    if (!chooseDBInstanceClass(engineName,
                               engineVersionName,
                               dbInstanceClass,
                               client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
                        "",
                        client);
        return false;
    }

    std::cout << "Creating a DB instance named '" << DB_INSTANCE_IDENTIFIER
              << "' with selected DB instance class '" << dbInstanceClass
              << "'.\nThis typically takes several minutes." << std::endl;

    // 13. Create a DB instance.
    Aws::RDS::Model::CreateDBInstanceRequest request;
    request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
    request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
    request.SetEngine(engineName);
    request.SetDBInstanceClass(dbInstanceClass);

    Aws::RDS::Model::CreateDBInstanceOutcome outcome =
        client.CreateDBInstance(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB instance creation has started."
                  << std::endl;
    }
    else {
        std::cerr << "Error with RDS::CreateDBInstance. "

```

```
        << outcome.GetError().GetMessage()
        << std::endl;
    cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME, DB_CLUSTER_IDENTIFIER,
"",
                    client);
    return false;
}
}

std::cout << "Waiting for the DB instance to become available." << std::endl;

counter = 0;
// 14. Wait for the DB instance to become available.
do {
    std::this_thread::sleep_for(std::chrono::seconds(1));
    ++counter;
    if (counter > 900) {
        std::cerr << "Wait for instance to become available timed out after "
        << counter
        << " seconds." << std::endl;
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                        DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER, client);
        return false;
    }

    dbInstance = Aws::RDS::Model::DBInstance();
    if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
        cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                        DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER, client);
        return false;
    }

    if ((counter % 20) == 0) {
        std::cout << "Current DB instance status is '"
        << dbInstance.GetDBInstanceStatus()
        << "' after " << counter << " seconds." << std::endl;
    }
} while (dbInstance.GetDBInstanceStatus() != "available");

if (dbInstance.GetDBInstanceStatus() == "available") {
    std::cout << "The DB instance has been created." << std::endl;
}
```

```

// 15. Display the connection string that can be used to connect a 'mysql' shell
to the database.
displayConnection(dbCluster);

printAsterisksLine();

if (askYesNoQuestion(
    "Do you want to create a snapshot of your DB cluster (y/n)? ") {
    Aws::String snapshotID(DB_CLUSTER_IDENTIFIER + "-" +
        Aws::String(Aws::Utils::UUID::RandomUUID()));
    {
        std::cout << "Creating a snapshot named " << snapshotID << "." <<
std::endl;
        std::cout << "This typically takes a few minutes." << std::endl;

        // 16. Create a snapshot of the DB cluster. (CreateDBClusterSnapshot)
        Aws::RDS::Model::CreateDBClusterSnapshotRequest request;
        request.SetDBClusterIdentifier(DB_CLUSTER_IDENTIFIER);
        request.SetDBClusterSnapshotIdentifier(snapshotID);

        Aws::RDS::Model::CreateDBClusterSnapshotOutcome outcome =
            client.CreateDBClusterSnapshot(request);

        if (outcome.IsSuccess()) {
            std::cout << "Snapshot creation has started."
                << std::endl;
        }
        else {
            std::cerr << "Error with Aurora::CreateDBClusterSnapshot. "
                << outcome.GetError().GetMessage()
                << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
            return false;
        }
    }

    std::cout << "Waiting for the snapshot to become available." << std::endl;

    Aws::RDS::Model::DBClusterSnapshot snapshot;
    counter = 0;
    do {
        std::this_thread::sleep_for(std::chrono::seconds(1));

```

```

        ++counter;
        if (counter > 600) {
            std::cerr << "Wait for snapshot to be available timed out after "
                << counter
                << " seconds." << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
            return false;
        }

        // 17. Wait for the snapshot to become available.
        Aws::RDS::Model::DescribeDBClusterSnapshotsRequest request;
        request.SetDBClusterSnapshotIdentifier(snapshotID);

        Aws::RDS::Model::DescribeDBClusterSnapshotsOutcome outcome =
            client.DescribeDBClusterSnapshots(request);

        if (outcome.IsSuccess()) {
            snapshot = outcome.GetResult().GetDBClusterSnapshots()[0];
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBClusterSnapshots. "
                << outcome.GetError().GetMessage()
                << std::endl;
            cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
client);
            return false;
        }

        if ((counter % 20) == 0) {
            std::cout << "Current snapshot status is '"
                << snapshot.GetStatus()
                << "' after " << counter << " seconds." << std::endl;
        }
    } while (snapshot.GetStatus() != "available");

    if (snapshot.GetStatus() != "available") {
        std::cout << "A snapshot has been created." << std::endl;
    }
}

printAsterisksLine();

```

```

    bool result = true;
    if (askYesNoQuestion(
        "Do you want to delete the DB cluster, DB instance, and parameter group
(y/n)? ")) {
        result = cleanUpResources(CLUSTER_PARAMETER_GROUP_NAME,
                                DB_CLUSTER_IDENTIFIER, DB_INSTANCE_IDENTIFIER,
                                client);
    }

    return result;
}

//! Routine which gets a DB cluster description.
/*!
 \sa describeDBCluster()
 \param dbClusterIdentifier: A DB cluster identifier.
 \param clusterResult: The 'DBCluster' object containing the description.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::describeDBCluster(const Aws::String &dbClusterIdentifier,
                                        Aws::RDS::Model::DBCluster &clusterResult,
                                        const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBClustersRequest request;
    request.SetDBClusterIdentifier(dbClusterIdentifier);

    Aws::RDS::Model::DescribeDBClustersOutcome outcome =
        client.DescribeDBClusters(request);

    bool result = true;
    if (outcome.IsSuccess()) {
        clusterResult = outcome.GetResult().GetDBClusters()[0];
    }
    else if (outcome.GetError().GetErrorType() !=
        Aws::RDS::RDSErrors::D_B_CLUSTER_NOT_FOUND_FAULT) {
        result = false;
        std::cerr << "Error with Aurora::GDescribeDBClusters. "
            << outcome.GetError().GetMessage()
            << std::endl;
    }

    // This example does not log an error if the DB cluster does not exist.
    // Instead, clusterResult is set to empty.
    else {

```

```

        clusterResult = Aws::RDS::Model::DBCluster();
    }

    return result;
}

//! Routine which gets DB parameters using the 'DescribeDBClusterParameters' api.
/*!
 \sa getDBClusterParameters()
 \param parameterGroupName: The name of the cluster parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBClusterParameters(const Aws::String &parameterGroupName,
                                             const Aws::String &namePrefix,
                                             const Aws::String &source,
                                             Aws::Vector<Aws::RDS::Model::Parameter>
&parametersResult,
                                             const Aws::RDS::RDSClient &client) {
    Aws::String marker; // The marker is used for pagination.
    do {
        Aws::RDS::Model::DescribeDBClusterParametersRequest request;
        request.SetDBClusterParameterGroupName(CLUSTER_PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBClusterParametersOutcome outcome =
            client.DescribeDBClusterParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {

```

```

        parametersResult.push_back(parameter);
    }
}
else {
    parametersResult.push_back(parameter);
}
}

marker = outcome.GetResult().GetMarker();
}
else {
    std::cerr << "Error with Aurora::DescribeDBClusterParameters. "
                << outcome.GetError().GetMessage()
                << std::endl;
    return false;
}
} while (!marker.empty());

return true;
}

//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::Aurora::getDBEngineVersions(const Aws::String &engineName,
                                         const Aws::String &parameterGroupFamily,

                                         Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersionsResult,
                                         const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }
}

```



```

    engineVersionsResult.clear();
    Aws::String marker; // The marker is used for pagination.
    do {
        if (!marker.empty()) {
            request.SetMarker(marker);
        }

        Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
            client.DescribeDBEngineVersions(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::DBEngineVersion> &engineVersions =
                outcome.GetResult().GetDBEngineVersions();

            engineVersionsResult.insert(engineVersionsResult.end(),
                                       engineVersions.begin(),
                                       engineVersions.end());
            marker = outcome.GetResult().GetMarker();
        }
        else {
            std::cerr << "Error with Aurora::DescribeDBEngineVersionsRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    } while (!marker.empty());

    return true;
}

//! Routine which gets a DB instance description.
/*!
 \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 after " << counter
                  << " seconds." << std::endl;
        return false;
    }

    Aws::RDS::Model::DBInstance dbInstance = Aws::RDS::Model::DBInstance();
    if (instanceDeleting) {
        if (!describeDBInstance(dbInstanceIdentifier, dbInstance, client)) {
            return false;
        }
        instanceDeleting = dbInstance.DBInstanceIdentifierHasBeenSet();
    }
}
```

```
Aws::RDS::Model::DBCluster dbCluster = Aws::RDS::Model::DBCluster();
if (clusterDeleting) {
    if (!describeDBCluster(dbClusterIdentifier, dbCluster, client)) {
        return false;
    }

    clusterDeleting = dbCluster.DBClusterIdentifierHasBeenSet();
}

if ((counter % 20) == 0) {
    if (instanceDeleting) {
        std::cout << "Current DB instance status is '"
            << dbInstance.GetDBInstanceStatus() << "'" << std::endl;
    }

    if (clusterDeleting) {
        std::cout << "Current DB cluster status is '"
            << dbCluster.GetStatus() << "'" << std::endl;
    }
}

if (!parameterGroupName.empty()) {
    // 21. Delete the DB cluster parameter group.
    Aws::RDS::Model::DeleteDBClusterParameterGroupRequest request;
    request.SetDBClusterParameterGroupName(parameterGroupName);

    Aws::RDS::Model::DeleteDBClusterParameterGroupOutcome outcome =
        client.DeleteDBClusterParameterGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB parameter group was successfully deleted."
            << std::endl;
    }
    else {
        std::cerr << "Error with Aurora::DeleteDBClusterParameterGroup. "
            << outcome.GetError().GetMessage()
            << std::endl;
        result = false;
    }
}

return result;
```

```
}
```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para C++ .
 - [CrearDBCluster](#)
 - [CrearDBClusterParameterGroup](#)
 - [Crear DBCluster instantánea](#)
 - [CrearDBInstance](#)
 - [DeleteDBCluster](#)
 - [DeleteDBClusterParameterGroup](#)
 - [DeleteDBInstance](#)
 - [DescribirDBClusterParameterGroups](#)
 - [Describe DBCluster los parámetros](#)
 - [Describe las DBCluster instantáneas](#)
 - [DescribirDBClusters](#)
 - [Describe las versiones DBEngine](#)
 - [DescribirDBInstances](#)
 - [DescribeOrderableDBInstanceOpciones](#)
 - [ModifyDBClusterParameterGroup](#)

Acciones

CreateDBCluster

En el siguiente ejemplo de código, se muestra cómo utilizar CreateDBCluster.

SDK para C++

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
```

- Para obtener más información sobre la API, consulta la [sección Crear DBCluster](#) en la referencia de la AWS SDK para C++ API.

CreateDBClusterParameterGroup

En el siguiente ejemplo de código, se muestra cómo utilizar CreateDBClusterParameterGroup.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
```

- Para obtener más información sobre la API, consulta la [sección Crear DBClusterParameterGroup](#) en la referencia de la AWS SDK para C++ API.

CreateDBClusterSnapshot

En el siguiente ejemplo de código, se muestra cómo utilizar CreateDBClusterSnapshot.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
    }
}
```

- Para obtener más información sobre la API, consulta [Crear DBCluster una instantánea](#) en la referencia AWS SDK para C++ de la API.

CreateDBInstance

En el siguiente ejemplo de código, se muestra cómo utilizar CreateDBInstance.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
```

- Para obtener más información sobre la API, consulta la [sección Crear DBInstance](#) en la referencia de la AWS SDK para C++ API.

DeleteDBCluster

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteDBCluster.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
    }
```

- Para obtener más información sobre la API, consulta [Eliminar DBCluster](#) en la referencia AWS SDK para C++ de la API.

DeleteDBClusterParameterGroup

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteDBClusterParameterGroup.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
```

- Para obtener más información sobre la API, consulta [Eliminar DBCluster ParameterGroup](#) en la referencia AWS SDK para C++ de la API.

DeleteDBInstance

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteDBInstance.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
    }

```

- Para obtener más información sobre la API, consulta [Eliminar DBInstance](#) en la referencia AWS SDK para C++ de la API.

DescribeDBClusterParameterGroups

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeDBClusterParameterGroups.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

    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;
    }
}

```

- Para obtener más información sobre la API, consulta la [sección Describir DBClusterParameterGroups](#) en la referencia de la AWS SDK para C++ API.

DescribeDBClusterParameters

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeDBClusterParameters.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

    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;
}
```

- Para obtener información detallada sobre la API, consulta [Describir DBCluster los parámetros](#) en la referencia de la AWS SDK para C++ API.

DescribeDBClusterSnapshots

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeDBClusterSnapshots.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
    }

```

- Para obtener más información sobre la API, consulte [Describir DBCluster las instantáneas](#) en la referencia de AWS SDK para C++ la API.

DescribeDBClusters

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeDBClusters.

SDK para C++

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

    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;
}

```

- Para obtener más información sobre la API, consulta la [sección Describir DBClusters](#) en la referencia de la AWS SDK para C++ API.

DescribeDBEngineVersions

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeDBEngineVersions.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

    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;
}

```

- Para obtener más información sobre la API, consulta la [sección Describir DBEngine las versiones](#) en la referencia de la AWS SDK para C++ API.

DescribeDBInstances

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeDBInstances.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

    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;
}


```

- Para obtener más información sobre la API, consulta la [sección Describir DBInstances](#) en la referencia de la AWS SDK para C++ API.

DescribeOrderableDBInstanceOptions

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeOrderableDBInstanceOptions`.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}


```

- Para obtener más información sobre la API, consulta [DescribeOrderableDBInstanceClasses opciones](#) en la referencia AWS SDK para C++ de la API.

ModifyDBClusterParameterGroup

En el siguiente ejemplo de código, se muestra cómo utilizar ModifyDBClusterParameterGroup.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
```

- Para obtener más información sobre la API, consulta la [sección Modificar DBClusterParameterGroup](#) en la referencia de la AWS SDK para C++ API.

Escenarios

Crear un rastreador de elementos de trabajo de Aurora Serverless

El siguiente ejemplo de código muestra cómo crear una aplicación web que realice un seguimiento de los elementos de trabajo de una base de datos de Amazon Aurora sin servidor y use Amazon Simple Email Service (Amazon SES) para enviar informes.

SDK para C++

Muestra cómo crear una aplicación web que realice un seguimiento de los elementos de trabajo almacenados en una base de datos de Amazon Aurora sin servidor e informe al respecto.

Para obtener el código fuente completo y las instrucciones sobre cómo configurar una API REST de C++ que consulte los datos de Amazon Aurora Serverless y para que la utilice una aplicación de React, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- Aurora
- Amazon RDS
- Servicio de datos de Amazon RDS
- Amazon SES

Ejemplos de escalado automático con SDK para C++

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el AWS SDK para C++ uso de Auto Scaling.

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Introducción](#)
- [Conceptos básicos](#)
- [Acciones](#)

Introducción

Introducción al escalado automático

El siguiente ejemplo de código muestra cómo empezar a usar Auto Scaling.

SDK para C++

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Código para el CMake archivo 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})

```

Código del archivo de origen 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;
}
```

- Para obtener más información sobre la API, consulte [DescribeAutoScalingGroups](#) la Referencia AWS SDK para C++ de la API.

Conceptos básicos

Conceptos básicos

En el siguiente ejemplo de código, se muestra cómo:

- Cree un grupo de Amazon EC2 Auto Scaling con una plantilla de lanzamiento y zonas de disponibilidad, y obtenga información sobre las instancias en ejecución.
- Habilita la recopilación de CloudWatch métricas de Amazon.
- Actualizar la capacidad deseada del grupo y esperar a que una instancia se inicie
- Terminar una instancia del grupo.
- Mostrar las actividades de escalado que se producen como respuesta a las solicitudes de los usuarios y a los cambios de capacidad
- Obtén estadísticas para CloudWatch las métricas y, a continuación, limpia los recursos.

SDK para C++

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

//! Routine which demonstrates using an Auto Scaling group
//! to manage Amazon EC2 instances.
/*!
 \sa groupsAndInstancesScenario()
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::groupsAndInstancesScenario(
    const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::String templateName;
    Aws::EC2::EC2Client ec2Client(clientConfig);

    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " "
        << std::endl;
    std::cout
        << "Welcome to the Amazon Elastic Compute Cloud (Amazon EC2) Auto
Scaling "
        << "demo for managing groups and instances." << std::endl;
    std::cout << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << " \n"
        << std::endl;

    std::cout << "This example requires an EC2 launch template." << std::endl;
    if (askYesNoQuestion(
        "Would you like to use an existing EC2 launch template (y/n)? ")) {

        // 1. Specify the name of an existing EC2 launch template.
        templateName = askQuestion(
            "Enter the name of the existing EC2 launch template. ");

        Aws::EC2::Model::DescribeLaunchTemplatesRequest request;
        request.AddLaunchTemplateName(templateName);
        Aws::EC2::Model::DescribeLaunchTemplatesOutcome outcome =
            ec2Client.DescribeLaunchTemplates(request);

        if (outcome.IsSuccess()) {
            std::cout << "Validated the EC2 launch template '" << templateName
                << "' exists by calling DescribeLaunchTemplate." << std::endl;
        }
        else {
            std::cerr << "Error validating the existence of the launch template. "
                << outcome.GetError().GetMessage()
                << std::endl;
        }
    }
}

```



```

}
else { // 2. Or create a new EC2 launch template.
    templateName = askQuestion("Enter the name for a new EC2 launch template:
");

    Aws::EC2::Model::CreateLaunchTemplateRequest request;
    request.SetLaunchTemplateName(templateName);

    Aws::EC2::Model::RequestLaunchTemplateData requestLaunchTemplateData;

requestLaunchTemplateData.SetInstanceType(EC2_LAUNCH_TEMPLATE_INSTANCE_TYPE);
    requestLaunchTemplateData.SetImageId(EC2_LAUNCH_TEMPLATE_IMAGE_ID);

    request.SetLaunchTemplateData(requestLaunchTemplateData);

    Aws::EC2::Model::CreateLaunchTemplateOutcome outcome =
        ec2Client.CreateLaunchTemplate(request);

    if (outcome.IsSuccess()) {
        std::cout << "The EC2 launch template '" << templateName << " was
created."
                << std::endl;
    }
    else if (outcome.GetError().GetExceptionName() ==
        "InvalidLaunchTemplateName.AlreadyExistsException") {
        std::cout << "The EC2 template '" << templateName << "' already exists"
                << std::endl;
    }
    else {
        std::cerr << "Error with EC2::CreateLaunchTemplate. "
                << outcome.GetError().GetMessage()
                << std::endl;
    }
}

}
Aws::AutoScaling::AutoScalingClient autoScalingClient(clientConfig);
std::cout << "Let's create an Auto Scaling group." << std::endl;
Aws::String groupName = askQuestion(
    "Enter a name for the Auto Scaling group: ");
// 3. Retrieve a list of EC2 Availability Zones.
Aws::Vector<Aws::EC2::Model::AvailabilityZone> availabilityZones;
{
    Aws::EC2::Model::DescribeAvailabilityZonesRequest request;

    Aws::EC2::Model::DescribeAvailabilityZonesOutcome outcome =

```

```

        ec2Client.DescribeAvailabilityZones(request);

    if (outcome.IsSuccess()) {
        std::cout
            << "EC2 instances can be created in the following Availability
Zones:"
            << std::endl;

        availabilityZones = outcome.GetResult().GetAvailabilityZones();
        for (size_t i = 0; i < availabilityZones.size(); ++i) {
            std::cout << "    " << i + 1 << ". "
                << availabilityZones[i].GetZoneName() << std::endl;
        }
    }
    else {
        std::cerr << "Error with EC2::DescribeAvailabilityZones. "
            << outcome.GetError().GetMessage()
            << std::endl;
        cleanupResources("", templateName, autoScalingClient, ec2Client);
        return false;
    }
}

int availabilityZoneChoice = askQuestionForIntRange(
    "Choose an Availability Zone: ", 1,
    static_cast<int>(availabilityZones.size()));
// 4. Create an Auto Scaling group with the specified Availability Zone.
{
    Aws::AutoScaling::Model::CreateAutoScalingGroupRequest request;
    request.SetAutoScalingGroupName(groupName);
    Aws::Vector<Aws::String> availabilityGroupZones;
    availabilityGroupZones.push_back(
        availabilityZones[availabilityZoneChoice - 1].GetZoneName());
    request.SetAvailabilityZones(availabilityGroupZones);
    request.SetMaxSize(1);
    request.SetMinSize(1);

    Aws::AutoScaling::Model::LaunchTemplateSpecification
launchTemplateSpecification;
    launchTemplateSpecification.SetLaunchTemplateName(templateName);
    request.SetLaunchTemplate(launchTemplateSpecification);

    Aws::AutoScaling::Model::CreateAutoScalingGroupOutcome outcome =
        autoScalingClient.CreateAutoScalingGroup(request);
}

```

```

    if (outcome.IsSuccess()) {
        std::cout << "Created Auto Scaling group '" << groupName << "'..."
            << std::endl;
    }
    else if (outcome.GetError().GetErrorType() ==
        Aws::AutoScaling::AutoScalingErrors::ALREADY_EXISTS_FAULT) {
        std::cout << "Auto Scaling group '" << groupName << "' already exists."
            << std::endl;
    }
    else {
        std::cerr << "Error with AutoScaling::CreateAutoScalingGroup. "
            << outcome.GetError().GetMessage()
            << std::endl;
        cleanupResources("", templateName, autoScalingClient, ec2Client);
        return false;
    }
}

Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> autoScalingGroups;
if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
    autoScalingClient)) {
    std::cout << "Here is the Auto Scaling group description." << std::endl;
    if (!autoScalingGroups.empty()) {
        logAutoScalingGroupInfo(autoScalingGroups);
    }
}
else {
    cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
    return false;
}

std::cout
    << "Waiting for the EC2 instance in the Auto Scaling group to become
active..."
    << std::endl;
if (!waitForInstances(groupName, autoScalingGroups, autoScalingClient)) {
    cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
    return false;
}

bool enableMetrics = askYesNoQuestion(
    "Do you want to collect metrics about the A"
    "Auto Scaling group during this demo (y/n)? ");

```

```
// 7. Optionally enable metrics collection for the Auto Scaling group.
if (enableMetrics) {
    Aws::AutoScaling::Model::EnableMetricsCollectionRequest request;
    request.SetAutoScalingGroupName(groupName);

    request.AddMetrics("GroupMinSize");
    request.AddMetrics("GroupMaxSize");
    request.AddMetrics("GroupDesiredCapacity");
    request.AddMetrics("GroupInServiceInstances");
    request.AddMetrics("GroupTotalInstances");
    request.SetGranularity("1Minute");

    Aws::AutoScaling::Model::EnableMetricsCollectionOutcome outcome =
        autoScalingClient.EnableMetricsCollection(request);
    if (outcome.IsSuccess()) {
        std::cout << "Auto Scaling metrics have been enabled."
                  << std::endl;
    }
    else {
        std::cerr << "Error with AutoScaling::EnableMetricsCollection. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }
}

std::cout << "Let's update the maximum number of EC2 instances in '" <<
groupName <<
    "' from 1 to 3." << std::endl;
askQuestion("Press enter to continue: ", alwaysTrueTest);
// 8. Update the Auto Scaling group, setting a new maximum size.
{
    Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
    request.SetAutoScalingGroupName(groupName);
    request.SetMaxSize(3);

    Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
        autoScalingClient.UpdateAutoScalingGroup(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
}
```

```

        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }
}

if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
    autoScalingClient)) {
    if (!autoScalingGroups.empty()) {
        const auto &instances = autoScalingGroups[0].GetInstances();
        std::cout
            << "The group still has one running EC2 instance, but it can
have up to 3.\n"
            << std::endl;
        logAutoScalingGroupInfo(autoScalingGroups);
    }
    else {
        std::cerr
            << "No EC2 launch groups were retrieved from DescribeGroup
request."
            << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }
}

std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << "\n"
    << std::endl;
std::cout << "Let's update the desired capacity in '" << groupName <<
    "' from 1 to 2." << std::endl;
askQuestion("Press enter to continue: ", alwaysTrueTest);
// 9. Update the Auto Scaling group, setting a new desired capacity.
{
    Aws::AutoScaling::Model::SetDesiredCapacityRequest request;
    request.SetAutoScalingGroupName(groupName);
    request.SetDesiredCapacity(2);

    Aws::AutoScaling::Model::SetDesiredCapacityOutcome outcome =
        autoScalingClient.SetDesiredCapacity(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error with AutoScaling::SetDesiredCapacityRequest. "
            << outcome.GetError().GetMessage()
            << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
    }
}

```

```

        return false;
    }
}

if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
                                       autoScalingClient)) {
    if (!autoScalingGroups.empty()) {
        std::cout
            << "Here is the current state of the group." << std::endl;
        logAutoScalingGroupInfo(autoScalingGroups);
    }
    else {
        std::cerr
            << "No EC2 launch groups were retrieved from DescribeGroup
request."
            << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }
}

std::cout << "Waiting for the new EC2 instance to start..." << std::endl;
waitForInstances(groupName, autoScalingGroups, autoScalingClient);

std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << "\n"
    << std::endl;

std::cout << "Let's terminate one of the EC2 instances in " << groupName << "."
    << std::endl;
std::cout << "Because the desired capacity is 2, another EC2 instance will start
"
    << "to replace the terminated EC2 instance."
    << std::endl;
std::cout << "The currently running EC2 instances are:" << std::endl;

if (autoScalingGroups.empty()) {
    std::cerr << "Error describing groups. No groups returned." << std::endl;
    cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
    return false;
}

int instanceNumber = 1;
Aws::Vector<Aws::String> instanceIDs = instancesToInstanceIDs(
    autoScalingGroups[0].GetInstances());

```

```

    for (const Aws::String &instanceID: instanceIDs) {
        std::cout << "    " << instanceNumber << ". " << instanceID << std::endl;
        ++instanceNumber;
    }

    instanceNumber = askQuestionForIntRange("Which EC2 instance do you want to stop?",
",
                                        1,
                                        static_cast<int>(instanceIDs.size()));

// 10. Terminate an EC2 instance in the Auto Scaling group.
{
    Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest request;
    request.SetInstanceId(instanceIDs[instanceNumber - 1]);
    request.SetShouldDecrementDesiredCapacity(false);

    Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome outcome
=
        autoScalingClient.TerminateInstanceInAutoScalingGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "Waiting for EC2 instance with ID '"
            << instanceIDs[instanceNumber - 1] << "' to terminate..."
            << std::endl;
    }
    else {
        std::cerr << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
            << outcome.GetError().GetMessage()
            << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }
}

waitForInstances(groupName, autoScalingGroups, autoScalingClient);

std::cout << "\n" << std::setfill('*') << std::setw(ASTERISK_FILL_WIDTH) << "\n"
    << std::endl;
std::cout << "Let's get a report of scaling activities for EC2 launch group '"
    << groupName << "'."
    << std::endl;
askQuestion("Press enter to continue: ", alwaysTrueTest);
// 11. Get a description of activities for the Auto Scaling group.

```

```

{
    Aws::AutoScaling::Model::DescribeScalingActivitiesRequest request;
    request.SetAutoScalingGroupName(groupName);

    Aws::Vector<Aws::AutoScaling::Model::Activity> allActivities;
    Aws::String nextToken; // Used for pagination;
    do {
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }
        Aws::AutoScaling::Model::DescribeScalingActivitiesOutcome outcome =
            autoScalingClient.DescribeScalingActivities(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::AutoScaling::Model::Activity> &activities =
                outcome.GetResult().GetActivities();
            allActivities.insert(allActivities.end(), activities.begin(),
activities.end());
            nextToken = outcome.GetResult().GetNextToken();
        }
        else {
            std::cerr << "Error with AutoScaling::DescribeScalingActivities. "
                << outcome.GetError().GetMessage()
                << std::endl;
            cleanupResources(groupName, templateName, autoScalingClient,
ec2Client);
            return false;
        }
    } while (!nextToken.empty());

    std::cout << "Found " << allActivities.size() << " activities."
        << std::endl;
    std::cout << "Activities are ordered with the most recent first."
        << std::endl;
    for (const Aws::AutoScaling::Model::Activity &activity: allActivities) {
        std::cout << activity.GetDescription() << std::endl;
        std::cout << activity.GetDetails() << std::endl;
    }
}

if (enableMetrics) {
    if (!logAutoScalingMetrics(groupName, clientConfig)) {
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }
}

```



```

    }
}

std::cout << "Let's clean up." << std::endl;
askQuestion("Press enter to continue: ", alwaysTrueTest);

// 13. Disable metrics collection if enabled.
if (enableMetrics) {
    Aws::AutoScaling::Model::DisableMetricsCollectionRequest request;
    request.SetAutoScalingGroupName(groupName);

    Aws::AutoScaling::Model::DisableMetricsCollectionOutcome outcome =
        autoScalingClient.DisableMetricsCollection(request);

    if (outcome.IsSuccess()) {
        std::cout << "Metrics collection has been disabled." << std::endl;
    }
    else {
        std::cerr << "Error with AutoScaling::DisableMetricsCollection. "
            << outcome.GetError().GetMessage()
            << std::endl;
        cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
        return false;
    }
}

return cleanupResources(groupName, templateName, autoScalingClient, ec2Client);
}

//! Routine which waits for EC2 instances in an Auto Scaling group to
//! complete startup or shutdown.
/*!
 \sa waitForInstances()
 \param groupName: An Auto Scaling group name.
 \param autoScalingGroups: Vector to receive 'AutoScalingGroup' records.
 \param client: 'AutoScalingClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::waitForInstances(const Aws::String &groupName,

    Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> &autoScalingGroups,
    const Aws::AutoScaling::AutoScalingClient
&client) {
    bool ready = false;

```

```
const std::vector<Aws::String> READY_STATES = {"InService", "Terminated"};

int count = 0;
int desiredCapacity = 0;
std::this_thread::sleep_for(std::chrono::seconds(4));
while (!ready) {
    if (WAIT_FOR_INSTANCES_TIMEOUT < count) {
        std::cerr << "Wait for instance timed out." << std::endl;
        return false;
    }

    std::this_thread::sleep_for(std::chrono::seconds(1));
    ++count;
    if (!describeGroup(groupName, autoScalingGroups, client)) {
        return false;
    }
    Aws::Vector<Aws::String> instanceIDs;
    if (!autoScalingGroups.empty()) {
        instanceIDs =
instancesToInstanceIDs(autoScalingGroups[0].GetInstances());
        desiredCapacity = autoScalingGroups[0].GetDesiredCapacity();
    }

    if (instanceIDs.empty()) {
        if (desiredCapacity == 0) {
            break;
        }
        else {
            if ((count % 5) == 0) {
                std::cout << "No instance IDs returned for group." << std::endl;
            }

            continue;
        }
    }

    // 6. Check lifecycle state of the instances using
DescribeAutoScalingInstances.
    Aws::AutoScaling::Model::DescribeAutoScalingInstancesRequest request;
    request.SetInstanceIds(instanceIDs);

    Aws::AutoScaling::Model::DescribeAutoScalingInstancesOutcome outcome =
        client.DescribeAutoScalingInstances(request);
```

```

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::AutoScaling::Model::AutoScalingInstanceDetails>
&instancesDetails =
                outcome.GetResult().GetAutoScalingInstances();
            ready = instancesDetails.size() >= desiredCapacity;
            for (const Aws::AutoScaling::Model::AutoScalingInstanceDetails &details:
instancesDetails) {
                if (!stringInVector(details.GetLifecycleState(), READY_STATES)) {
                    ready = false;
                    break;
                }
            }
            // Log the status while waiting.
            if (((count % 5) == 1) || ready) {
                logInstancesLifecycleState(instancesDetails);
            }
        }
        else {
            std::cerr << "Error with AutoScaling::DescribeAutoScalingInstances. "
                << outcome.GetError().GetMessage()
                << std::endl;
            return false;
        }
    }

    if (!describeGroup(groupName, autoScalingGroups, client)) {
        return false;
    }

    return true;
}

//! Routine to cleanup resources created in 'groupsAndInstancesScenario'.
/*!
 \sa cleanupResources()
 \param groupName: Optional Auto Scaling group name.
 \param templateName: Optional EC2 launch template name.
 \param autoScalingClient: 'AutoScalingClient' instance.
 \param ec2Client: 'EC2Client' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::cleanupResources(const Aws::String &groupName,
                                           const Aws::String &templateName,

```

```

const Aws::AutoScaling::AutoScalingClient
&autoScalingClient,
const Aws::EC2::EC2Client &ec2Client) {
    bool result = true;

    // 14. Delete the Auto Scaling group.
    if (!groupName.empty() &&
        (askYesNoQuestion(
            Aws::String("Delete the Auto Scaling group '" + groupName +
                "' (y/n)?")))) {
        {
            Aws::AutoScaling::Model::UpdateAutoScalingGroupRequest request;
            request.SetAutoScalingGroupName(groupName);
            request.SetMinSize(0);
            request.SetDesiredCapacity(0);

            Aws::AutoScaling::Model::UpdateAutoScalingGroupOutcome outcome =
                autoScalingClient.UpdateAutoScalingGroup(request);

            if (outcome.IsSuccess()) {
                std::cout
                    << "The minimum size and desired capacity of the Auto
Scaling group "
                    << "was set to zero before terminating the instances."
                    << std::endl;
            }
            else {
                std::cerr << "Error with AutoScaling::UpdateAutoScalingGroup. "
                    << outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
        }
    }

    Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> autoScalingGroups;
    if (AwsDoc::AutoScaling::describeGroup(groupName, autoScalingGroups,
        autoScalingClient)) {
        if (!autoScalingGroups.empty()) {
            Aws::Vector<Aws::String> instanceIDs = instancesToInstanceIDs(
                autoScalingGroups[0].GetInstances());
            for (const Aws::String &instanceID: instanceIDs) {
                Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupRequest request;
                request.SetInstanceId(instanceID);
                request.SetShouldDecrementDesiredCapacity(true);
            }
        }
    }
}

```

```

Aws::AutoScaling::Model::TerminateInstanceInAutoScalingGroupOutcome outcome =
    autoScalingClient.TerminateInstanceInAutoScalingGroup(
        request);

    if (outcome.IsSuccess()) {
        std::cout << "Initiating termination of EC2 instance '"
            << instanceID << "'." << std::endl;
    }
    else {
        std::cerr
            << "Error with
AutoScaling::TerminateInstanceInAutoScalingGroup. "
            << outcome.GetError().GetMessage() << std::endl;
        result = false;
    }
}
}

std::cout
    << "Waiting for the EC2 instances to terminate before deleting
the "
    << "Auto Scaling group..." << std::endl;
waitForInstances(groupName, autoScalingGroups, autoScalingClient);
}

{
    Aws::AutoScaling::Model::DeleteAutoScalingGroupRequest request;
    request.SetAutoScalingGroupName(groupName);

    Aws::AutoScaling::Model::DeleteAutoScalingGroupOutcome outcome =
        autoScalingClient.DeleteAutoScalingGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "Auto Scaling group '" << groupName << "' was deleted."
            << std::endl;
    }
    else {
        std::cerr << "Error with AutoScaling::DeleteAutoScalingGroup. "
            << outcome.GetError().GetMessage()
            << std::endl;
        result = false;
    }
}

```

```

    }
}

// 15. Delete the EC2 launch template.
if (!templateName.empty() && (askYesNoQuestion(
    Aws::String("Delete the EC2 launch template '" + templateName +
        "' (y/n)?"))) {
    Aws::EC2::Model::DeleteLaunchTemplateRequest request;
    request.SetLaunchTemplateName(templateName);

    Aws::EC2::Model::DeleteLaunchTemplateOutcome outcome =
        ec2Client.DeleteLaunchTemplate(request);

    if (outcome.IsSuccess()) {
        std::cout << "EC2 launch template '" << templateName << "' was deleted."
            << std::endl;
    }
    else {
        std::cerr << "Error with EC2::DeleteLaunchTemplate. "
            << outcome.GetError().GetMessage()
            << std::endl;
        result = false;
    }
}

return result;
}

//! Routine which retrieves Auto Scaling group descriptions.
/*!
 \sa describeGroup()
 \param groupName: An Auto Scaling group name.
 \param autoScalingGroups: Vector to receive 'AutoScalingGroup' records.
 \param client: 'AutoScalingClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::AutoScaling::describeGroup(const Aws::String &groupName,

    Aws::Vector<Aws::AutoScaling::Model::AutoScalingGroup> &autoScalingGroup,
        const Aws::AutoScaling::AutoScalingClient

&client) {
    // 5. Retrieve a description of the Auto Scaling group.
    Aws::AutoScaling::Model::DescribeAutoScalingGroupsRequest request;
    Aws::Vector<Aws::String> groupNames;

```

```
groupNames.push_back(groupName);
request.SetAutoScalingGroupNames(groupNames);

Aws::AutoScaling::Model::DescribeAutoScalingGroupsOutcome outcome =
    client.DescribeAutoScalingGroups(request);

if (outcome.IsSuccess()) {
    autoScalingGroup = outcome.GetResult().GetAutoScalingGroups();
}
else {
    std::cerr << "Error with AutoScaling::DescribeAutoScalingGroups. "
                << outcome.GetError().GetMessage()
                << std::endl;
}

return outcome.IsSuccess();
}
```


- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para C++ .
 - [CreateAutoScalingGroup](#)
 - [DeleteAutoScalingGroup](#)
 - [DescribeAutoScalingGroups](#)
 - [DescribeAutoScalingInstances](#)
 - [DescribeScalingActivities](#)
 - [DisableMetricsCollection](#)
 - [EnableMetricsCollection](#)
 - [SetDesiredCapacity](#)
 - [TerminateInstanceInAutoScalingGroup](#)
 - [UpdateAutoScalingGroup](#)

Acciones

CreateAutoScalingGroup

En el siguiente ejemplo de código, se muestra cómo utilizar CreateAutoScalingGroup.

SDK para C++

 Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;  
  
    }
```

- Para obtener más información sobre la API, consulta [CreateAutoScalingGroup](#) la Referencia AWS SDK para C++ de la API.

DeleteAutoScalingGroup

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteAutoScalingGroup.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
    }
}
```

- Para obtener más información sobre la API, consulta [DeleteAutoScalingGroup](#) la Referencia AWS SDK para C++ de la API.

DescribeAutoScalingGroups

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeAutoScalingGroups.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
```

```
}
```

- Para obtener más información sobre la API, consulta [DescribeAutoScalingGroups](#) la Referencia AWS SDK para C++ de la API.

DescribeAutoScalingInstances

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeAutoScalingInstances`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
```

```
}

```

- Para obtener más información sobre la API, consulta [DescribeAutoScalingInstances](#) la Referencia AWS SDK para C++ de la API.

DescribeScalingActivities

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeScalingActivities.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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());
    }
} while (outcome.IsSuccess() && !nextToken.empty());

```

```
        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;
}
```

- Para obtener más información sobre la API, consulta [DescribeScalingActivities](#) la Referencia AWS SDK para C++ de la API.

DisableMetricsCollection

En el siguiente ejemplo de código, se muestra cómo utilizar `DisableMetricsCollection`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
}
```

- Para obtener más información sobre la API, consulta [DisableMetricsCollection](#) la Referencia AWS SDK para C++ de la API.

EnableMetricsCollection

En el siguiente ejemplo de código, se muestra cómo utilizar `EnableMetricsCollection`.

SDK para C++

Note

Hay más información al respecto [en GitHub](#). Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
}

```

- Para obtener más información sobre la API, consulta [EnableMetricsCollection](#) la Referencia AWS SDK para C++ de la API.

SetDesiredCapacity

En el siguiente ejemplo de código, se muestra cómo utilizar SetDesiredCapacity.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

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;
}
}
```

- Para obtener más información sobre la API, consulta [SetDesiredCapacity](#) la Referencia AWS SDK para C++ de la API.

TerminateInstanceInAutoScalingGroup

En el siguiente ejemplo de código, se muestra cómo utilizar `TerminateInstanceInAutoScalingGroup`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
}
```

- Para obtener más información sobre la API, consulta [TerminateInstanceInAutoScalingGroup](#) la Referencia AWS SDK para C++ de la API.

UpdateAutoScalingGroup

En el siguiente ejemplo de código, se muestra cómo utilizar UpdateAutoScalingGroup.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
}
```

- Para obtener más información sobre la API, consulta [UpdateAutoScalingGroup](#) la Referencia AWS SDK para C++ de la API.

CloudTrail ejemplos de uso de SDK para C++

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK para C++ with CloudTrail.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas


- [Acciones](#)

Acciones

CreateTrail

En el siguiente ejemplo de código, se muestra cómo utilizar CreateTrail.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener más información sobre la API, consulta [CreateTrail](#)la Referencia AWS SDK para C++ de la API.

DeleteTrail

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteTrail.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener más información sobre la API, consulta [DeleteTrail](#) en la Referencia AWS SDK para C++ de la API.

DescribeTrail

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeTrail.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener más información sobre la API, consulta [DescribeTrail](#) en la Referencia AWS SDK para C++ de la API.

LookupEvents

En el siguiente ejemplo de código, se muestra cómo utilizar LookupEvents.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
// 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;
}
```

- Para obtener más información sobre la API, consulta [LookupEvents](#) la Referencia AWS SDK para C++ de la API.

CloudWatch ejemplos de uso de SDK para C++

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK para C++ with CloudWatch.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Acciones](#)

Acciones

DeleteAlarms

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteAlarms.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Incluir los archivos requeridos.

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

Elimine la alarma.

```
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;
}
```


- Para obtener más información sobre la API, consulta [DeleteAlarms](#) la Referencia AWS SDK para C++ de la API.

DescribeAlarmsForMetric

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeAlarmsForMetric`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Incluir los archivos requeridos.

```
#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>
```

Describe las alarmas

```
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();
}
```

- Para obtener más información sobre la API, consulta [DescribeAlarmsForMetric](#) la Referencia AWS SDK para C++ de la API.

DisableAlarmActions

En el siguiente ejemplo de código, se muestra cómo utilizar `DisableAlarmActions`.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Incluir los archivos requeridos.

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

Deshabilite las acciones de alarma.

```
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;
}
```

- Para obtener más información sobre la API, consulta [DisableAlarmActions](#) la Referencia AWS SDK para C++ de la API.

EnableAlarmActions

En el siguiente ejemplo de código, se muestra cómo utilizar `EnableAlarmActions`.

SDK para C++

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Incluir los archivos requeridos.

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

Habilite las acciones de alarma.

```
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;
```

- Para obtener más información sobre la API, consulta [EnableAlarmActions](#) la Referencia AWS SDK para C++ de la API.

ListMetrics

En el siguiente ejemplo de código, se muestra cómo utilizar `ListMetrics`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Incluir los archivos requeridos.

```
#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>
```

Enumere las métricas.

```
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();
}

```

- Para obtener más información sobre la API, consulta [ListMetrics](#) la Referencia AWS SDK para C++ de la API.

PutMetricAlarm

En el siguiente ejemplo de código, se muestra cómo utilizar PutMetricAlarm.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Incluir los archivos requeridos.

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

Cree la alarma para ver la métrica.

```
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;
}
```


- Para obtener más información sobre la API, consulta [PutMetricAlarm](#) la Referencia AWS SDK para C++ de la API.

PutMetricData

En el siguiente ejemplo de código, se muestra cómo utilizar PutMetricData.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Incluir los archivos requeridos.

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

Coloque datos en la métrica

```
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;
}
```

- Para obtener más información sobre la API, consulta [PutMetricData](#) la Referencia AWS SDK para C++ de la API.

CloudWatch Ejemplos de registros con SDK for C++

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso AWS SDK para C++ de CloudWatch registros.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas


- [Acciones](#)

Acciones

DeleteSubscriptionFilter

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteSubscriptionFilter.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Incluir los archivos requeridos.

```
#include <aws/core/Aws.h>
#include <aws/core/Utils/Outcome.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/DeleteSubscriptionFilterRequest.h>
#include <iostream>
```

Elimine el filtro de suscripción.

```
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;
}
```

- Para obtener más información sobre la API, consulta [DeleteSubscriptionFilter](#) la Referencia AWS SDK para C++ de la API.

DescribeSubscriptionFilters

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeSubscriptionFilters.

SDK para C++

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Incluir los archivos requeridos.

```
#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>
```

Enumere los filtros de suscripción.

```
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();
}

```

- Para obtener más información sobre la API, consulta [DescribeSubscriptionFilters](#) la Referencia AWS SDK para C++ de la API.

PutSubscriptionFilter

En el siguiente ejemplo de código, se muestra cómo utilizar PutSubscriptionFilter.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Incluir los archivos requeridos.

```

#include <aws/core/Aws.h>
#include <aws/logs/CloudWatchLogsClient.h>
#include <aws/logs/model/PutSubscriptionFilterRequest.h>
#include <aws/core/utils/Outcome.h>

```

```
#include <iostream>
```

Cree el filtro de suscripción.

```
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;
}
```

- Para obtener más información sobre la API, consulta [PutSubscriptionFilter](#) la Referencia AWS SDK para C++ de la API.

CodeBuild ejemplos de uso de SDK para C++

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK para C++ with CodeBuild.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Acciones](#)

Acciones

ListBuilds

En el siguiente ejemplo de código, se muestra cómo utilizar ListBuilds.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
#!/ 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;
}
```

- Para obtener más información sobre la API, consulta [ListBuilds](#) la Referencia AWS SDK para C++ de la API.

ListProjects

En el siguiente ejemplo de código, se muestra cómo utilizar `ListProjects`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
//! 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;
}
```

- Para obtener más información sobre la API, consulta [ListProjects](#) la Referencia AWS SDK para C++ de la API.

StartBuild

En el siguiente ejemplo de código, se muestra cómo utilizar StartBuild.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener más información sobre la API, consulta [StartBuild](#) la Referencia AWS SDK para C++ de la API.

Ejemplos de proveedor de identidad de Amazon Cognito usando SDK para C++

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar situaciones comunes mediante el AWS SDK para C++ uso de Amazon Cognito Identity Provider.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Introducción](#)
- [Acciones](#)
- [Escenarios](#)

Introducción

Introducción a Amazon Cognito

En el siguiente ejemplo de código se muestra cómo empezar a utilizar Amazon Cognito.

SDK para C++

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Código para el CMake archivo 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})
```

Código del archivo de origen 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;
}
```

- Para obtener más información sobre la API, consulte [ListUserPools](#) la Referencia AWS SDK para C++ de la API.

Acciones

AdminGetUser

En el siguiente ejemplo de código, se muestra cómo utilizar AdminGetUser.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

    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;
    }
}

```

- Para obtener más información sobre la API, consulta [AdminGetUser](#) la Referencia AWS SDK para C++ de la API.

AdminInitiateAuth

En el siguiente ejemplo de código, se muestra cómo utilizar AdminInitiateAuth.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).


```
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;
}
}
```

- Para obtener más información sobre la API, consulta [AdminInitiateAuth](#) la Referencia AWS SDK para C++ de la API.

AdminRespondToAuthChallenge

En el siguiente ejemplo de código, se muestra cómo utilizar `AdminRespondToAuthChallenge`.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}

```

- Para obtener más información sobre la API, consulta [AdminRespondToAuthChallenge](#) la Referencia AWS SDK para C++ de la API.

AssociateSoftwareToken

En el siguiente ejemplo de código, se muestra cómo utilizar AssociateSoftwareToken.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

    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 << "\n0r 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;
}

```

- Para obtener más información sobre la API, consulta [AssociateSoftwareToken](#) la Referencia AWS SDK para C++ de la API.

ConfirmSignUp

En el siguiente ejemplo de código, se muestra cómo utilizar ConfirmSignUp.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

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;
}
```

- Para obtener más información sobre la API, consulta [ConfirmSignUp](#) la Referencia AWS SDK para C++ de la API.

DeleteUser

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteUser.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
    }
}

```

- Para obtener más información sobre la API, consulta [DeleteUser](#) la Referencia AWS SDK para C++ de la API.

ResendConfirmationCode

En el siguiente ejemplo de código, se muestra cómo utilizar ResendConfirmationCode.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

    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;
}
```

- Para obtener más información sobre la API, consulta [ResendConfirmationCode](#) la Referencia AWS SDK para C++ de la API.

SignUp

En el siguiente ejemplo de código, se muestra cómo utilizar SignUp.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
```

- Para obtener más información sobre la API, consulta [SignUp](#) la Referencia AWS SDK para C++ de la API.

VerifySoftwareToken

En el siguiente ejemplo de código, se muestra cómo utilizar `VerifySoftwareToken`.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
```

- Para obtener más información sobre la API, consulta [VerifySoftwareToken](#) la Referencia AWS SDK para C++ de la API.

Escenarios

Registro de un usuario en un grupo de usuarios que requiera MFA

En el siguiente ejemplo de código, se muestra cómo:

- Registre y confirme a un usuario con un nombre de usuario, una contraseña y una dirección de correo electrónico.
- Configure la autenticación multifactor asociando una aplicación MFA al usuario.
- Inicie sesión con una contraseña y un código MFA.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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();
}

```

```
}
```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para C++ .
 - [AdminGetUser](#)
 - [AdminInitiateAuth](#)
 - [AdminRespondToAuthChallenge](#)
 - [AssociateSoftwareToken](#)
 - [ConfirmDevice](#)
 - [ConfirmSignUp](#)
 - [InitiateAuth](#)
 - [ListUsers](#)
 - [ResendConfirmationCode](#)
 - [RespondToAuthChallenge](#)
 - [SignUp](#)
 - [VerifySoftwareToken](#)

Ejemplos de DynamoDB usando SDK para C++

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante DynamoDB. AWS SDK para C++

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Introducción](#)
- [Conceptos básicos](#)
- [Acciones](#)
- [Escenarios](#)

Introducción

Introducción a DynamoDB

En el siguiente ejemplo de código, se muestra cómo empezar a utilizar DynamoDB.

SDK para C++

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Código para el CMake archivo 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})

```

Código del archivo de código fuente `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;
}
```

- Para obtener más información sobre la API, consulte [ListTables](#) la Referencia AWS SDK para C++ de la API.

Conceptos básicos

Conceptos básicos

En el siguiente ejemplo de código, se muestra cómo:

- Creación de una tabla que pueda contener datos de películas.
- Colocar, obtener y actualizar una sola película en la tabla.
- Escribir los datos de películas en la tabla a partir de un archivo JSON de ejemplo.
- Consultar películas que se hayan estrenado en un año determinado.
- Buscar películas que se hayan estrenado en un intervalo de años.
- Eliminación de una película de la tabla y, a continuación, eliminar la tabla.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
{
    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::JsonValue> 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::JsonValue &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::JsonValue 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;
}
```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para C++ .
 - [BatchWriteItem](#)
 - [CreateTable](#)
 - [DeleteItem](#)
 - [DeleteTable](#)
 - [DescribeTable](#)
 - [GetItem](#)
 - [PutItem](#)
 - [Query](#)
 - [Scan](#)
 - [UpdateItem](#)

Acciones

BatchExecuteStatement

En el siguiente ejemplo de código, se muestra cómo utilizar BatchExecuteStatement.

SDK para C++

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Utilizar lotes de instrucciones INSERT para agregar elementos.

```
// 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;
    }
}

```

Utilizar lotes de instrucciones SELECT para obtener elementos.

```
// 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;
    }
}

```

Utilizar lotes de instrucciones UPDATE para actualizar elementos.

```

// 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;
}
}

```

Utilizar lotes de instrucciones DELETE para eliminar elementos.

```

// 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;
        }
    }
}

```

- Para obtener más información sobre la API, consulta [BatchExecuteStatement](#) la Referencia AWS SDK para C++ de la API.

BatchGetItem

En el siguiente ejemplo de código, se muestra cómo utilizar BatchGetItem.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

//! 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;
}
```

- Para obtener más información sobre la API, consulta [BatchGetItem](#) la Referencia AWS SDK para C++ de la API.

BatchWriteItem

En el siguiente ejemplo de código, se muestra cómo utilizar `BatchWriteItem`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
//! Batch write items from a JSON file.
/*!
 \sa batchWriteItem()
 \param jsonFilePath: JSON file path.
 \param clientConfiguration: AWS client configuration.
 \return bool: Function succeeded.
```

```

*/

/*
 * The input for this routine is a JSON file that you can download from the
 * following URL:
 * https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SampleData.html.
 *
 * The JSON data uses the BatchWriteItem API request syntax. The JSON strings are
 * converted to AttributeValue objects. These AttributeValue objects will then
 * generate
 * JSON strings when constructing the BatchWriteItem request, essentially outputting
 * their input.
 *
 * This is perhaps an artificial example, but it demonstrates the APIs.
 */

bool AwsDoc::DynamoDB::batchWriteItem(const Aws::String &jsonFilePath,
                                       const Aws::Client::ClientConfiguration
&clientConfiguration) {
    std::ifstream fileStream(jsonFilePath);

    if (!fileStream) {
        std::cerr << "Error: could not open file '" << jsonFilePath << "'."
        << std::endl;
    }

    std::stringstream stringStream;
    stringStream << fileStream.rdbuf();
    Aws::Utils::Json::JsonValue jsonValue(stringStream);

    Aws::DynamoDB::Model::BatchWriteItemRequest batchWriteItemRequest;
    Aws::Map<Aws::String, Aws::Utils::Json::JsonView> level1Map =
jsonValue.View().GetAllObjects();
    for (const auto &level1Entry: level1Map) {
        const Aws::Utils::Json::JsonView &entriesView = level1Entry.second;
        const Aws::String &tableName = level1Entry.first;
        // The JSON entries at this level are as follows:
        // key - table name
        // value - list of request objects
        if (!entriesView.IsListType()) {
            std::cerr << "Error: JSON file entry '"
                << tableName << "' is not a list." << std::endl;
            continue;
        }
    }
}

```

```

        Aws::Utils::Array<Aws::Utils::Json::JsonValue> entries =
entriesView.AsArray();

        Aws::Vector<Aws::DynamoDB::Model::WriteRequest> writeRequests;
        if (AwsDoc::DynamoDB::addWriteRequests(tableName, entries,
                                                writeRequests)) {
            batchWriteItemRequest.AddRequestItems(tableName, writeRequests);
        }
    }

    Aws::DynamoDB::DynamoDBClient dynamoClient(clientConfiguration);

    Aws::DynamoDB::Model::BatchWriteItemOutcome outcome =
dynamoClient.BatchWriteItem(
    batchWriteItemRequest);

    if (outcome.IsSuccess()) {
        std::cout << "DynamoDB::BatchWriteItem was successful." << std::endl;
    }
    else {
        std::cerr << "Error with DynamoDB::BatchWriteItem. "
        << outcome.GetError().GetMessage()
        << std::endl;
        return false;
    }

    return outcome.IsSuccess();
}

//! Convert requests in JSON format to a vector of WriteRequest objects.
/*!
    \sa addWriteRequests()
    \param tableName: Name of the table for the write operations.
    \param requestsJson: Request data in JSON format.
    \param writeRequests: Vector to receive the WriteRequest objects.
    \return bool: Function succeeded.
*/
bool AwsDoc::DynamoDB::addWriteRequests(const Aws::String &tableName,
                                        const
                                        Aws::Utils::Array<Aws::Utils::Json::JsonValue> &requestsJson,

                                        Aws::Vector<Aws::DynamoDB::Model::WriteRequest> &writeRequests) {
    for (size_t i = 0; i < requestsJson.GetLength(); ++i) {

```

```

const Aws::Utils::Json::JsonValue &requestsEntry = requestsJson[i];
if (!requestsEntry.IsObject()) {
    std::cerr << "Error: incorrect requestsEntry type "
                << requestsEntry.WriteReadable() << std::endl;
    return false;
}

    Aws::Map<Aws::String, Aws::Utils::Json::JsonValue> requestsMap =
requestsEntry.GetAllObjects();

    for (const auto &request: requestsMap) {
        const Aws::String &requestType = request.first;
        const Aws::Utils::Json::JsonValue &requestJsonView = request.second;

        if (requestType == "PutRequest") {
            if (!requestJsonView.ValueExists("Item")) {
                std::cerr << "Error: item key missing for requests "
                            << requestJsonView.WriteReadable() << std::endl;
                return false;
            }
            Aws::Map<Aws::String, Aws::DynamoDB::Model::AttributeValue>
attributes;
            if (!getAttributeObjectsMap(requestJsonView.GetObject("Item"),
attributes)) {
                std::cerr << "Error getting attributes "
                            << requestJsonView.WriteReadable() << std::endl;
                return false;
            }

            Aws::DynamoDB::Model::PutRequest putRequest;
            putRequest.SetItem(attributes);
            writeRequests.push_back(
                Aws::DynamoDB::Model::WriteRequest().WithPutRequest(
                    putRequest));
        }
        else {
            std::cerr << "Error: unimplemented request type '" << requestType
                    << "'." << std::endl;
        }
    }
}

return true;
}

```



```

//! Generate a map of AttributeValue objects from JSON records.
/*!
 \sa getAttributeObjectsMap()
 \param jsonView: JSONView of attribute records.
 \param writeRequests: Map to receive the AttributeValue objects.
 \return bool: Function succeeded.
 */
bool
AwsDoc::DynamoDB::getAttributeObjectsMap(const Aws::Utils::Json::JsonView &jsonView,
                                         Aws::Map<Aws::String,
                                         Aws::DynamoDB::Model::AttributeValue> &attributes) {
    Aws::Map<Aws::String, Aws::Utils::Json::JsonView> objectsMap =
    jsonView.GetAllObjects();
    for (const auto &entry: objectsMap) {
        const Aws::String &attributeKey = entry.first;
        const Aws::Utils::Json::JsonView &attributeJsonView = entry.second;

        if (!attributeJsonView.IsObject()) {
            std::cerr << "Error: attribute not an object "
                << attributeJsonView.WriteReadable() << std::endl;
            return false;
        }

        attributes.emplace(attributeKey,
                           Aws::DynamoDB::Model::AttributeValue(attributeJsonView));
    }

    return true;
}


```

- Para obtener más información sobre la API, consulta [BatchWriteItem](#) la Referencia AWS SDK para C++ de la API.

CreateTable

En el siguiente ejemplo de código, se muestra cómo utilizar CreateTable.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

```

Código que espera a que se active la tabla.

```

//! 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;
}
```

- Para obtener más información sobre la API, consulta [CreateTable](#) la Referencia AWS SDK para C++ de la API.

DeleteItem

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteItem.

SDK para C++

Note

Hay más información al respecto [en GitHub](#). Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

```

Código que espera a que se active la tabla.

```

//! 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;
}

```

- Para obtener más información sobre la API, consulta [DeleteItem](#) la Referencia AWS SDK para C++ de la API.

DeleteTable

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteTable.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener más información sobre la API, consulta [DeleteTable](#) la Referencia AWS SDK para C++ de la API.

DescribeTable

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeTable.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
#!/ 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();
}

```

- Para obtener más información sobre la API, consulta [DescribeTable](#) la Referencia AWS SDK para C++ de la API.

ExecuteStatement

En el siguiente ejemplo de código, se muestra cómo utilizar ExecuteStatement.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Utilizar una instrucción INSERT para agregar un elemento.

```

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;
    }
}

```

Utilizar una instrucción SELECT para obtener un elemento.

```

// 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;
        }
    }
}

```

Utilizar una instrucción UPDATE para actualizar un elemento.

```

// 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;
    }
}

```

Utilizar una instrucción DELETE para eliminar un elemento.

```

// 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;
    }
}
}

```

- Para obtener más información sobre la API, consulta [ExecuteStatement](#) la Referencia AWS SDK para C++ de la API.

GetItem

En el siguiente ejemplo de código, se muestra cómo utilizar `GetItem`.

SDK para C++

Note

Hay más información al respecto en [GitHub](#). Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
#!/ 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();
}

```

- Para obtener más información sobre la API, consulta [GetItem](#) la Referencia AWS SDK para C++ de la API.

ListTables

En el siguiente ejemplo de código, se muestra cómo utilizar ListTables.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

/*! 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;
}
```

- Para obtener más información sobre la API, consulta [ListTables](#) la Referencia AWS SDK para C++ de la API.

PutItem

En el siguiente ejemplo de código, se muestra cómo utilizar PutItem.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
//! 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);
}

```

Código que espera a que se active la tabla.

```

//! 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;
}
```

- Para obtener más información sobre la API, consulta [PutItem](#) la Referencia AWS SDK para C++ de la API.

Query

En el siguiente ejemplo de código, se muestra cómo utilizar Query.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
//! 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;
}

```

- Para obtener información sobre la API, consulte [Query](#) en la referencia de la API de AWS SDK para C++ .

Scan

En el siguiente ejemplo de código, se muestra cómo utilizar Scan.

SDK para C++

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
#!/ 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;
}

```

- Para obtener información sobre la API, consulte [Scan](#) en la referencia de la API de AWS SDK para C++ .

UpdateItem

En el siguiente ejemplo de código, se muestra cómo utilizar UpdateItem.

SDK para C++

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
#!/ 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);
}

```

Código que espera a que se active la tabla.

```

//! 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;
}

```

- Para obtener más información sobre la API, consulta [UpdateItem](#) la Referencia AWS SDK para C++ de la API.

UpdateTable

En el siguiente ejemplo de código, se muestra cómo utilizar UpdateTable.

SDK para C++

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

//! 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);
}

```

Código que espera a que se active la tabla.

```

/*! 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;
}
```

- Para obtener más información sobre la API, consulta [UpdateTable](#) la Referencia AWS SDK para C++ de la API.

Escenarios

Creación de una aplicación sin servidor para administrar fotos

En el siguiente ejemplo de código se muestra cómo crear una aplicación sin servidor que permita a los usuarios administrar fotos mediante etiquetas.

SDK para C++

Muestra cómo desarrollar una aplicación de administración de activos fotográficos que detecte las etiquetas de las imágenes mediante Amazon Rekognition y las almacene para su posterior recuperación.

Para obtener el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en [GitHub](#).

Para profundizar en el origen de este ejemplo, consulte la publicación en [Comunidad de AWS](#).

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Consultar una tabla mediante lotes de instrucciones PartiQL

En el siguiente ejemplo de código, se muestra cómo:

- Obtención de un lote de elementos mediante la ejecución de varias instrucciones SELECT.
- Agregar un lote de elementos mediante la ejecución de varias instrucciones INSERT.
- Actualizar un lote de elementos con la ejecución de varias instrucciones UPDATE.
- Eliminación de un lote de elementos con la ejecución de varias instrucciones DELETE.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
```


- Para obtener más información sobre la API, consulta [BatchExecuteStatement](#) la Referencia AWS SDK para C++ de la API.

Consultar una tabla con PartiQL

En el siguiente ejemplo de código, se muestra cómo:

- Obtener un artículo mediante una instrucción SELECT.
- Agregar un elemento mediante una instrucción INSERT.
- Actualizar un elemento mediante una instrucción UPDATE.
- Eliminar un elemento mediante una instrucción DELETE.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
// 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;
}
```

- Para obtener más información sobre la API, consulta [ExecuteStatement](#) la Referencia AWS SDK para C++ de la API.

EC2 Ejemplos de Amazon que utilizan SDK para C++

En los siguientes ejemplos de código, se muestra cómo realizar acciones e implementar situaciones comunes AWS SDK para C++ con Amazon EC2.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Introducción](#)
- [Acciones](#)

Introducción

Hola Amazon EC2

El siguiente ejemplo de código muestra cómo empezar a utilizar Amazon EC2.

SDK para C++

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Código para el CMake archivo 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})
```

Código del archivo de origen 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;
}

```


- Para obtener más información sobre la API, consulte [DescribeSecurityGroups](#) la Referencia AWS SDK para C++ de la API.

Acciones

AllocateAddress

En el siguiente ejemplo de código, se muestra cómo utilizar `AllocateAddress`.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

//! 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;
}

```

- Para obtener más información sobre la API, consulta [AllocateAddress](#) la Referencia AWS SDK para C++ de la API.

AssociateAddress

En el siguiente ejemplo de código, se muestra cómo utilizar AssociateAddress.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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();
}

```

- Para obtener más información sobre la API, consulta [AssociateAddress](#) la Referencia AWS SDK para C++ de la API.

AuthorizeSecurityGroupIngress

En el siguiente ejemplo de código, se muestra cómo utilizar AuthorizeSecurityGroupIngress.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

//! 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();
}

```

Función de utilidad para crear una regla de entrada.

```

//! 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);
}
```

- Para obtener más información sobre la API, consulta [AuthorizeSecurityGroupIngress](#) la Referencia AWS SDK para C++ de la API.

CreateKeyPair

En el siguiente ejemplo de código, se muestra cómo utilizar CreateKeyPair.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
//! 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();
}

```

- Para obtener más información sobre la API, consulta [CreateKeyPair](#) la Referencia AWS SDK para C++ de la API.

CreateSecurityGroup

En el siguiente ejemplo de código, se muestra cómo utilizar CreateSecurityGroup.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

//! 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;
}

```

- Para obtener más información sobre la API, consulta [CreateSecurityGroup](#) la Referencia AWS SDK para C++ de la API.

CreateTags

En el siguiente ejemplo de código, se muestra cómo utilizar CreateTags.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener más información sobre la API, consulta [CreateTags](#) la Referencia AWS SDK para C++ de la API.

DeleteKeyPair

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteKeyPair.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
#!/ 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();
}

```

- Para obtener más información sobre la API, consulta [DeleteKeyPair](#) la Referencia AWS SDK para C++ de la API.

DeleteSecurityGroup

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteSecurityGroup.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

//! 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();
}

```

- Para obtener más información sobre la API, consulta [DeleteSecurityGroup](#) la Referencia AWS SDK para C++ de la API.

DescribeAddresses

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeAddresses.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

//! 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();
}

```

- Para obtener más información sobre la API, consulta [DescribeAddresses](#) la Referencia AWS SDK para C++ de la API.

DescribeAvailabilityZones

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeAvailabilityZones.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

//! 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();
}


```

- Para obtener más información sobre la API, consulta [DescribeAvailabilityZones](#) la Referencia AWS SDK para C++ de la API.

DescribeInstances

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeInstances.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
#!/ 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;
}

```

- Para obtener más información sobre la API, consulta [DescribeInstances](#) la Referencia AWS SDK para C++ de la API.

DescribeKeyPairs

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeKeyPairs.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

/*! 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();
}

```

- Para obtener más información sobre la API, consulta [DescribeKeyPairs](#) la Referencia AWS SDK para C++ de la API.

DescribeRegions

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeRegions.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

//! 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();
}
```

- Para obtener más información sobre la API, consulta [DescribeRegions](#) la Referencia AWS SDK para C++ de la API.

DescribeSecurityGroups

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeSecurityGroups.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).


```

//! 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;
}

```

- Para obtener más información sobre la API, consulta [DescribeSecurityGroups](#) la Referencia AWS SDK para C++ de la API.

MonitorInstances

En el siguiente ejemplo de código, se muestra cómo utilizar `MonitorInstances`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

/*! 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();
}


```

- Para obtener más información sobre la API, consulta [MonitorInstances](#) la Referencia AWS SDK para C++ de la API.

RebootInstances

En el siguiente ejemplo de código, se muestra cómo utilizar RebootInstances.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
#!/ 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();
}

```

- Para obtener más información sobre la API, consulta [RebootInstances](#) la Referencia AWS SDK para C++ de la API.

ReleaseAddress

En el siguiente ejemplo de código, se muestra cómo utilizar `ReleaseAddress`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

//! 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();
}

```

- Para obtener más información sobre la API, consulta [ReleaseAddress](#) la Referencia AWS SDK para C++ de la API.

RunInstances

En el siguiente ejemplo de código, se muestra cómo utilizar RunInstances.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

/*! 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;
}
```

- Para obtener más información sobre la API, consulta [RunInstances](#) la Referencia AWS SDK para C++ de la API.

StartInstances

En el siguiente ejemplo de código, se muestra cómo utilizar StartInstances.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
#!/ 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();
}

```

- Para obtener más información sobre la API, consulta [StartInstances](#) la Referencia AWS SDK para C++ de la API.

StopInstances

En el siguiente ejemplo de código, se muestra cómo utilizar StopInstances.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

//! 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;
}


```

- Para obtener más información sobre la API, consulta [StopInstances](#) la Referencia AWS SDK para C++ de la API.

TerminateInstances

En el siguiente ejemplo de código, se muestra cómo utilizar TerminateInstances.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener más información sobre la API, consulta [TerminateInstances](#) la Referencia AWS SDK para C++ de la API.

UnmonitorInstances

En el siguiente ejemplo de código, se muestra cómo utilizar `UnmonitorInstances`.

SDK para C++

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener más información sobre la API, consulta [UnmonitorInstances](#) la Referencia AWS SDK para C++ de la API.

EventBridge ejemplos de uso de SDK para C++

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK para C++ with EventBridge.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas


- [Acciones](#)

Acciones

PutEvents

En el siguiente ejemplo de código, se muestra cómo utilizar PutEvents.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Incluir los archivos requeridos.

```
#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>
```

Enviar el evento.

```
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;
}
```

- Para obtener más información sobre la API, consulta [PutEvents](#) la Referencia AWS SDK para C++ de la API.

PutRule

En el siguiente ejemplo de código, se muestra cómo utilizar PutRule.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Incluir los archivos requeridos.

```
#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>
```

Crear la regla.

```
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;
}
```

- Para obtener más información sobre la API, consulta [PutRule](#) la Referencia AWS SDK para C++ de la API.

PutTargets

En el siguiente ejemplo de código, se muestra cómo utilizar PutTargets.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Incluir los archivos requeridos.

```
#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>
```

Añada el destino.

```
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;
}
```

- Para obtener más información sobre la API, consulta [PutTargets](#) la Referencia AWS SDK para C++ de la API.

AWS Glue ejemplos de uso de SDK para C++

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK para C++ with AWS Glue.

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Introducción](#)
- [Conceptos básicos](#)

- [Acciones](#)

Introducción

Hola AWS Glue

En el siguiente ejemplo de código se muestra cómo empezar a utilizar AWS Glue.

SDK para C++

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Código para el CMake archivo 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})
```

Código del archivo de origen 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;
}
```

- Para obtener más información sobre la API, consulte [ListJobs](#) la Referencia AWS SDK para C++ de la API.

Conceptos básicos

Conceptos básicos

En el siguiente ejemplo de código, se muestra cómo:

- Crear un rastreador que rastree un bucket de Amazon S3 público y generar una base de datos de metadatos con formato CSV.
- Incluya información sobre bases de datos y tablas en su AWS Glue Data Catalog.
- Crear un trabajo para extraer datos CSV del bucket de S3, transformar los datos y cargar el resultado con formato JSON en otro bucket de S3.
- Incluir información sobre las ejecuciones de trabajos, ver algunos de los datos transformados y limpiar los recursos.

Para obtener más información, consulte el [tutorial: Introducción a AWS Glue Studio](#).

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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


- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para C++ .
 - [CreateCrawler](#)
 - [CreateJob](#)
 - [DeleteCrawler](#)
 - [DeleteDatabase](#)
 - [DeleteJob](#)
 - [DeleteTable](#)
 - [GetCrawler](#)
 - [GetDatabase](#)
 - [GetDatabases](#)
 - [GetJob](#)
 - [GetJobRun](#)
 - [GetJobRuns](#)
 - [GetTables](#)
 - [ListJobs](#)
 - [StartCrawler](#)
 - [StartJobRun](#)

Acciones

CreateCrawler

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateCrawler`.

SDK para C++

 Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
```


- Para obtener más información sobre la API, consulta [CreateCrawler](#) la Referencia AWS SDK para C++ de la API.

CreateJob

En el siguiente ejemplo de código, se muestra cómo utilizar CreateJob.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
    }
```

- Para obtener más información sobre la API, consulta [CreateJob](#) la Referencia AWS SDK para C++ de la API.

DeleteCrawler

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteCrawler.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
    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;
    }
```

- Para obtener más información sobre la API, consulta [DeleteCrawler](#) la Referencia AWS SDK para C++ de la API.

DeleteDatabase

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteDatabase.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
```

- Para obtener más información sobre la API, consulta [DeleteDatabase](#) la Referencia AWS SDK para C++ de la API.

DeleteJob

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteJob.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
```

- Para obtener más información sobre la API, consulta [DeleteJob](#) la Referencia AWS SDK para C++ de la API.

GetCrawler

En el siguiente ejemplo de código, se muestra cómo utilizar `GetCrawler`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
```

- Para obtener más información sobre la API, consulta [GetCrawler](#) la Referencia AWS SDK para C++ de la API.

GetDatabase

En el siguiente ejemplo de código, se muestra cómo utilizar GetDatabase.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}

```

- Para obtener más información sobre la API, consulta [GetDatabase](#) la Referencia AWS SDK para C++ de la API.

GetJobRun

En el siguiente ejemplo de código, se muestra cómo utilizar GetJobRun.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

    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;
}
```

- Para obtener más información sobre la API, consulta [GetJobRun](#) la Referencia AWS SDK para C++ de la API.

GetJobRuns

En el siguiente ejemplo de código, se muestra cómo utilizar GetJobRuns.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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());
```

- Para obtener más información sobre la API, consulta [GetJobRuns](#) la Referencia AWS SDK para C++ de la API.

GetTables

En el siguiente ejemplo de código, se muestra cómo utilizar GetTables.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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();
    }

```

- Para obtener más información sobre la API, consulta [GetTables](#) la Referencia AWS SDK para C++ de la API.

ListJobs

En el siguiente ejemplo de código, se muestra cómo utilizar ListJobs.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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());
```

- Para obtener más información sobre la API, consulta [ListJobs](#) la Referencia AWS SDK para C++ de la API.

StartCrawler

En el siguiente ejemplo de código, se muestra cómo utilizar StartCrawler.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
    }

```

- Para obtener más información sobre la API, consulta [StartCrawler](#) la Referencia AWS SDK para C++ de la API.

StartJobRun

En el siguiente ejemplo de código, se muestra cómo utilizar StartJobRun.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

    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;
}
```

- Para obtener más información sobre la API, consulta [StartJobRun](#) la Referencia AWS SDK para C++ de la API.

HealthImaging ejemplos de uso de SDK para C++

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK para C++ with HealthImaging.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Introducción](#)
- [Acciones](#)
- [Escenarios](#)

Introducción

Hola HealthImaging

En el siguiente ejemplo de código se muestra cómo empezar a utilizar HealthImaging.

SDK para C++

Código para el CMake archivo 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})
```

Código del archivo de origen `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;
}

```

- Para obtener más información sobre la API, consulte [ListDatastores](#) la Referencia AWS SDK para C++ de la API.

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Acciones

DeleteImageSet

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteImageSet.

SDK para 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();
}

```

- Para obtener más información sobre la API, consulta [DeleteImageSet](#) la Referencia AWS SDK para C++ de la API.

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

GetDICOMImportJob

En el siguiente ejemplo de código, se muestra cómo utilizar GetDICOMImportJob.

SDK para 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;
}

```

- Para obtener información sobre la API, consulta [Get DICOMImport Job](#) in AWS SDK para C++ API Reference.

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

GetImageFrame

En el siguiente ejemplo de código, se muestra cómo utilizar GetImageFrame.

SDK para 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();
}
```

- Para obtener más información sobre la API, consulta [GetImageFrame](#) la Referencia AWS SDK para C++ de la API.

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

GetImageSetMetadata

En el siguiente ejemplo de código, se muestra cómo utilizar GetImageSetMetadata.

SDK para C++

Función de utilidad para obtener metadatos del conjunto de imágenes.

```

//! Routine which gets a HealthImaging image set's metadata.
/*!
 \param dataStoreID: The HealthImaging data store ID.
 \param imageSetID: The HealthImaging image set ID.
 \param versionID: The HealthImaging image set version ID, ignored if empty.
 \param outputPath: The path where the metadata will be stored as gzipped json.
 \param clientConfig: Aws client configuration.
 \\return bool: Function succeeded.
 */
bool AwsDoc::Medical_Imaging::getImageSetMetadata(const Aws::String &dataStoreID,
                                                  const Aws::String &imageSetID,
                                                  const Aws::String &versionID,
                                                  const Aws::String &outputFilePath,
                                                  const
                                                  Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::Model::GetImageSetMetadataRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);
    if (!versionID.empty()) {
        request.SetVersionId(versionID);
    }
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::GetImageSetMetadataOutcome outcome =
    client.GetImageSetMetadata(

```



```
        request);
    if (outcome.IsSuccess()) {
        std::ofstream file(outputFilePath, std::ios::binary);
        auto &metadata = outcome.GetResult().GetImageSetMetadataBlob();
        file << metadata.rdbuf();
    }
    else {
        std::cerr << "Failed to get image set metadata: "
            << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

Obtener metadatos del conjunto de imágenes sin versión.

```
    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;
    }
```

Obtener metadatos del conjunto de imágenes con la versión.

```
    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;
    }
```

- Para obtener más información sobre la API, consulta [GetImageSetMetadata](#) la Referencia AWS SDK para C++ de la API.

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

SearchImageSets

En el siguiente ejemplo de código, se muestra cómo utilizar SearchImageSets.

SDK para C++

La función de utilidad para buscar conjuntos de imágenes.

```

//! 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;
}

```

Caso de uso núm. 1: operador IGUAL.

```

    Aws::Vector<Aws::String> imageIDsForPatientID;
    Aws::MedicalImaging::Model::SearchCriteria searchCriteriaEqualsPatientID;
    Aws::Vector<Aws::MedicalImaging::Model::SearchFilter> patientIDSearchFilters
= {

    Aws::MedicalImaging::Model::SearchFilter().WithOperator(Aws::MedicalImaging::Model::Operator
    .WithValues({Aws::MedicalImaging::Model::SearchByAttributeValue().WithDICOMPatientId(patient
        });

    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;

```

```

    }
}

```

Caso de uso #2: el operador BETWEEN usa DICOMStudy fecha y DICOMStudy hora.

```

    Aws::MedicalImaging::Model::SearchByAttributeValue useCase2StartDate;

    useCase2StartDate.SetDICOMStudyDateAndTime(Aws::MedicalImaging::Model::DICOMStudyDateAndTime(
        .WithDICOMStudyDate("19990101")
        .WithDICOMStudyTime("000000.000"));

    Aws::MedicalImaging::Model::SearchByAttributeValue useCase2EndDate;

    useCase2EndDate.SetDICOMStudyDateAndTime(Aws::MedicalImaging::Model::DICOMStudyDateAndTime(
        .WithDICOMStudyDate(Aws::Utils::DateTime(std::chrono::system_clock::now()).ToLocalTimeStrin
        "%m%d"))
        .WithDICOMStudyTime("000000.000"));

    Aws::MedicalImaging::Model::SearchFilter useCase2SearchFilter;
    useCase2SearchFilter.SetValues({useCase2StartDate, useCase2EndDate});

    useCase2SearchFilter.SetOperator(Aws::MedicalImaging::Model::Operator::BETWEEN);

    Aws::MedicalImaging::Model::SearchCriteria useCase2SearchCriteria;
    useCase2SearchCriteria.SetFilters({useCase2SearchFilter});

    Aws::Vector<Aws::String> usesCase2Results;
    result = AwsDoc::Medical_Imaging::searchImageSets(dataStoreID,
                                                        useCase2SearchCriteria,
                                                        usesCase2Results,
                                                        clientConfig);

    if (result) {
        std::cout << usesCase2Results.size() << " image sets found for between
1999/01/01 and present."
                << std::endl;
        for (auto &imageSetResult : usesCase2Results) {
            std::cout << " Image set with ID '" << imageSetResult << std::endl;
        }
    }
}

```

Caso de uso núm. 3: el operador ENTRE usa CreatedAt. Los estudios de tiempo se habían mantenido previamente.

```

    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;
        }
    }
}

```

Caso de uso #4: operador EQUAL en DICOMSeries InstanceUID y BETWEEN en UpdatedAt y ordena la respuesta en orden ASC en el campo 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;
        }
    }
}

```

- Para obtener más información sobre la API, consulta la Referencia de la API. [SearchImageSets](#) AWS SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

StartDICOMImportJob

En el siguiente ejemplo de código, se muestra cómo utilizar `StartDICOMImportJob`.

SDK para 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();
}
```

- Para obtener más información sobre la API, consulta [Start DICOMImport Job](#) in AWS SDK para C++ API Reference.

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Escenarios

Introducción a los conjuntos y marcos de imágenes

El siguiente ejemplo de código muestra cómo importar archivos DICOM y descargar marcos de imágenes en HealthImaging ellos.

La implementación está estructurada como una aplicación de línea de comandos.

- Configure los recursos para una tarea de importación DICOM.
- Importe los archivos DICOM en un almacén de datos.

- Recupera el conjunto de imágenes IDs para el trabajo de importación.
- Recupere el marco de imágenes IDs de los conjuntos de imágenes.
- Descargue, decodifique y verifique los marcos de imágenes.
- Eliminación de recursos.

SDK para C++

Cree una CloudFormation pila con los recursos necesarios.

```
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: ");
    }
}

```

Copie los archivos DICOM en el bucket de importación de Amazon S3.

```

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;
}

```

Importe los archivos DICOM en el almacén de datos de Amazon S3.

```

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;
}

```

Obtenga los conjuntos de imágenes que ha creado el trabajo de importación DICOM.

```

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;
}

```

Obtenga información sobre los marcos de imágenes para los conjuntos de imágenes.

```

bool AwsDoc::Medical_Imaging::getImageFramesForImageSet(const Aws::String
&dataStoreID,
                                                    const Aws::String
&imageSetID,
                                                    const Aws::String
&outDirectory,
                                                    Aws::Vector<ImageFrameInfo>
&imageFrames,
                                                    const
Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::String fileName = outDirectory + "/" + imageSetID + "_metadata.json.gzip";
    bool result = false;
    if (getImageSetMetadata(dataStoreID, imageSetID, "", // Empty string for version
ID.
                                fileName, clientConfiguration)) {
        try {
            std::string metadataGZip;
            {
                std::ifstream inFileStream(fileName.c_str(), std::ios::binary);
                if (!inFileStream) {

```



```

        throw std::runtime_error("Failed to open file " + fileName);
    }

    std::stringstream stringStream;
    stringstream << inFileStream.rdbuf();
    metadataGZip = stringStream.str();
}
std::string metadataJson = gzip::decompress(metadataGZip.data(),
                                           metadataGZip.size());

// Use JMESPath to extract the image set IDs.
// https://jmespath.org/specification.html
jsoncons::json doc = jsoncons::json::parse(metadataJson);
std::string jmesPathExpression = "Study.Series.*.Instances[*.*]";
jsoncons::json instances = jsoncons::jmespath::search(doc,
jmesPathExpression);
    for (auto &instance: instances.array_range()) {
        jmesPathExpression = "DICOM.RescaleSlope";
        std::string rescaleSlope = jsoncons::jmespath::search(instance,
jmesPathExpression).to_string();
        jmesPathExpression = "DICOM.RescaleIntercept";
        std::string rescaleIntercept = jsoncons::jmespath::search(instance,
jmesPathExpression).to_string();

        jmesPathExpression = "ImageFrames[*][*]";
        jsoncons::json imageFramesJson =
jsoncons::jmespath::search(instance,
jmesPathExpression);

        for (auto &imageFrame: imageFramesJson.array_range()) {
            ImageFrameInfo imageFrameIDs;
            imageFrameIDs.mImageSetId = imageSetID;
            imageFrameIDs.mImageFrameId = imageFrame.find(
                "ID")->value().as_string();
            imageFrameIDs.mRescaleIntercept = rescaleIntercept;
            imageFrameIDs.mRescaleSlope = rescaleSlope;
            imageFrameIDs.MinPixelValue = imageFrame.find(
                "MinPixelValue")->value().as_string();
            imageFrameIDs.MaxPixelValue = imageFrame.find(
                "MaxPixelValue")->value().as_string();
        }
    }
}

```

```

        jmesPathExpression =
"max_by(PixelDataChecksumFromBaseToFullResolution, &Width).Checksum";
        jsoncons::json checksumJson =
jsoncons::jmespath::search(imageFrame,

jmesPathExpression);
        imageFrameIDs.mFullResolutionChecksum =
checksumJson.as_integer<uint32_t>();

        imageFrames.emplace_back(imageFrameIDs);
    }
}

    result = true;
}
catch (const std::exception &e) {
    std::cerr << "getImageFramesForImageSet failed because " << e.what()
        << std::endl;
}
}

return result;
}

//! Routine which gets a HealthImaging image set's metadata.
/*!
 \param dataStoreID: The HealthImaging data store ID.
 \param imageSetID: The HealthImaging image set ID.
 \param versionID: The HealthImaging image set version ID, ignored if empty.
 \param outputPath: The path where the metadata will be stored as gzipped json.
 \param clientConfig: Aws client configuration.
 \return bool: Function succeeded.
*/
bool AwsDoc::Medical_Imaging::getImageSetMetadata(const Aws::String &dataStoreID,
                                                const Aws::String &imageSetID,
                                                const Aws::String &versionID,
                                                const Aws::String &outputFilePath,
                                                const
Aws::Client::ClientConfiguration &clientConfig) {
    Aws::MedicalImaging::Model::GetImageSetMetadataRequest request;
    request.SetDatastoreId(dataStoreID);
    request.SetImageSetId(imageSetID);
    if (!versionID.empty()) {
        request.SetVersionId(versionID);
    }
}

```

```

    }
    Aws::MedicalImaging::MedicalImagingClient client(clientConfig);
    Aws::MedicalImaging::Model::GetImageSetMetadataOutcome outcome =
client.GetImageSetMetadata(
    request);
    if (outcome.IsSuccess()) {
        std::ofstream file(outputFilePath, std::ios::binary);
        auto &metadata = outcome.GetResult().GetImageSetMetadataBlob();
        file << metadata.rdbuf();
    }
    else {
        std::cerr << "Failed to get image set metadata: "
            << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}

```

Descargar, decodificar y verificar los fotogramas de imágenes.

```

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;

```

```
}

```

Eliminación de recursos.

```
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;
}

```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para C++ .

- [DeleteImageSet](#)
- [Consigue un DICOMImport trabajo](#)
- [GetImageFrame](#)
- [GetImageSetMetadata](#)
- [SearchImageSets](#)
- [Iniciar DICOMImport trabajo](#)

 Note

Hay más en marcha GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Ejemplos de IAM usando SDK para C++

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso de AWS SDK para C++ IAM.

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Introducción](#)
- [Conceptos básicos](#)
- [Acciones](#)

Introducción

Introducción a IAM

El siguiente ejemplo de código muestra cómo empezar a utilizar IAM.

SDK para C++

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Código para el CMake archivo 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})

```

Código del archivo de origen iam.cpp.

```

#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;
}
```

- Para obtener más información sobre la API, consulte [ListPolicies](#) la Referencia AWS SDK para C++ de la API.

Conceptos básicos

Conceptos básicos


En el siguiente ejemplo de código, se muestra cómo crear un usuario y asumir un rol.

Warning

Para evitar riesgos de seguridad, no utilice a los usuarios de IAM para la autenticación cuando desarrolle software especialmente diseñado o trabaje con datos reales. En cambio, utilice la federación con un proveedor de identidades como [AWS IAM Identity Center](#).

- Crear un usuario que no tenga permisos.
- Crear un rol que conceda permiso para enumerar los buckets de Amazon S3 para la cuenta.
- Agregar una política para que el usuario asuma el rol.
- Asumir el rol y enumerar los buckets de S3 con credenciales temporales, y después limpiar los recursos.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
```


- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para C++ .
 - [AttachRolePolicy](#)
 - [CreateAccessKey](#)
 - [CreatePolicy](#)
 - [CreateRole](#)
 - [CreateUser](#)
 - [DeleteAccessKey](#)
 - [DeletePolicy](#)
 - [DeleteRole](#)
 - [DeleteUser](#)
 - [DeleteUserPolicy](#)
 - [DetachRolePolicy](#)
 - [PutUserPolicy](#)

Acciones

AttachRolePolicy

En el siguiente ejemplo de código, se muestra cómo utilizar `AttachRolePolicy`.

SDK para C++

 Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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();
}
```

- Para obtener más información sobre la API, consulta [AttachRolePolicy](#) la Referencia AWS SDK para C++ de la API.

CreateAccessKey

En el siguiente ejemplo de código, se muestra cómo utilizar CreateAccessKey.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}

```

- Para obtener más información sobre la API, consulta [CreateAccessKey](#) la Referencia AWS SDK para C++ de la API.

CreateAccountAlias

En el siguiente ejemplo de código, se muestra cómo utilizar CreateAccountAlias.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

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();
}
```

- Para obtener más información sobre la API, consulta [CreateAccountAlias](#) la Referencia AWS SDK para C++ de la API.

CreatePolicy

En el siguiente ejemplo de código, se muestra cómo utilizar CreatePolicy.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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();
}

```

```
}
```

- Para obtener más información sobre la API, consulta [CreatePolicy](#) la Referencia AWS SDK para C++ de la API.

CreateRole

En el siguiente ejemplo de código, se muestra cómo utilizar CreateRole.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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();
}
```

```
}
```

- Para obtener más información sobre la API, consulta [CreateRole](#) la Referencia AWS SDK para C++ de la API.

CreateUser

En el siguiente ejemplo de código, se muestra cómo utilizar CreateUser.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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();
```

- Para obtener más información sobre la API, consulta [CreateUser](#) la Referencia AWS SDK para C++ de la API.

DeleteAccessKey

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteAccessKey.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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();
}
```

- Para obtener más información sobre la API, consulta [DeleteAccessKey](#) la Referencia AWS SDK para C++ de la API.

DeleteAccountAlias

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteAccountAlias.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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();
}
```

- Para obtener más información sobre la API, consulta [DeleteAccountAlias](#) la Referencia AWS SDK para C++ de la API.

DeletePolicy

En el siguiente ejemplo de código, se muestra cómo utilizar DeletePolicy.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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();
}
```

- Para obtener más información sobre la API, consulta [DeletePolicy](#) la Referencia AWS SDK para C++ de la API.

DeleteServerCertificate

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteServerCertificate.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
```

- Para obtener más información sobre la API, consulta [DeleteServerCertificate](#) la Referencia AWS SDK para C++ de la API.

DeleteUser

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteUser.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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();
```

- Para obtener más información sobre la API, consulta [DeleteUser](#) la Referencia AWS SDK para C++ de la API.

DetachRolePolicy

En el siguiente ejemplo de código, se muestra cómo utilizar DetachRolePolicy.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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();
```

- Para obtener más información sobre la API, consulta [DetachRolePolicy](#) la Referencia AWS SDK para C++ de la API.

GetAccessKeyLastUsed

En el siguiente ejemplo de código, se muestra cómo utilizar `GetAccessKeyLastUsed`.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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();
}
```

- Para obtener más información sobre la API, consulta [GetAccessKeyLastUsed](#) la Referencia AWS SDK para C++ de la API.

GetPolicy

En el siguiente ejemplo de código, se muestra cómo utilizar GetPolicy.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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();
}
```

- Para obtener más información sobre la API, consulta [GetPolicy](#) la Referencia AWS SDK para C++ de la API.

GetServerCertificate

En el siguiente ejemplo de código, se muestra cómo utilizar `GetServerCertificate`.

SDK para C++

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
```

- Para obtener más información sobre la API, consulta [GetServerCertificate](#) la Referencia AWS SDK para C++ de la API.

ListAccessKeys

En el siguiente ejemplo de código, se muestra cómo utilizar `ListAccessKeys`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
```

- Para obtener más información sobre la API, consulta [ListAccessKeys](#) la Referencia AWS SDK para C++ de la API.

ListAccountAliases

En el siguiente ejemplo de código, se muestra cómo utilizar ListAccountAliases.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
```

- Para obtener más información sobre la API, consulta [ListAccountAliases](#) la Referencia AWS SDK para C++ de la API.

ListPolicies

En el siguiente ejemplo de código, se muestra cómo utilizar `ListPolicies`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}

```

- Para obtener más información sobre la API, consulta [ListPolicies](#) la Referencia AWS SDK para C++ de la API.

ListServerCertificates

En el siguiente ejemplo de código, se muestra cómo utilizar ListServerCertificates.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

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;
}
```

- Para obtener más información sobre la API, consulta [ListServerCertificates](#) la Referencia AWS SDK para C++ de la API.

ListUsers

En el siguiente ejemplo de código, se muestra cómo utilizar ListUsers.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}

```

- Para obtener más información sobre la API, consulta [ListUsers](#) la Referencia AWS SDK para C++ de la API.

PutRolePolicy

En el siguiente ejemplo de código, se muestra cómo utilizar PutRolePolicy.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

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();
}

```

- Para obtener más información sobre la API, consulta [PutRolePolicy](#) la Referencia AWS SDK para C++ de la API.

UpdateAccessKey

En el siguiente ejemplo de código, se muestra cómo utilizar UpdateAccessKey.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

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();
}
```

- Para obtener más información sobre la API, consulta [UpdateAccessKey](#) la Referencia AWS SDK para C++ de la API.

UpdateServerCertificate

En el siguiente ejemplo de código, se muestra cómo utilizar UpdateServerCertificate.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
```

- Para obtener más información sobre la API, consulta [UpdateServerCertificate](#) la Referencia AWS SDK para C++ de la API.

UpdateUser

En el siguiente ejemplo de código, se muestra cómo utilizar UpdateUser.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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();
}
```

- Para obtener más información sobre la API, consulta [UpdateUser](#) la Referencia AWS SDK para C++ de la API.

AWS IoT ejemplos de uso de SDK para C++

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK para C++ with AWS IoT.

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Introducción](#)
- [Conceptos básicos](#)
- [Acciones](#)

Introducción

Hola AWS IoT

En el siguiente ejemplo de código se muestra cómo empezar a utilizar AWS IoT.

SDK para C++

Código para el CMake archivo 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})

```

Código del archivo de origen 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;
}
```


- Para obtener información sobre la API, consulte [listThings](#) en la Referencia de la API de AWS SDK para C++ .

 Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).


Conceptos básicos

Conceptos básicos

En el siguiente ejemplo de código, se muestra cómo:

- Crea una AWS IoT cosa.
- Generar un certificado de dispositivo.
- Actualiza AWS IoT cualquier cosa con atributos.
- Devolver un punto de conexión único.
- Enumere sus AWS IoT certificados.
- Actualiza y AWS IoT oculta.
- Escribir la información de estado.
- Crear una regla.
- Mostrar las reglas.
- Buscar objetos con el nombre de objeto.
- Elimina cualquier AWS IoT cosa.

SDK para C++

 Note

Hay más en marcha GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Crea cualquier AWS IoT cosa.

```

    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();
}

```

Genere y asocie un certificado de dispositivo.

```

    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();
}

```

Realiza varias operaciones en la AWS IoT cosa.

```

    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 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;
}
```

Eliminación de recursos.

```
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();  
}
```

Acciones

AttachThingPrincipal

En el siguiente ejemplo de código, se muestra cómo utilizar `AttachThingPrincipal`.

SDK para C++

 Note

Hay más en marcha GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener más información sobre la API, consulta [AttachThingPrincipal](#) la Referencia AWS SDK para C++ de la API.

CreateKeysAndCertificate

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateKeysAndCertificate`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

/*! 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();
}

```

- Para obtener más información sobre la API, consulta [CreateKeysAndCertificate](#) la Referencia AWS SDK para C++ de la API.

CreateThing

En el siguiente ejemplo de código, se muestra cómo utilizar CreateThing.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

//! 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();
}

```

- Para obtener más información sobre la API, consulta [CreateThing](#) la Referencia AWS SDK para C++ de la API.

CreateTopicRule

En el siguiente ejemplo de código, se muestra cómo utilizar CreateTopicRule.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

/*! 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();
}
```

- Para obtener más información sobre la API, consulta [CreateTopicRule](#) la Referencia AWS SDK para C++ de la API.

DeleteCertificate

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteCertificate.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener más información sobre la API, consulta [DeleteCertificate](#) la Referencia AWS SDK para C++ de la API.

DeleteThing

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteThing.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener más información sobre la API, consulta [DeleteThing](#) la Referencia AWS SDK para C++ de la API.

DeleteTopicRule

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteTopicRule.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

//! 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();
}


```

- Para obtener más información sobre la API, consulta [DeleteTopicRule](#) la Referencia AWS SDK para C++ de la API.

DescribeEndpoint

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeEndpoint.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

//! 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();
}

```

- Para obtener más información sobre la API, consulta [DescribeEndpoint](#) la Referencia AWS SDK para C++ de la API.

DescribeThing

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeThing.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
#!/ 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();
}

```

- Para obtener más información sobre la API, consulta [DescribeThing](#) la Referencia AWS SDK para C++ de la API.

DetachThingPrincipal

En el siguiente ejemplo de código, se muestra cómo utilizar `DetachThingPrincipal`.

SDK para C++

Note

Hay más información al respecto [en GitHub](#). Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

//! 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();
}

```

- Para obtener más información sobre la API, consulta [DetachThingPrincipal](#) la Referencia AWS SDK para C++ de la API.

ListCertificates

En el siguiente ejemplo de código, se muestra cómo utilizar ListCertificates.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

/*! 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;
}
```

- Para obtener más información sobre la API, consulta [ListCertificates](#) la Referencia AWS SDK para C++ de la API.

SearchIndex

En el siguiente ejemplo de código, se muestra cómo utilizar SearchIndex.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
#!/ 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;
}

```

- Para obtener más información sobre la API, consulta [SearchIndex](#) la Referencia AWS SDK para C++ de la API.

UpdateIndexingConfiguration

En el siguiente ejemplo de código, se muestra cómo utilizar UpdateIndexingConfiguration.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

//! 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();
}

```

- Para obtener más información sobre la API, consulta [UpdateIndexingConfiguration](#) la Referencia AWS SDK para C++ de la API.

UpdateThing

En el siguiente ejemplo de código, se muestra cómo utilizar UpdateThing.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

```
}
```

- Para obtener más información sobre la API, consulta [UpdateThing](#) la Referencia AWS SDK para C++ de la API.

AWS IoT data ejemplos de uso de SDK para C++

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK para C++ with AWS IoT data.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Acciones](#)

Acciones

GetThingShadow

En el siguiente ejemplo de código, se muestra cómo utilizar GetThingShadow.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
//! 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();
}

```

- Para obtener más información sobre la API, consulta [GetThingShadow](#) la Referencia AWS SDK para C++ de la API.

UpdateThingShadow

En el siguiente ejemplo de código, se muestra cómo utilizar UpdateThingShadow.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

//! 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();
}

```

- Para obtener más información sobre la API, consulta [UpdateThingShadow](#) la Referencia AWS SDK para C++ de la API.

Ejemplos de Lambda usando SDK para C++

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK para C++ mediante Lambda.

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Introducción](#)
- [Conceptos básicos](#)
- [Acciones](#)
- [Escenarios](#)

Introducción

Introducción a Lambda

El siguiente ejemplo de código muestra cómo empezar a utilizar Lambda.

SDK para C++

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Código para el CMake archivo 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})
```

Código del archivo de origen 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;
}

```

- Para obtener más información sobre la API, consulte [ListFunctions](#) la Referencia AWS SDK para C++ de la API.

Conceptos básicos

Conceptos básicos

En el siguiente ejemplo de código, se muestra cómo:

- Crear un rol de IAM y una función de Lambda y, a continuación, cargar el código de controlador.
- Invocar la función con un único parámetro y obtener resultados.
- Actualizar el código de la función y configurar con una variable de entorno.

- Invocar la función con un nuevo parámetro y obtener resultados. Mostrar el registro de ejecución devuelto.
- Enumerar las funciones de su cuenta y, luego, limpiar los recursos.

Para obtener información, consulte [Crear una función de Lambda con la consola](#).

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
#!/ Get started with functions scenario.
/*!
 \param clientConfig: AWS client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::Lambda::getStartedWithFunctionsScenario(
    const Aws::Client::ClientConfiguration &clientConfig) {

    Aws::Lambda::LambdaClient client(clientConfig);

    // 1. Create an AWS Identity and Access Management (IAM) role for Lambda
    function.
    Aws::String roleArn;
    if (!getIamRoleArn(roleArn, clientConfig)) {
        return false;
    }

    // 2. Create a Lambda function.
    int seconds = 0;
    do {
        Aws::Lambda::Model::CreateFunctionRequest request;
        request.SetFunctionName(LAMBDA_NAME);
        request.SetDescription(LAMBDA_DESCRIPTION); // Optional.
#ifdef USE_CPP_LAMBDA_FUNCTION
        request.SetRuntime(Aws::Lambda::Model::Runtime::provided_al2);
        request.SetTimeout(15);
        request.SetMemorySize(128);
#endif
    } while (seconds < 10);
}
```

```
        // 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::JsonValue> values =

```

```

        jsonValue.View().GetAllObjects();
    auto iter = values.find("result");
    if (iter != values.end() && iter->second.IsIntegerType()) {
        {
            std::cout << INCREMENT_RESULT_PREFIX
                << iter->second.AsInteger() << std::endl;
        }
    }
    else {
        std::cout << "There was an error in execution. Here is the log."
            << std::endl;
        Aws::Utils::ByteBuffer buffer =
    Aws::Utils::HashingUtils::Base64Decode(
            invokeResult.GetLogResult());
        std::cout << "With log " << buffer.GetUnderlyingData() << std::endl;
    }
}

std::cout
    << "The Lambda function will now be updated with new code. Press return
to continue, ";
    Aws::String answer;
    std::getline(std::cin, answer);

// 4. Update the Lambda function code.
{
    Aws::Lambda::Model::UpdateFunctionCodeRequest request;
    request.SetFunctionName(LAMBDA_NAME);
    std::ifstream ifstream(CALCULATOR_LAMBDA_CODE.c_str(),
        std::ios_base::in | std::ios_base::binary);
    if (!ifstream.is_open()) {
        std::cerr << "Error opening file " << INCREMENT_LAMBDA_CODE << "." <<
std::endl;
    }

#ifdef USE_CPP_LAMBDA_FUNCTION
    std::cerr
        << "The cpp Lambda function must be built following the
instructions in the cpp_lambda/README.md file. "
        << std::endl;
#endif

    deleteLambdaFunction(client);
    deleteIamRole(clientConfig);
    return false;
}

```

```
    }

    Aws::StringStream buffer;
    buffer << ifstream.rdbuf();
    request.SetZipFile(
        Aws::Utils::ByteBuffer((unsigned char *) buffer.str().c_str(),
                                buffer.str().length()));

    request.SetPublish(true);

    Aws::Lambda::Model::UpdateFunctionCodeOutcome outcome =
client.UpdateFunctionCode(
    request);

    if (outcome.IsSuccess()) {
        std::cout << "The lambda code was successfully updated." << std::endl;
    }
    else {
        std::cerr << "Error with Lambda::UpdateFunctionCode. "
        << outcome.GetError().GetMessage()
        << std::endl;
    }
}
}

std::cout
    << "This function uses an environment variable to control the logging
level."
    << std::endl;
std::cout
    << "UpdateFunctionConfiguration will be used to set the LOG_LEVEL to
DEBUG."
    << std::endl;
seconds = 0;

// 5. Update the Lambda function configuration.
do {
    ++seconds;
    std::this_thread::sleep_for(std::chrono::seconds(1));
    Aws::Lambda::Model::UpdateFunctionConfigurationRequest request;
    request.SetFunctionName(LAMBDA_NAME);
    Aws::Lambda::Model::Environment environment;
    environment.AddVariables("LOG_LEVEL", "DEBUG");
    request.SetEnvironment(environment);
```

```

    Aws::Lambda::Model::UpdateFunctionConfigurationOutcome outcome =
client.UpdateFunctionConfiguration(
    request);

    if (outcome.IsSuccess()) {
        std::cout << "The lambda configuration was successfully updated."
            << std::endl;
        break;
    }

    // RESOURCE_IN_USE: function code update not completed.
    else if (outcome.GetError().GetErrorType() !=
        Aws::Lambda::LambdaErrors::RESOURCE_IN_USE) {
        if ((seconds % 10) == 0) { // Log status every 10 seconds.
            std::cout << "Lambda function update in progress . After " <<
seconds
                << " seconds elapsed." << std::endl;
        }
    }
    else {
        std::cerr << "Error with Lambda::UpdateFunctionConfiguration. "
            << outcome.GetError().GetMessage()
            << std::endl;
    }

} while (0 < seconds);

if (0 > seconds) {
    std::cerr << "Function failed to become active." << std::endl;
}
else {
    std::cout << "Updated function active after " << seconds << " seconds."
        << std::endl;
}

std::cout
    << "\nThe new code applies an arithmetic operator to two variables, x an
y."
    << std::endl;
std::vector<Aws::String> operators = {"plus", "minus", "times", "divided-by"};
for (size_t i = 0; i < operators.size(); ++i) {
    std::cout << "    " << i + 1 << " " << operators[i] << std::endl;
}

```

```

// 6. Invoke the updated Lambda function.
do {
    int operatorIndex = askQuestionForIntRange("Select an operator index 1 - 4
", 1,
                                             4);
    int x = askQuestionForInt("Enter an integer for the x value ");
    int y = askQuestionForInt("Enter an integer for the y value ");

    Aws::Utils::Json::JsonValue calculateJsonPayload;
    calculateJsonPayload.WithString("action", operators[operatorIndex - 1]);
    calculateJsonPayload.WithInteger("x", x);
    calculateJsonPayload.WithInteger("y", y);
    Aws::Lambda::Model::InvokeResult calculatedResult;
    if (invokeLambdaFunction(calculateJsonPayload,
                            Aws::Lambda::Model::LogType::Tail,
                            calculatedResult, client)) {
        Aws::Utils::Json::JsonValue jsonValue(calculatedResult.GetPayload());
        Aws::Map<Aws::String, Aws::Utils::Json::JsonValue> 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;
}
```


- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para C++ .
 - [CreateFunction](#)
 - [DeleteFunction](#)
 - [GetFunction](#)
 - [Invoke](#)
 - [ListFunctions](#)
 - [UpdateFunctionCode](#)
 - [UpdateFunctionConfiguration](#)

Acciones

CreateFunction

En el siguiente ejemplo de código, se muestra cómo utilizar CreateFunction.

SDK para C++

 Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

    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;
    }
}
```

- Para obtener más información sobre la API, consulta [CreateFunction](#) la Referencia AWS SDK para C++ de la API.

DeleteFunction

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteFunction.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
```

- Para obtener más información sobre la API, consulta [DeleteFunction](#) la Referencia AWS SDK para C++ de la API.

GetFunction

En el siguiente ejemplo de código, se muestra cómo utilizar GetFunction.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
```

- Para obtener más información sobre la API, consulta [GetFunction](#) la Referencia AWS SDK para C++ de la API.

Invoke

En el siguiente ejemplo de código, se muestra cómo utilizar Invoke.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
```

- Para obtener información sobre la API, consulte [Invocar](#) en la Referencia de la API de AWS SDK para C++ .

ListFunctions

En el siguiente ejemplo de código, se muestra cómo utilizar `ListFunctions`.

SDK para C++

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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());

```

- Para obtener más información sobre la API, consulta [ListFunctions](#) la Referencia AWS SDK para C++ de la API.

UpdateFunctionCode

En el siguiente ejemplo de código, se muestra cómo utilizar UpdateFunctionCode.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

    Aws::Client::ClientConfiguration clientConfig;
    // Optional: Set to the AWS Region in which the bucket was created
    (overrides config file).
    // clientConfig.region = "us-east-1";

    Aws::Lambda::LambdaClient client(clientConfig);

    Aws::Lambda::Model::UpdateFunctionCodeRequest request;
    request.SetFunctionName(LAMBDA_NAME);
    std::ifstream ifstream(CALCULATOR_LAMBDA_CODE.c_str(),
        std::ios_base::in | std::ios_base::binary);

```



```

        if (!ifstream.is_open()) {
            std::cerr << "Error opening file " << INCREMENT_LAMBDA_CODE << "." <<
std::endl;

#ifdef USE_CPP_LAMBDA_FUNCTION
            std::cerr
                << "The cpp Lambda function must be built following the
instructions in the cpp_lambda/README.md file. "
                << std::endl;
#endif

            deleteLambdaFunction(client);
            deleteIamRole(clientConfig);
            return false;
        }

        Aws::StringStream buffer;
        buffer << ifstream.rdbuf();
        request.SetZipFile(
            Aws::Utils::ByteBuffer((unsigned char *) buffer.str().c_str(),
buffer.str().length()));

        request.SetPublish(true);

        Aws::Lambda::Model::UpdateFunctionCodeOutcome outcome =
client.UpdateFunctionCode(
    request);

        if (outcome.IsSuccess()) {
            std::cout << "The lambda code was successfully updated." << std::endl;
        }
        else {
            std::cerr << "Error with Lambda::UpdateFunctionCode. "
                << outcome.GetError().GetMessage()
                << std::endl;
        }
    }
}


```

- Para obtener más información sobre la API, consulta [UpdateFunctionCode](#) la Referencia AWS SDK para C++ de la API.

UpdateFunctionConfiguration

En el siguiente ejemplo de código, se muestra cómo utilizar UpdateFunctionConfiguration.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
```

- Para obtener más información sobre la API, consulta [UpdateFunctionConfiguration](#) la Referencia AWS SDK para C++ de la API.

Escenarios

Creación de una aplicación sin servidor para administrar fotos

En el siguiente ejemplo de código se muestra cómo crear una aplicación sin servidor que permita a los usuarios administrar fotos mediante etiquetas.

SDK para C++

Muestra cómo desarrollar una aplicación de administración de activos fotográficos que detecte las etiquetas de las imágenes mediante Amazon Rekognition y las almacene para su posterior recuperación.

Para obtener el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en [GitHub](#).

Para profundizar en el origen de este ejemplo, consulte la publicación en [Comunidad de AWS](#).

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

MediaConvert ejemplos de uso de SDK para C++

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK para C++ with MediaConvert.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Acciones](#)

Acciones

CreateJob

En el siguiente ejemplo de código, se muestra cómo utilizar CreateJob.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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


```

- Para obtener más información sobre la API, consulta [CreateJob](#) la Referencia AWS SDK para C++ de la API.

GetJob

En el siguiente ejemplo de código, se muestra cómo utilizar GetJob.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener más información sobre la API, consulta [GetJob](#) la Referencia AWS SDK para C++ de la API.

ListJobs

En el siguiente ejemplo de código, se muestra cómo utilizar ListJobs.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
#!/ 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;
}
```

- Para obtener más información sobre la API, consulta [ListJobs](#) la Referencia AWS SDK para C++ de la API.

Ejemplos de Amazon RDS usando SDK para C++

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK para C++ mediante Amazon RDS.

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Introducción](#)
- [Conceptos básicos](#)
- [Acciones](#)
- [Escenarios](#)

Introducción

Introducción a Amazon RDS

El siguiente ejemplo de código muestra cómo empezar a utilizar Amazon RDS.

SDK para C++

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Código para el CMake archivo 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})

```

Código del archivo de origen 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;
}
```

- Para obtener información detallada sobre la API, consulte la [sección Describir DBInstances](#) en la referencia de la AWS SDK para C++ API.

Conceptos básicos

Conceptos básicos

En el siguiente ejemplo de código, se muestra cómo:

- Cree un grupo de parámetros de base de datos personalizado y defina los valores de los parámetros.
- Cree una instancia de base de datos que esté configurada para utilizar el grupo de parámetros. La instancia de base de datos también contiene una base de datos.
- Cree una instantánea de la instancia.
- Elimine la instancia y el grupo de parámetros.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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.GetDataTypes() == "integer")) {
        std::cout << "The " << autoIncParameter.GetParameterName()
                  << " is described as: " <<
                  autoIncParameter.GetDescription() << "." << std::endl;
        if (autoIncParameter.ParameterValueHasBeenSet()) {
            std::cout << "The current value is "
                      << autoIncParameter.GetParameterValue()
                      << "." << std::endl;
        }
        std::vector<int> splitValues = splitToInts(
            autoIncParameter.GetAllowedValues(), '-');
        if (splitValues.size() == 2) {
            int newValue = askQuestionForIntRange(
                Aws::String("Enter a new value in the range ") +
                autoIncParameter.GetAllowedValues() + ": ",
                splitValues[0], splitValues[1]);
            autoIncParameter.SetParameterValue(std::to_string(newValue));
            updateParameters.push_back(autoIncParameter);
        }
        else {
            std::cerr << "Error parsing " << autoIncParameter.GetAllowedValues()
                      << std::endl;
        }
    }
}
}

```

```

{
    // 5. Modify the auto increment parameters in the group.
    Aws::RDS::Model::ModifyDBParameterGroupRequest request;
    request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
    request.SetParameters(updateParameters);

    Aws::RDS::Model::ModifyDBParameterGroupOutcome outcome =
        client.ModifyDBParameterGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB parameter group was successfully modified."
                  << std::endl;
    }
    else {
        std::cerr << "Error with RDS::ModifyDBParameterGroup. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }
}

std::cout
    << "You can get a list of parameters you've set by specifying a source
of 'user'."
    << std::endl;

    Aws::Vector<Aws::RDS::Model::Parameter> userParameters;
    // 6. Display the modified parameters in the group.
    if (!getDBParameters(PARAMETER_GROUP_NAME, NO_NAME_PREFIX, "user",
userParameters,
                        client)) {
        cleanUpResources(PARAMETER_GROUP_NAME, "", client);
        return false;
    }

    for (const auto &userParameter: userParameters) {
        std::cout << " " << userParameter.GetParameterName() << ", " <<
            userParameter.GetDescription() << ", parameter value - "
            << userParameter.GetParameterValue() << std::endl;
    }

    printAsterisksLine();
    std::cout << "Checking for an existing DB instance." << std::endl;

```

```

    Aws::RDS::Model::DBInstance dbInstance;
    // 7. Check if the DB instance already exists.
    if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
        cleanUpResources(PARAMETER_GROUP_NAME, "", client);
        return false;
    }

    if (dbInstance.DbInstancePortHasBeenSet()) {
        std::cout << "The DB instance already exists." << std::endl;
    }
    else {
        std::cout << "Let's create a DB instance." << std::endl;
        const Aws::String administratorName = askQuestion(
            "Enter an administrator username for the database: ");
        const Aws::String administratorPassword = askQuestion(
            "Enter a password for the administrator (at least 8 characters): ");
        Aws::Vector<Aws::RDS::Model::DBEngineVersion> engineVersions;

        // 8. Get a list of available engine versions.
        if (!getDBEngineVersions(DB_ENGINE, dbParameterGroupFamily, engineVersions,
            client)) {
            cleanUpResources(PARAMETER_GROUP_NAME, "", client);
            return false;
        }

        std::cout << "The available engines for your parameter group are:" <<
std::endl;

        int index = 1;
        for (const Aws::RDS::Model::DBEngineVersion &engineVersion: engineVersions)
        {
            std::cout << " " << index << ": " << engineVersion.GetEngineVersion()
                << std::endl;
            ++index;
        }
        int choice = askQuestionForIntRange("Which engine do you want to use? ", 1,
static_cast<int>(engineVersions.size()));
        const Aws::RDS::Model::DBEngineVersion engineVersion = engineVersions[choice
-
                                                                    1];

        Aws::String dbInstanceClass;
        // 9. Get a list of micro instance classes.

```

```

    if (!chooseMicroDBInstanceClass(engineVersion.GetEngine(),
                                    engineVersion.GetEngineVersion(),
                                    dbInstanceClass,
                                    client)) {
        cleanUpResources(PARAMETER_GROUP_NAME, "", client);
        return false;
    }

    std::cout << "Creating a DB instance named '" << DB_INSTANCE_IDENTIFIER
              << "' and database '" << DB_NAME << "'.\n"
              << "The DB instance is configured to use your custom parameter
group '"
              << PARAMETER_GROUP_NAME << "',\n"
              << "selected engine version " << engineVersion.GetEngineVersion()
              << ",\n"
              << "selected DB instance class '" << dbInstanceClass << "',"
              << " and " << DB_ALLOCATED_STORAGE << " GiB of " <<
DB_STORAGE_TYPE
              << " storage.\nThis typically takes several minutes." <<
std::endl;

    Aws::RDS::Model::CreateDBInstanceRequest request;
    request.SetDBName(DB_NAME);
    request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
    request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
    request.SetEngine(engineVersion.GetEngine());
    request.SetEngineVersion(engineVersion.GetEngineVersion());
    request.SetDBInstanceClass(dbInstanceClass);
    request.SetStorageType(DB_STORAGE_TYPE);
    request.SetAllocatedStorage(DB_ALLOCATED_STORAGE);
    request.SetMasterUsername(administratorName);
    request.SetMasterUserPassword(administratorPassword);

    Aws::RDS::Model::CreateDBInstanceOutcome outcome =
        client.CreateDBInstance(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB instance creation has started."
                  << std::endl;
    }
    else {
        std::cerr << "Error with RDS::CreateDBInstance. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

```

```
        cleanUpResources(PARAMETER_GROUP_NAME, "", client);
        return false;
    }
}

std::cout << "Waiting for the DB instance to become available." << std::endl;

int counter = 0;
// 11. Wait for the DB instance to become available.
do {
    std::this_thread::sleep_for(std::chrono::seconds(1));
    ++counter;
    if (counter > 900) {
        std::cerr << "Wait for instance to become available timed out after "
            << counter
            << " seconds." << std::endl;
        cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER, client);
        return false;
    }

    dbInstance = Aws::RDS::Model::DBInstance();
    if (!describeDBInstance(DB_INSTANCE_IDENTIFIER, dbInstance, client)) {
        cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER, client);
        return false;
    }

    if ((counter % 20) == 0) {
        std::cout << "Current DB instance status is '"
            << dbInstance.GetDBInstanceStatus()
            << "' after " << counter << " seconds." << std::endl;
    }
} while (dbInstance.GetDBInstanceStatus() != "available");

if (dbInstance.GetDBInstanceStatus() == "available") {
    std::cout << "The DB instance has been created." << std::endl;
}

printAsterisksLine();

// 12. Display the connection string that can be used to connect a 'mysql' shell
to the database.
displayConnection(dbInstance);

printAsterisksLine();
```

```

if (askYesNoQuestion(
    "Do you want to create a snapshot of your DB instance (y/n)? ") {
    Aws::String snapshotID(DB_INSTANCE_IDENTIFIER + "-" +
        Aws::String(Aws::Utils::UUID::RandomUUID()));
    {
        std::cout << "Creating a snapshot named " << snapshotID << "." <<
std::endl;
        std::cout << "This typically takes a few minutes." << std::endl;

        // 13. Create a snapshot of the DB instance.
        Aws::RDS::Model::CreateDBSnapshotRequest request;
        request.SetDBInstanceIdentifier(DB_INSTANCE_IDENTIFIER);
        request.SetDBSnapshotIdentifier(snapshotID);

        Aws::RDS::Model::CreateDBSnapshotOutcome outcome =
            client.CreateDBSnapshot(request);

        if (outcome.IsSuccess()) {
            std::cout << "Snapshot creation has started."
                << std::endl;
        }
        else {
            std::cerr << "Error with RDS::CreateDBSnapshot. "
                << outcome.GetError().GetMessage()
                << std::endl;
            cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
            return false;
        }
    }

    std::cout << "Waiting for snapshot to become available." << std::endl;

    Aws::RDS::Model::DBSnapshot snapshot;
    counter = 0;
    do {
        std::this_thread::sleep_for(std::chrono::seconds(1));
        ++counter;
        if (counter > 600) {
            std::cerr << "Wait for snapshot to be available timed out after "
                << counter
                << " seconds." << std::endl;

```



```

        cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
        return false;
    }

    // 14. Wait for the snapshot to become available.
    Aws::RDS::Model::DescribeDBSnapshotsRequest request;
    request.SetDBSnapshotIdentifier(snapshotID);

    Aws::RDS::Model::DescribeDBSnapshotsOutcome outcome =
        client.DescribeDBSnapshots(request);

    if (outcome.IsSuccess()) {
        snapshot = outcome.GetResult().GetDBSnapshots()[0];
    }
    else {
        std::cerr << "Error with RDS::DescribeDBSnapshots. "
            << outcome.GetError().GetMessage()
            << std::endl;
        cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
        return false;
    }

    if ((counter % 20) == 0) {
        std::cout << "Current snapshot status is '"
            << snapshot.GetStatus()
            << "' after " << counter << " seconds." << std::endl;
    }
} while (snapshot.GetStatus() != "available");

if (snapshot.GetStatus() != "available") {
    std::cout << "A snapshot has been created." << std::endl;
}
}

printAsterisksLine();

bool result = true;
if (askYesNoQuestion(
    "Do you want to delete the DB instance and parameter group (y/n)? ")) {
    result = cleanUpResources(PARAMETER_GROUP_NAME, DB_INSTANCE_IDENTIFIER,
client);
}

```

```

    return result;
}

//! Routine which gets DB parameters using the 'DescribeDBParameters' api.
/*!
 \sa getDBParameters()
 \param parameterGroupName: The name of the parameter group.
 \param namePrefix: Prefix string to filter results by parameter name.
 \param source: A source such as 'user', ignored if empty.
 \param parametersResult: Vector of 'Parameter' objects returned by the routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::getDBParameters(const Aws::String &parameterGroupName,
                                   const Aws::String &namePrefix,
                                   const Aws::String &source,
                                   Aws::Vector<Aws::RDS::Model::Parameter>
&parametersResult,
                                   const Aws::RDS::RDSClient &client) {
    Aws::String marker;
    do {
        Aws::RDS::Model::DescribeDBParametersRequest request;
        request.SetDBParameterGroupName(PARAMETER_GROUP_NAME);
        if (!marker.empty()) {
            request.SetMarker(marker);
        }
        if (!source.empty()) {
            request.SetSource(source);
        }

        Aws::RDS::Model::DescribeDBParametersOutcome outcome =
            client.DescribeDBParameters(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::RDS::Model::Parameter> &parameters =
                outcome.GetResult().GetParameters();
            for (const Aws::RDS::Model::Parameter &parameter: parameters) {
                if (!namePrefix.empty()) {
                    if (parameter.GetParameterName().find(namePrefix) == 0) {
                        parametersResult.push_back(parameter);
                    }
                }
            }
        }
    }
}

```

```

        else {
            parametersResult.push_back(parameter);
        }
    }

    marker = outcome.GetResult().GetMarker();
}
else {
    std::cerr << "Error with RDS::DescribeDBParameters. "
                << outcome.GetError().GetMessage()
                << std::endl;
    return false;
}
} while (!marker.empty());

return true;
}

//! Routine which gets available DB engine versions for an engine name and
//! an optional parameter group family.
/*!
 \sa getDBEngineVersions()
 \param engineName: A DB engine name.
 \param parameterGroupFamily: A parameter group family name, ignored if empty.
 \param engineVersionsResult: Vector of 'DBEngineVersion' objects returned by the
 routine.
 \param client: 'RDSClient' instance.
 \return bool: Successful completion.
 */
bool AwsDoc::RDS::getDBEngineVersions(const Aws::String &engineName,
                                       const Aws::String &parameterGroupFamily,
                                       Aws::Vector<Aws::RDS::Model::DBEngineVersion>
&engineVersionsResult,
                                       const Aws::RDS::RDSClient &client) {
    Aws::RDS::Model::DescribeDBEngineVersionsRequest request;
    request.SetEngine(engineName);
    if (!parameterGroupFamily.empty()) {
        request.SetDBParameterGroupFamily(parameterGroupFamily);
    }

    engineVersionsResult.clear();
    Aws::String marker; // Used for pagination.

```

```

do {
    if (!marker.empty()) {
        request.SetMarker(marker);
    }

    Aws::RDS::Model::DescribeDBEngineVersionsOutcome outcome =
        client.DescribeDBEngineVersions(request);

    if (outcome.IsSuccess()) {
        auto &engineVersions = outcome.GetResult().GetDBEngineVersions();
        engineVersionsResult.insert(engineVersionsResult.end(),
engineVersions.begin(),
                                engineVersions.end());
        marker = outcome.GetResult().GetMarker();
    }
    else {
        std::cerr << "Error with RDS::DescribeDBEngineVersionsRequest. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
        return false;
    }
} while (!marker.empty());

return true;
}

//! Routine which gets a DB instance description.
/*!
 \sa describeDBInstance()
 \param 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 after " <<
counter
                << " seconds." << std::endl;
        return false;
    }

    dbInstance = Aws::RDS::Model::DBInstance();
    // 16. Wait for the DB instance to be deleted.
    if (!describeDBInstance(dbInstanceIdentifier, dbInstance, client)) {
        return false;
    }

    if (dbInstance.DBInstanceIdentifierHasBeenSet() && (counter % 20) == 0)
{
        std::cout << "Current DB instance status is '"
                << dbInstance.GetDBInstanceStatus()
                << "' after " << counter << " seconds." << std::endl;
    }
} while (dbInstance.DBInstanceIdentifierHasBeenSet());
}

if (!parameterGroupName.empty()) {
    // 17. Delete the parameter group.
    Aws::RDS::Model::DeleteDBParameterGroupRequest request;
    request.SetDBParameterGroupName(parameterGroupName);

    Aws::RDS::Model::DeleteDBParameterGroupOutcome outcome =
        client.DeleteDBParameterGroup(request);

    if (outcome.IsSuccess()) {
        std::cout << "The DB parameter group was successfully deleted."
                << std::endl;
    }
    else {
        std::cerr << "Error with RDS::DeleteDBParameterGroup. "
                << outcome.GetError().GetMessage()
                << std::endl;
        result = false;
    }
}
```



```
    return result;
}
```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para C++ .
 - [CrearDBInstance](#)
 - [Crear DBParameter grupo](#)
 - [CrearDBSnapshot](#)
 - [DeleteDBInstance](#)
 - [Eliminar DBParameter grupo](#)
 - [DescribeDBEngine las versiones](#)
 - [DescribirDBInstances](#)
 - [Describe DBParameter los grupos](#)
 - [DescribirDBParameters](#)
 - [DescribirDBSnapshots](#)
 - [DescribeOrderableDBInstanceOpciones](#)
 - [Modificar DBParameter grupo](#)

Acciones

CreateDBInstance

En el siguiente ejemplo de código, se muestra cómo utilizar CreateDBInstance.

SDK para C++

Note

Hay más en marcha GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
// 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;
}
```

- Para obtener más información sobre la API, consulta la [sección Crear DBInstance](#) en la referencia de la AWS SDK para C++ API.

CreateDBParameterGroup

En el siguiente ejemplo de código, se muestra cómo utilizar CreateDBParameterGroup.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
```

- Para obtener más información sobre la API, consulta [Crear un DBParameter grupo](#) en la referencia AWS SDK para C++ de la API.

CreateDBSnapshot

En el siguiente ejemplo de código, se muestra cómo utilizar CreateDBSnapshot.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
    }
}
```

- Para obtener más información sobre la API, consulta la [sección Crear DBSnapshot](#) en la referencia de la AWS SDK para C++ API.

DeleteDBInstance

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteDBInstance.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
    }
}
```

- Para obtener más información sobre la API, consulta [Eliminar DBInstance](#) en la referencia AWS SDK para C++ de la API.

DeleteDBParameterGroup

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteDBParameterGroup.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
```

- Para obtener más información sobre la API, consulta [Eliminar DBParameter grupo](#) en la referencia AWS SDK para C++ de la API.

DescribeDBEngineVersions

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeDBEngineVersions.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

    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;
}
```

- Para obtener más información sobre la API, consulta la [sección Describir DBEngine las versiones](#) en la referencia de la AWS SDK para C++ API.

DescribeDBInstances

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeDBInstances.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).


```

    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;
    }
}

```

- Para obtener más información sobre la API, consulta la [sección Describir DBInstances](#) en la referencia de la AWS SDK para C++ API.

DescribeDBParameterGroups

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeDBParameterGroups.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
```

```
}

```

- Para obtener más información sobre la API, consulta la [sección Describir DBParameter los grupos](#) en la referencia de la AWS SDK para C++ API.

DescribeDBParameters

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeDBParameters.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
```

- Para obtener más información sobre la API, consulta la [sección Describir DBParameters](#) en la referencia de la AWS SDK para C++ API.

DescribeDBSnapshots

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeDBSnapshots.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
    }
}
```

- Para obtener más información sobre la API, consulta la [sección Describir DBSnapshots](#) en la referencia de la AWS SDK para C++ API.

DescribeOrderableDBInstanceOptions

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeOrderableDBInstanceOptions.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

    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;
}
```

- Para obtener más información sobre la API, consulta [DescribeOrderableDBInstancias opciones](#) en la referencia AWS SDK para C++ de la API.

ModifyDBParameterGroup

En el siguiente ejemplo de código, se muestra cómo utilizar `ModifyDBParameterGroup`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
```


- Para obtener más información sobre la API, consulta [Modificar el DBParameter grupo](#) en la referencia de la AWS SDK para C++ API.

Escenarios

Crear un rastreador de elementos de trabajo de Aurora Serverless

El siguiente ejemplo de código muestra cómo crear una aplicación web que realice un seguimiento de los elementos de trabajo de una base de datos de Amazon Aurora sin servidor y use Amazon Simple Email Service (Amazon SES) para enviar informes.

SDK para C++

Muestra cómo crear una aplicación web que realice un seguimiento de los elementos de trabajo almacenados en una base de datos de Amazon Aurora sin servidor e informe al respecto.

Para obtener el código fuente completo y las instrucciones sobre cómo configurar una API REST de C++ que consulte los datos de Amazon Aurora Serverless y para que la utilice una aplicación de React, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- Aurora
- Amazon RDS
- Servicio de datos de Amazon RDS
- Amazon SES

Ejemplos de servicios de datos de Amazon RDS con SDK para C++

+

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar situaciones comunes mediante Amazon RDS Data Service. AWS SDK para C++

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Escenarios](#)

Escenarios

Crear un rastreador de elementos de trabajo de Aurora Serverless

El siguiente ejemplo de código muestra cómo crear una aplicación web que realice un seguimiento de los elementos de trabajo de una base de datos de Amazon Aurora sin servidor y use Amazon Simple Email Service (Amazon SES) para enviar informes.

SDK para C++

Muestra cómo crear una aplicación web que realice un seguimiento de los elementos de trabajo almacenados en una base de datos de Amazon Aurora sin servidor e informe al respecto.

Para obtener el código fuente completo y las instrucciones sobre cómo configurar una API REST de C++ que consulte los datos de Amazon Aurora Serverless y para que la utilice una aplicación de React, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- Aurora
- Amazon RDS
- Servicio de datos de Amazon RDS
- Amazon SES

Ejemplos de Amazon Rekognition con SDK para C++

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar situaciones comunes mediante Amazon AWS SDK para C++ Rekognition.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Introducción](#)
- [Acciones](#)
- [Escenarios](#)

Introducción

Introducción a Amazon Rekognition

En el siguiente ejemplo de código se muestra cómo empezar a utilizar Amazon Rekognition.

SDK para C++

Note

Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Código para el CMake archivo 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})

```

Código del archivo de origen 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;
}
```

- Para obtener más información sobre la API, consulte [ListCollections](#) la Referencia AWS SDK para C++ de la API.

Acciones

DetectLabels

En el siguiente ejemplo de código, se muestra cómo utilizar DetectLabels.

Para obtener información, consulte [Detección de etiquetas en una imagen](#).

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener más información sobre la API, consulta [DetectLabels](#) la Referencia AWS SDK para C++ de la API.

Escenarios

Creación de una aplicación sin servidor para administrar fotos

En el siguiente ejemplo de código se muestra cómo crear una aplicación sin servidor que permita a los usuarios administrar fotos mediante etiquetas.

SDK para C++

Muestra cómo desarrollar una aplicación de administración de activos fotográficos que detecte las etiquetas de las imágenes mediante Amazon Rekognition y las almacene para su posterior recuperación.

Para obtener el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en [GitHub](#).

Para profundizar en el origen de este ejemplo, consulte la publicación en [Comunidad de AWS](#).

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Ejemplos de Amazon S3 usando SDK para C++

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar situaciones comunes AWS SDK para C++ mediante Amazon S3.

Los conceptos básicos son ejemplos de código que muestran cómo realizar las operaciones esenciales dentro de un servicio.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Introducción](#)

- [Conceptos básicos](#)
- [Acciones](#)
- [Escenarios](#)

Introducción

Introducción a Amazon S3

El siguiente ejemplo de código muestra cómo empezar a utilizar Amazon S3.

SDK para C++

Note

Hay más información al respecto [en GitHub](#). Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Código para el CMake archivo 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})
```

Código del archivo de origen 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;
}
```

- Para obtener más información sobre la API, consulte [ListBuckets](#) la Referencia AWS SDK para C++ de la API.

Conceptos básicos

Conceptos básicos

En el siguiente ejemplo de código, se muestra cómo:

- Creación de un bucket y cargar un archivo en el bucket.
- Descargar un objeto desde un bucket.
- Copiar un objeto en una subcarpeta de un bucket.
- Obtención de una lista de los objetos de un bucket.
- Eliminación del bucket y todos los objetos que incluye.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
#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::OStream 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();
}

```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para C++ .
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)

- [DeleteObjects](#)
- [GetObject](#)
- [ListObjectsV2](#)
- [PutObject](#)

Acciones

AbortMultipartUpload

En el siguiente ejemplo de código, se muestra cómo utilizar `AbortMultipartUpload`.

SDK para C++

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
#!/ 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();
}

```

- Para obtener más información sobre la API, consulta [AbortMultipartUpload](#) la Referencia AWS SDK para C++ de la API.

CompleteMultipartUpload

En el siguiente ejemplo de código, se muestra cómo utilizar CompleteMultipartUpload.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

//! 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;
}

```

- Para obtener más información sobre la API, consulta [CompleteMultipartUpload](#) la Referencia AWS SDK para C++ de la API.

CopyObject

En el siguiente ejemplo de código, se muestra cómo utilizar CopyObject.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

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();
}

```

- Para obtener más información sobre la API, consulta [CopyObject](#) la Referencia AWS SDK para C++ de la API.

CreateBucket

En el siguiente ejemplo de código, se muestra cómo utilizar CreateBucket.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

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();
}
```

- Para obtener más información sobre la API, consulta [CreateBucket](#) la Referencia AWS SDK para C++ de la API.

CreateMultipartUpload

En el siguiente ejemplo de código, se muestra cómo utilizar CreateMultipartUpload.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

//! 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;
}


```

- Para obtener más información sobre la API, consulta [CreateMultipartUpload](#) la Referencia AWS SDK para C++ de la API.

DeleteBucket

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteBucket.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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();
}
```

- Para obtener más información sobre la API, consulta [DeleteBucket](#) la Referencia AWS SDK para C++ de la API.

DeleteBucketPolicy

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteBucketPolicy.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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();
}
```

- Para obtener más información sobre la API, consulta [DeleteBucketPolicy](#) la Referencia AWS SDK para C++ de la API.

DeleteBucketWebsite

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteBucketWebsite.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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();
}
```

- Para obtener más información sobre la API, consulta [DeleteBucketWebsite](#) la Referencia AWS SDK para C++ de la API.

DeleteObject

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteObject.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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();
}
```

- Para obtener más información sobre la API, consulta [DeleteObject](#) la Referencia AWS SDK para C++ de la API.

DeleteObjects

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteObjects.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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();
}

```

- Para obtener más información sobre la API, consulta [DeleteObjects](#) la Referencia AWS SDK para C++ de la API.

GetBucketAcl

En el siguiente ejemplo de código, se muestra cómo utilizar `GetBucketAcl`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

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";
}
```


- Para obtener más información sobre la API, consulta [GetBucketAcl](#) la Referencia AWS SDK para C++ de la API.

GetBucketPolicy

En el siguiente ejemplo de código, se muestra cómo utilizar `GetBucketPolicy`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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();
}
```

- Para obtener más información sobre la API, consulta [GetBucketPolicy](#) la Referencia AWS SDK para C++ de la API.

GetBucketWebsite

En el siguiente ejemplo de código, se muestra cómo utilizar `GetBucketWebsite`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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();
}

```

- Para obtener más información sobre la API, consulta [GetBucketWebsite](#) la Referencia AWS SDK para C++ de la API.

GetObject

En el siguiente ejemplo de código, se muestra cómo utilizar `GetObject`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

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();
}
```

- Para obtener más información sobre la API, consulta [GetObject](#) la Referencia AWS SDK para C++ de la API.

GetObjectAcl

En el siguiente ejemplo de código, se muestra cómo utilizar `GetObjectAcl`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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";
    }
}

```

- Para obtener más información sobre la API, consulta [GetObjectACL](#) la Referencia AWS SDK para C++ de la API.

GetObjectAttributes

En el siguiente ejemplo de código, se muestra cómo utilizar `GetObjectAttributes`.

SDK para C++

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

// ! 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;
}

```

- Para obtener más información sobre la API, consulta [GetObjectAttributes](#) la Referencia AWS SDK para C++ de la API.

ListBuckets

En el siguiente ejemplo de código, se muestra cómo utilizar ListBuckets.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

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;
}

```

- Para obtener más información sobre la API, consulta [ListBuckets](#) la Referencia AWS SDK para C++ de la API.

ListObjectsV2

En el siguiente ejemplo de código, se muestra cómo utilizar ListObjectsV2.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

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;
}
```

- Para obtener más información sobre la API, consulta la [ListObjectsversión 2](#) en la referencia de la AWS SDK para C++ API.

PutBucketAcl

En el siguiente ejemplo de código, se muestra cómo utilizar PutBucketAcl.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}

```

- Para obtener más información sobre la API, consulta [PutBucketAcl](#) la Referencia AWS SDK para C++ de la API.

PutBucketPolicy

En el siguiente ejemplo de código, se muestra cómo utilizar PutBucketPolicy.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

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"
        "};
}

```

- Para obtener más información sobre la API, consulta [PutBucketPolicy](#) la Referencia AWS SDK para C++ de la API.

PutBucketWebsite

En el siguiente ejemplo de código, se muestra cómo utilizar PutBucketWebsite.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

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

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

    Aws::S3::Model::ErrorDocument errorDocument;
    errorDocument.SetKey(errorPage);

    Aws::S3::Model::WebsiteConfiguration websiteConfiguration;
    websiteConfiguration.SetIndexDocument(indexDocument);
    websiteConfiguration.SetErrorDocument(errorDocument);
}

```

```

Aws::S3::Model::PutBucketWebsiteRequest request;
request.SetBucket(bucketName);
request.SetWebsiteConfiguration(websiteConfiguration);

Aws::S3::Model::PutBucketWebsiteOutcome outcome =
    client.PutBucketWebsite(request);

if (!outcome.IsSuccess()) {
    std::cerr << "Error: PutBucketWebsite: "
              << outcome.GetError().GetMessage() << std::endl;
} else {
    std::cout << "Success: Set website configuration for bucket '"
              << bucketName << "'." << std::endl;
}

return outcome.IsSuccess();
}

```

- Para obtener más información sobre la API, consulta [PutBucketWebsite](#) la Referencia AWS SDK para C++ de la API.

PutObject

En el siguiente ejemplo de código, se muestra cómo utilizar PutObject.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

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();
}


```

- Para obtener más información sobre la API, consulta [PutObject](#) la Referencia AWS SDK para C++ de la API.

PutObjectAcl

En el siguiente ejemplo de código, se muestra cómo utilizar PutObjectAcl.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}

```

- Para obtener más información sobre la API, consulta [PutObject](#) en la Referencia AWS SDK para C++ de la API.

UploadPart

En el siguiente ejemplo de código, se muestra cómo utilizar UploadPart.

SDK para C++

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

//! 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);
}
```

- Para obtener más información sobre la API, consulta [UploadPart](#) la Referencia AWS SDK para C++ de la API.

Escenarios

Crear una URL prefirmada

En el siguiente ejemplo de código se muestra cómo crear una URL prefirmada para Amazon S3 y cargar un objeto.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Genere una URL prefirmada para descargar un objeto.

```
//! 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);
    }

```

Descargue mediante libcurl.

```

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;
}

```

Genere una URL prefirmada para cargar un objeto.

```

/*! 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);
    }

```

Cargue mediante libcurl.

```

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;
}
}
```

Creación de una aplicación sin servidor para administrar fotos

En el siguiente ejemplo de código se muestra cómo crear una aplicación sin servidor que permita a los usuarios administrar fotos mediante etiquetas.

SDK para C++

Muestra cómo desarrollar una aplicación de administración de activos fotográficos que detecte las etiquetas de las imágenes mediante Amazon Rekognition y las almacene para su posterior recuperación.

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en [GitHub](#).

Para profundizar en el origen de este ejemplo, consulte la publicación en [Comunidad de AWS](#).


Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Trabajo con la integridad de los objetos de Amazon S3

El siguiente ejemplo de código muestra cómo trabajar con las características de integridad de objetos de S3.

SDK para C++

 Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Ejecute un escenario interactivo en el que se demuestren las características de integridad de objetos de Amazon S3.

```
//! 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;
}
```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para C++ .
 - [AbortMultipartUpload](#)
 - [CompleteMultipartUpload](#)
 - [CreateMultipartUpload](#)
 - [DeleteObject](#)
 - [GetObjectAttributes](#)

- [PutObject](#)
- [UploadPart](#)

Ejemplos de Secrets Manager usando SDK para C++

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK para C++ mediante Secrets Manager.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Acciones](#)

Acciones

GetSecretValue

En el siguiente ejemplo de código, se muestra cómo utilizar `GetSecretValue`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
#!/ 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();
}

```

- Para obtener más información sobre la API, consulta [GetSecretValue](#) la Referencia AWS SDK para C++ de la API.

Ejemplos de Amazon SES usando SDK para C++

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK para C++ con Amazon SES.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Acciones](#)
- [Escenarios](#)

Acciones

CreateReceiptFilter

En el siguiente ejemplo de código, se muestra cómo utilizar CreateReceiptFilter.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
//! 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();
}

```

- Para obtener más información sobre la API, consulta [CreateReceiptFilter](#) la Referencia AWS SDK para C++ de la API.

CreateReceiptRule

En el siguiente ejemplo de código, se muestra cómo utilizar CreateReceiptRule.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

/*! 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(ruleSetName);  
    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();  
}
```


- Para obtener más información sobre la API, consulta [CreateReceiptRuleSet](#) la Referencia AWS SDK para C++ de la API.

CreateReceiptRuleSet

En el siguiente ejemplo de código, se muestra cómo utilizar CreateReceiptRuleSet.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener más información sobre la API, consulta [CreateReceiptRuleSet](#) la Referencia AWS SDK para C++ de la API.

CreateTemplate

En el siguiente ejemplo de código, se muestra cómo utilizar CreateTemplate.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener más información sobre la API, consulta [CreateTemplate](#) la Referencia AWS SDK para C++ de la API.

DeleteIdentity

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteIdentity.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener más información sobre la API, consulta [DeleteIdentity](#) la Referencia AWS SDK para C++ de la API.

DeleteReceiptFilter

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteReceiptFilter.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener más información sobre la API, consulta [DeleteReceiptFilter](#) la Referencia AWS SDK para C++ de la API.

DeleteReceiptRule

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteReceiptRule.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener más información sobre la API, consulta [DeleteReceiptRule](#) la Referencia AWS SDK para C++ de la API.

DeleteReceiptRuleSet

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteReceiptRuleSet.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
#!/ 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();
}

```

- Para obtener más información sobre la API, consulta [DeleteReceiptRuleSet](#) la Referencia AWS SDK para C++ de la API.

DeleteTemplate

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteTemplate.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

/*! 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();
}
```

- Para obtener más información sobre la API, consulta [DeleteTemplate](#) la Referencia AWS SDK para C++ de la API.

GetTemplate

En el siguiente ejemplo de código, se muestra cómo utilizar GetTemplate.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
///  
/*!  
 \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();
}

```

- Para obtener más información sobre la API, consulta [GetTemplate](#) la Referencia AWS SDK para C++ de la API.

ListIdentities

En el siguiente ejemplo de código, se muestra cómo utilizar `ListIdentities`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

/*! 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;
}
```

- Para obtener más información sobre la API, consulta [ListIdentities](#) la Referencia AWS SDK para C++ de la API.

ListReceiptFilters

En el siguiente ejemplo de código, se muestra cómo utilizar ListReceiptFilters.

SDK para C++

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener más información sobre la API, consulta [ListReceiptFilters](#) la Referencia AWS SDK para C++ de la API.

SendEmail

En el siguiente ejemplo de código, se muestra cómo utilizar `SendEmail`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener más información sobre la API, consulta [SendEmail](#) en la Referencia AWS SDK para C++ de la API.

SendTemplatedEmail

En el siguiente ejemplo de código, se muestra cómo utilizar `SendTemplatedEmail`.

SDK para C++

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener más información sobre la API, consulta [SendTemplatedEmail](#) la Referencia AWS SDK para C++ de la API.

UpdateTemplate

En el siguiente ejemplo de código, se muestra cómo utilizar UpdateTemplate.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener más información sobre la API, consulta [UpdateTemplate](#) la Referencia AWS SDK para C++ de la API.

VerifyEmailIdentity

En el siguiente ejemplo de código, se muestra cómo utilizar `VerifyEmailIdentity`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
//! 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();
}

```

- Para obtener más información sobre la API, consulta [VerifyEmailIdentity](#) la Referencia AWS SDK para C++ de la API.

Escenarios

Crear un rastreador de elementos de trabajo de Aurora Serverless

El siguiente ejemplo de código muestra cómo crear una aplicación web que realice un seguimiento de los elementos de trabajo de una base de datos de Amazon Aurora sin servidor y use Amazon Simple Email Service (Amazon SES) para enviar informes.

SDK para C++

Muestra cómo crear una aplicación web que realice un seguimiento de los elementos de trabajo almacenados en una base de datos de Amazon Aurora sin servidor e informe al respecto.

Para obtener el código fuente completo y las instrucciones sobre cómo configurar una API REST de C++ que consulte los datos de Amazon Aurora Serverless y para que la utilice una aplicación de React, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- Aurora
- Amazon RDS
- Servicio de datos de Amazon RDS
- Amazon SES

Ejemplos de Amazon SNS usando SDK para C++

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK para C++ mediante Amazon SNS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Introducción](#)
- [Acciones](#)
- [Escenarios](#)

Introducción

Hola Amazon SNS

El siguiente ejemplo de código muestra cómo empezar a utilizar Amazon SNS.

SDK para C++

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Código para el CMake archivo 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})

```

Código del archivo de origen 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;  
}
```

- Para obtener más información sobre la API, consulte [ListTopics](#) la Referencia AWS SDK para C++ de la API.

Acciones

CreateTopic

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateTopic`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener más información sobre la API, consulta [CreateTopic](#) la Referencia AWS SDK para C++ de la API.

DeleteTopic

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteTopic.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener más información sobre la API, consulta [DeleteTopic](#) la Referencia AWS SDK para C++ de la API.

GetSMSAttributes

En el siguiente ejemplo de código, se muestra cómo utilizar GetSMSAttributes.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

```
}
```

- Para obtener información sobre la API, consulta la referencia [Get SMSAttributes](#) in AWS SDK para C++ API.

GetTopicAttributes

En el siguiente ejemplo de código, se muestra cómo utilizar `GetTopicAttributes`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
//! 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();
}

```

- Para obtener más información sobre la API, consulta [GetTopicAttributes](#) la Referencia AWS SDK para C++ de la API.

ListSubscriptions

En el siguiente ejemplo de código, se muestra cómo utilizar ListSubscriptions.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

//! 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;
}
```

- Para obtener más información sobre la API, consulta [ListSubscriptions](#) la Referencia AWS SDK para C++ de la API.

ListTopics

En el siguiente ejemplo de código, se muestra cómo utilizar ListTopics.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
#!/ 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;
}

```

- Para obtener más información sobre la API, consulta [ListTopics](#) la Referencia AWS SDK para C++ de la API.

Publish

En el siguiente ejemplo de código, se muestra cómo utilizar Publish.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

/*! 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();
}

```

Publique un mensaje con un atributo.

```

    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;
}


```

- Para obtener detalles sobre la API, consulte [Publish](#) en la Referencia de la API de AWS SDK para C++ .

SetSMSAttributes

En el siguiente ejemplo de código, se muestra cómo utilizar SetSMSAttributes.

SDK para C++

 Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Cómo utilizar Amazon SNS para establecer el atributo predeterminado `SMSType` .

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

- Para obtener más información sobre la API, consulte [Set SMSAttributes](#) in AWS SDK para C++ API Reference.

Subscribe

En el siguiente ejemplo de código, se muestra cómo utilizar `Subscribe`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Suscriba una dirección de correo electrónico a un tema.

```
#!/ 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();
}

```

Suscriba una aplicación móvil a un tema.

```

//! 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();
}

```

Suscriba una función de Lambda a un tema.

```

//! 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();
}

```

```
}
```

Suscriba una cola de SQS a un tema.

```
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;
    }
}
```

Suscribirse con un filtro a un tema.

```

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 ostream;
    ostream << "Filter messages for \"" << queueName
        << "\"'s subscription to the topic \""
        << topicName << "\"? (y/n)";

    // Add filter if user answers yes.
    if (askYesNoQuestion(ostream.str())) {
        Aws::String jsonPolicy = getFilterPolicyFromUser();
        if (!jsonPolicy.empty()) {
            filteringMessages = true;

```



```

        std::cout << "This is the filter policy for this
subscription."
                << std::endl;
        std::cout << jsonPolicy << std::endl;

        request.AddAttributes("FilterPolicy", jsonPolicy);
    }
    else {
        std::cout
            << "Because you did not select any attributes, no
filter "
            << "will be added to this subscription." <<
std::endl;
    }
}
} // if (isFifoTopic)
Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

if (outcome.IsSuccess()) {
    Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
    std::cout << "The queue '" << queueName
                << "' has been subscribed to the topic '"
                << "'" << topicName << "'" << std::endl;
    std::cout << "with the subscription ARN '" << subscriptionARN << "."
                << std::endl;
    subscriptionARNS.push_back(subscriptionARN);
}
else {
    std::cerr << "Error with TopicsAndQueues::Subscribe. "
                << outcome.GetError().GetMessage()
                << std::endl;

    cleanUp(topicARN,
            queueURLS,
            subscriptionARNS,
            snsClient,
            sqsClient);

    return false;
}

```

//! Routine that lets the user select attributes for a subscription filter policy.

```

/#!
\s a 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;
}
```

- Para obtener detalles sobre la API, consulte [Subscribe](#) en la Referencia de la API de AWS SDK para C++ .

Unsubscribe

En el siguiente ejemplo de código, se muestra cómo utilizar Unsubscribe.

SDK para C++

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener detalles sobre la API, consulte [Unsubscribe](#) en la Referencia de la API de AWS SDK para C++ .

Escenarios

Creación de una aplicación sin servidor para administrar fotos

En el siguiente ejemplo de código se muestra cómo crear una aplicación sin servidor que permita a los usuarios administrar fotos mediante etiquetas.

SDK para C++

Muestra cómo desarrollar una aplicación de administración de activos fotográficos que detecte las etiquetas de las imágenes mediante Amazon Rekognition y las almacene para su posterior recuperación.

Para ver el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulta el ejemplo completo en [GitHub](#).

Para profundizar en el origen de este ejemplo, consulte la publicación en [Comunidad de AWS](#).

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda

- Amazon Rekognition
- Amazon S3
- Amazon SNS

Publicación de un mensaje SMS

En el siguiente ejemplo de código se muestra cómo publicar mensajes SMS mediante Amazon SNS.

SDK para C++

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

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

- Para obtener detalles sobre la API, consulte [Publish](#) en la Referencia de la API de AWS SDK para C++ .

Publicación de mensajes en colas

En el siguiente ejemplo de código, se muestra cómo:

- Crear un tema (FIFO o no FIFO)
- Suscribirse a varias colas al tema con la opción de aplicar un filtro
- Publicar mensajes en el tema

- Sondar las colas en busca de los mensajes recibidos

SDK para C++

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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 ostream;
        ostream << "Enter a name for " << (first ? "an" : "the next")
                << " SQS queue. ";
        queueName = askQuestion(ostream.str());

        // 2. Create an SQS queue.
        Aws::SQS::Model::CreateQueueRequest request;
        if (isFifoTopic) {
            request.AddAttributes(Aws::SQS::Model::QueueAttributeName::FifoQueue,
                                "true");
            queueName = queueName + FIFO_SUFFIX;

            if (first) // Only explain this once.
            {
                std::cout
                    << "Because you are creating a FIFO SQS queue, '.fifo'
must "
                    << "be appended to the queue name." << std::endl;
            }
        }

        request.SetQueueName(queueName);
        queueNames.push_back(queueName);

        Aws::SQS::Model::CreateQueueOutcome outcome =
            sqsClient.CreateQueue(request);

        if (outcome.IsSuccess()) {
            queueURL = outcome.GetResult().GetQueueUrl();
            std::cout << "Your new SQS queue with the name '" << queueName
                    << "' and the queue URL " << std::endl;
            std::cout << "'" << queueURL << "' has been created." << std::endl;
        }
    }
}

```

```

    }
    else {
        std::cerr << "Error with SQS::CreateQueue. "
                  << outcome.GetError().GetMessage()
                  << std::endl;

        cleanUp(topicARN,
                queueURLS,
                subscriptionARNS,
                snsClient,
                sqsClient);

        return false;
    }
}
queueURLS.push_back(queueURL);

if (first) // Only explain this once.
{
    std::cout
        << "The queue URL is used to retrieve the queue ARN, which is "
        << "used to create a subscription." << std::endl;
}

Aws::String queueARN;
{
    // 3. Get the SQS queue ARN attribute.
    Aws::SQS::Model::GetQueueAttributesRequest request;
    request.SetQueueUrl(queueURL);

request.AddAttributeNames(Aws::SQS::Model::QueueAttributeName::QueueArn);

    Aws::SQS::Model::GetQueueAttributesOutcome outcome =
        sqsClient.GetQueueAttributes(request);

    if (outcome.IsSuccess()) {
        const Aws::Map<Aws::SQS::Model::QueueAttributeName, Aws::String>
&attributes =
            outcome.GetResult().GetAttributes();
        const auto &iter = attributes.find(
            Aws::SQS::Model::QueueAttributeName::QueueArn);
        if (iter != attributes.end()) {
            queueARN = iter->second;
            std::cout << "The queue ARN '" << queueARN

```

```
        << "" has been retrieved."
        << std::endl;
    }
    else {
        std::cerr
            << "Error ARN attribute not returned by
GetQueueAttribute."
            << std::endl;

        cleanUp(topicARN,
                queueURLS,
                subscriptionARNS,
                snsClient,
                sqsClient);

        return false;
    }
}
else {
    std::cerr << "Error with SQS::GetQueueAttributes. "
                << outcome.GetError().GetMessage()
                << std::endl;

    cleanUp(topicARN,
            queueURLS,
            subscriptionARNS,
            snsClient,
            sqsClient);

    return false;
}
}

if (first) {
    std::cout
        << "An IAM policy must be attached to an SQS queue, enabling it
to receive "
        << "messages from an SNS topic." << std::endl;
}

{
    // 4. Set the SQS queue policy attribute with a policy enabling the
receipt of SNS messages.
    Aws::SQS::Model::SetQueueAttributesRequest request;
```

```

request.SetQueueUrl(queueURL);
Aws::String policy = createPolicyForQueue(queueARN, topicARN);
request.AddAttributes(Aws::SQS::Model::QueueAttributeName::Policy,
                    policy);

Aws::SQS::Model::SetQueueAttributesOutcome outcome =
    sqsClient.SetQueueAttributes(request);

if (outcome.IsSuccess()) {
    std::cout << "The attributes for the queue '" << queueName
              << "' were successfully updated." << std::endl;
}
else {
    std::cerr << "Error with SQS::SetQueueAttributes. "
              << outcome.GetError().GetMessage()
              << std::endl;

    cleanUp(topicARN,
            queueURLS,
            subscriptionARNS,
            snsClient,
            sqsClient);

    return false;
}
}

printAsterisksLine();

{
    // 5. Subscribe the SQS queue to the SNS topic.
    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("sqs");
    request.SetEndpoint(queueARN);
    if (isFifoTopic) {
        if (first) {
            std::cout << "Subscriptions to a FIFO topic can have filters."
                      << std::endl;

            std::cout
                << "If you add a filter to this subscription, then only
the filtered messages "
                << "will be received in the queue." << std::endl;
            std::cout << "For information about message filtering, "

```

```

        << "see https://docs.aws.amazon.com/sns/latest/dg/sns-
message-filtering.html"
        << std::endl;
        std::cout << "For this example, you can filter messages by a \""
        << TONE_ATTRIBUTE << "\" attribute." << std::endl;
    }

    std::ostringstream ostream;
    ostream << "Filter messages for \"" << queueName
        << "\"'s subscription to the topic \""
        << topicName << "\"? (y/n)";

    // Add filter if user answers yes.
    if (askYesNoQuestion(ostream.str())) {
        Aws::String jsonPolicy = getFilterPolicyFromUser();
        if (!jsonPolicy.empty()) {
            filteringMessages = true;

            std::cout << "This is the filter policy for this
subscription."
                << std::endl;
            std::cout << jsonPolicy << std::endl;

            request.AddAttributes("FilterPolicy", jsonPolicy);
        }
        else {
            std::cout
                << "Because you did not select any attributes, no
filter "
                << "will be added to this subscription." <<
std::endl;
        }
    }
    // if (isFifoTopic)
    Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
        std::cout << "The queue '" << queueName
            << "' has been subscribed to the topic '"
            << topicName << "'" << std::endl;
        std::cout << "with the subscription ARN '" << subscriptionARN << "."

```

```
        << std::endl;
        subscriptionARNS.push_back(subscriptionARN);
    }
    else {
        std::cerr << "Error with TopicsAndQueues::Subscribe. "
                  << outcome.GetError().GetMessage()
                  << std::endl;

        cleanUp(topicARN,
                queueURLS,
                subscriptionARNS,
                snsClient,
                sqsClient);

        return false;
    }
}

first = false;
}

first = true;
do {
    printAsterisksLine();

    // 6. Publish a message to the SNS topic.
    Aws::SNS::Model::PublishRequest request;
    request.SetTopicArn(topicARN);
    Aws::String message = askQuestion("Enter a message text to publish. ");
    request.SetMessage(message);
    if (isFifoTopic) {
        if (first) {
            std::cout
                << "Because you are using a FIFO topic, you must set a
message group ID."
                << std::endl;
            std::cout
                << "All messages within the same group will be received in
the "
                << "order they were published." << std::endl;
        }
        Aws::String messageGroupID = askQuestion(
            "Enter a message group ID for this message. ");
        request.SetMessageGroupId(messageGroupID);
```

```

        if (!contentBasedDeduplication) {
            if (first) {
                std::cout
                    << "Because you are not using content-based
deduplication, "
                    << "you must enter a deduplication ID." << std::endl;
            }
            Aws::String deduplicationID = askQuestion(
                "Enter a deduplication ID for this message. ");
            request.SetMessageDeduplicationId(deduplicationID);
        }
    }

    if (filteringMessages && askYesNoQuestion(
        "Add an attribute to this message? (y/n) ")) {
        for (size_t i = 0; i < TONES.size(); ++i) {
            std::cout << " " << (i + 1) << ". " << TONES[i] << std::endl;
        }
        int selection = askQuestionForIntRange(
            "Enter a number for an attribute. ",
            1, static_cast<int>(TONES.size()));
        Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
        messageAttributeValue.SetDataType("String");
        messageAttributeValue.SetStringValue(TONES[selection - 1]);
        request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
    }

    Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

    if (outcome.IsSuccess()) {
        std::cout << "Your message was successfully published." << std::endl;
    }
    else {
        std::cerr << "Error with TopicsAndQueues::Publish. "
            << outcome.GetError().GetMessage()
            << std::endl;

        cleanUp(topicARN,
            queueURLS,
            subscriptionARNS,
            snsClient,
            sqsClient);

        return false;
    }

```



```
    }

    first = false;
} while (askYesNoQuestion("Post another message? (y/n) "));

printAsterisksLine();

std::cout << "Now the SQS queue will be polled to retrieve the messages."
    << std::endl;
askQuestion("Press any key to continue...", alwaysTrueTest);

for (size_t i = 0; i < queueURLS.size(); ++i) {
    // 7. Poll an SQS queue for its messages.
    std::vector<Aws::String> messages;
    std::vector<Aws::String> receiptHandles;
    while (true) {
        Aws::SQS::Model::ReceiveMessageRequest request;
        request.SetMaxNumberOfMessages(10);
        request.SetQueueUrl(queueURLS[i]);

        // Setting WaitTimeSeconds to non-zero enables long polling.
        // For information about long polling, see
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
        request.SetWaitTimeSeconds(1);
        Aws::SQS::Model::ReceiveMessageOutcome outcome =
            sqsClient.ReceiveMessage(request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::SQS::Model::Message> &newMessages =
outcome.GetResult().GetMessages();
            if (newMessages.empty()) {
                break;
            }
            else {
                for (const Aws::SQS::Model::Message &message: newMessages) {
                    messages.push_back(message.GetBody());
                    receiptHandles.push_back(message.GetReceiptHandle());
                }
            }
        }
        else {
            std::cerr << "Error with SQS::ReceiveMessage. "
                << outcome.GetError().GetMessage()

```

```

        << std::endl;

        cleanUp(topicARN,
                queueURLS,
                subscriptionARNS,
                snsClient,
                sqsClient);

        return false;
    }
}

printAsterisksLine();

if (messages.empty()) {
    std::cout << "No messages were ";
}
else if (messages.size() == 1) {
    std::cout << "One message was ";
}
else {
    std::cout << messages.size() << " messages were ";
}
std::cout << "received by the queue '" << queueNames[i]
            << "'." << std::endl;
for (const Aws::String &message: messages) {
    std::cout << "  Message : '" << message << "'."
                << std::endl;
}

// 8. Delete a batch of messages from an SQS queue.
if (!receiptHandles.empty()) {
    Aws::SQS::Model::DeleteMessageBatchRequest request;
    request.SetQueueUrl(queueURLS[i]);
    int id = 1; // Ids must be unique within a batch delete request.
    for (const Aws::String &receiptHandle: receiptHandles) {
        Aws::SQS::Model::DeleteMessageBatchRequestEntry entry;
        entry.SetId(std::to_string(id));
        ++id;
        entry.SetReceiptHandle(receiptHandle);
        request.AddEntries(entry);
    }

    Aws::SQS::Model::DeleteMessageBatchOutcome outcome =

```

```

        sqsClient.DeleteMessageBatch(request);

        if (outcome.IsSuccess()) {
            std::cout << "The batch deletion of messages was successful."
                << std::endl;
        }
        else {
            std::cerr << "Error with SQS::DeleteMessageBatch. "
                << outcome.GetError().GetMessage()
                << std::endl;
            cleanUp(topicARN,
                queueURLS,
                subscriptionARNS,
                snsClient,
                sqsClient);

            return false;
        }
    }
}

return cleanUp(topicARN,
    queueURLS,
    subscriptionARNS,
    snsClient,
    sqsClient,
    true); // askUser
}

bool AwsDoc::TopicsAndQueues::cleanUp(const Aws::String &topicARN,
    const Aws::Vector<Aws::String> &queueURLS,
    const Aws::Vector<Aws::String>
    &subscriptionARNS,
    const Aws::SNS::SNSClient &snsClient,
    const Aws::SQS::SQSClient &sqsClient,
    bool askUser) {

    bool result = true;
    printAsterisksLine();
    if (!queueURLS.empty() && askUser &&
        askYesNoQuestion("Delete the SQS queues? (y/n) ")) {

        for (const auto &queueURL: queueURLS) {
            // 9. Delete an SQS queue.

```

```
Aws::SQS::Model::DeleteQueueRequest request;
request.SetQueueUrl(queueURL);

Aws::SQS::Model::DeleteQueueOutcome outcome =
    sqsClient.DeleteQueue(request);

if (outcome.IsSuccess()) {
    std::cout << "The queue with URL '" << queueURL
                << "' was successfully deleted." << std::endl;
}
else {
    std::cerr << "Error with SQS::DeleteQueue. "
                << outcome.GetError().GetMessage()
                << std::endl;
    result = false;
}
}

for (const auto &subscriptionARN: subscriptionARNS) {
    // 10. Unsubscribe an SNS subscription.
    Aws::SNS::Model::UnsubscribeRequest request;
    request.SetSubscriptionArn(subscriptionARN);

    Aws::SNS::Model::UnsubscribeOutcome outcome =
        snsClient.Unsubscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Unsubscribe of subscription ARN '" << subscriptionARN
                    << "' was successful." << std::endl;
    }
    else {
        std::cerr << "Error with TopicsAndQueues::Unsubscribe. "
                    << outcome.GetError().GetMessage()
                    << std::endl;
        result = false;
    }
}
}

printAsterisksLine();
if (!topicARN.empty() && askUser &&
    askYesNoQuestion("Delete the SNS topic? (y/n) ")) {

    // 11. Delete an SNS topic.
```

```

    Aws::SNS::Model::DeleteTopicRequest request;
    request.SetTopicArn(topicARN);

    Aws::SNS::Model::DeleteTopicOutcome outcome =
snsClient.DeleteTopic(request);

    if (outcome.IsSuccess()) {
        std::cout << "The topic with ARN '" << topicARN
            << "' was successfully deleted." << std::endl;
    }
    else {
        std::cerr << "Error with TopicsAndQueues::DeleteTopicRequest. "
            << outcome.GetError().GetMessage()
            << std::endl;
        result = false;
    }
}

return result;
}

//! Create an IAM policy that gives an SQS queue permission to receive messages from
an SNS topic.
/*!
 \sa createPolicyForQueue()
 \param queueARN: The SQS queue Amazon Resource Name (ARN).
 \param topicARN: The SNS topic ARN.
 \return Aws::String: The policy as JSON.
 */
Aws::String AwsDoc::TopicsAndQueues::createPolicyForQueue(const Aws::String
&queueARN,
                                                    const Aws::String
&topicARN) {
    std::ostringstream policyStream;
    policyStream << R"({
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "sns.amazonaws.com"
                },
                "Action": "sqs:SendMessage",
                "Resource": ")" << queueARN << R"(",
                "Condition": {

```

```
        "ArnEquals": {
            "aws:SourceArn": ")" << topicARN << R("("
        }
    }
}
]
}));

return policyStream.str();
}
```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para C++ .
 - [CreateQueue](#)
 - [CreateTopic](#)
 - [DeleteMessageBatch](#)
 - [DeleteQueue](#)
 - [DeleteTopic](#)
 - [GetQueueAttributes](#)
 - [Publish](#)
 - [ReceiveMessage](#)
 - [SetQueueAttributes](#)
 - [Subscribe](#)
 - [Unsubscribe](#)

Ejemplos de Amazon SQS usando SDK para C++

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes AWS SDK para C++ mediante Amazon SQS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Introducción](#)
- [Acciones](#)
- [Escenarios](#)

Introducción

Introducción a Amazon SQS

El siguiente ejemplo de código muestra cómo empezar a utilizar Amazon SQS.

SDK para C++

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Código para el CMake archivo 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})

```

Código del archivo de origen 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;
}

```

- Para obtener más información sobre la API, consulte [ListQueues](#) la Referencia AWS SDK para C++ de la API.

Acciones

ChangeMessageVisibility

En el siguiente ejemplo de código, se muestra cómo utilizar ChangeMessageVisibility.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

    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();
}
```

- Para obtener más información sobre la API, consulta [ChangeMessageVisibility](#) la Referencia AWS SDK para C++ de la API.

CreateQueue

En el siguiente ejemplo de código, se muestra cómo utilizar CreateQueue.

SDK para C++

 Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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();
}
```

- Para obtener más información sobre la API, consulta [CreateQueue](#) la Referencia AWS SDK para C++ de la API.

DeleteMessage

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteMessage.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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();
}
```

- Para obtener más información sobre la API, consulta [DeleteMessage](#) la Referencia AWS SDK para C++ de la API.

DeleteMessageBatch

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteMessageBatch.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
```

- Para obtener más información sobre la API, consulta [DeleteMessageBatch](#) la Referencia AWS SDK para C++ de la API.

DeleteQueue

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteQueue.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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();
}

```

- Para obtener más información sobre la API, consulta [DeleteQueue](#) la Referencia AWS SDK para C++ de la API.

GetQueueAttributes

En el siguiente ejemplo de código, se muestra cómo utilizar `GetQueueAttributes`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).


```
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;
    }
}
```

- Para obtener más información sobre la API, consulta [GetQueueAttributes](#) la Referencia AWS SDK para C++ de la API.

GetQueueUrl

En el siguiente ejemplo de código, se muestra cómo utilizar `GetQueueUrl`.

SDK para C++

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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();
}
```

- Para obtener más información sobre la API, consulta [GetQueueUrl](#) la Referencia AWS SDK para C++ de la API.

ListQueues

En el siguiente ejemplo de código, se muestra cómo utilizar `ListQueues`.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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;
}
```

- Para obtener más información sobre la API, consulta [ListQueues](#) la Referencia AWS SDK para C++ de la API.

ReceiveMessage

En el siguiente ejemplo de código, se muestra cómo utilizar ReceiveMessage.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

    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();  
}
```

- Para obtener más información sobre la API, consulta [ReceiveMessage](#) la Referencia AWS SDK para C++ de la API.

SendMessage

En el siguiente ejemplo de código, se muestra cómo utilizar SendMessage.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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();
}

```

- Para obtener más información sobre la API, consulta [SendMessage](#) la Referencia AWS SDK para C++ de la API.

SetQueueAttributes

En el siguiente ejemplo de código, se muestra cómo utilizar SetQueueAttributes.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

    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();
}

```

Configurar una cola de mensajes fallidos.

```

    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();
}

```

Configurar una cola de Amazon SQS para utilizar sondeos largos.

```

    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();
}
```

- Para obtener más información sobre la API, consulta [SetQueueAttributes](#) la Referencia AWS SDK para C++ de la API.

Escenarios

Publicación de mensajes en colas

En el siguiente ejemplo de código, se muestra cómo:

- Crear un tema (FIFO o no FIFO)
- Suscribirse a varias colas al tema con la opción de aplicar un filtro
- Publicar mensajes en el tema
- Sondear las colas en busca de los mensajes recibidos

SDK para C++

Note

Hay más información al respecto en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

    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 ostream;
ostream << "Enter a name for " << (first ? "an" : "the next")
        << " SQS queue. ";
queueName = askQuestion(ostream.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,

```



```

    }
    else {
        std::cerr << "Error with SQS::SetQueueAttributes. "
                  << outcome.GetError().GetMessage()
                  << std::endl;

        cleanUp(topicARN,
                queueURLS,
                subscriptionARNs,
                snsClient,
                sqsClient);

        return false;
    }
}

printAsterisksLine();

{
    // 5. Subscribe the SQS queue to the SNS topic.
    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("sqs");
    request.SetEndpoint(queueARN);
    if (isFifoTopic) {
        if (first) {
            std::cout << "Subscriptions to a FIFO topic can have filters."
                      << std::endl;

            std::cout
                << "If you add a filter to this subscription, then only
the filtered messages "
                << "will be received in the queue." << std::endl;
            std::cout << "For information about message filtering, "
                << "see https://docs.aws.amazon.com/sns/latest/dg/sns-
message-filtering.html"
                << std::endl;
            std::cout << "For this example, you can filter messages by a \""
                << TONE_ATTRIBUTE << "\" attribute." << std::endl;
        }

        std::ostringstream ostream;
        ostream << "Filter messages for \"" << queueName
                << "\"'s subscription to the topic \""
                << topicName << "\"? (y/n)";
    }
}

```

```

// Add filter if user answers yes.
if (askYesNoQuestion(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();
}

```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para C++ .
 - [CreateQueue](#)
 - [CreateTopic](#)
 - [DeleteMessageBatch](#)
 - [DeleteQueue](#)
 - [DeleteTopic](#)
 - [GetQueueAttributes](#)
 - [Publish](#)
 - [ReceiveMessage](#)
 - [SetQueueAttributes](#)
 - [Subscribe](#)
 - [Unsubscribe](#)

AWS STS ejemplos de uso de SDK para C++

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso del AWS SDK para C++ with AWS STS.

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Acciones](#)

Acciones

AssumeRole

En el siguiente ejemplo de código, se muestra cómo utilizar AssumeRole.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
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();
}
```

- Para obtener más información sobre la API, consulta [AssumeRole](#) la Referencia AWS SDK para C++ de la API.

Ejemplos de Amazon Transcribe Streaming con SDK para C++

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante Amazon Transcribe Streaming. AWS SDK para C++

Las acciones son extractos de código de programas más grandes y deben ejecutarse en contexto. Mientras las acciones muestran cómo llamar a las distintas funciones de servicio, es posible ver las acciones en contexto en los escenarios relacionados.

Los escenarios son ejemplos de código que muestran cómo llevar a cabo una tarea específica a través de llamadas a varias funciones dentro del servicio o combinado con otros Servicios de AWS.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas


- [Acciones](#)
- [Escenarios](#)

Acciones

StartStreamTranscription

En el siguiente ejemplo de código, se muestra cómo utilizar `StartStreamTranscription`.

SDK para C++

 Note

Hay más información. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
int main() {
    Aws::SDKOptions options;

    Aws::InitAPI(options);
    {
        //TODO(User): Set to the region of your AWS account.
        const Aws::String region = Aws::Region::US_WEST_2;

        //Load a profile that has been granted AmazonTranscribeFullAccess AWS
        managed permission policy.
        Aws::Client::ClientConfiguration config;
#ifdef _WIN32
        // ATTENTION: On Windows with the AWS C++ SDK, this example only runs if the
        SDK is built
        // with the curl library.
        // For more information, see the accompanying ReadMe.
        // For more information, see "Building the SDK for Windows with curl".
        // https://docs.aws.amazon.com/sdk-for-cpp/v1/developer-guide/setup-
        windows.html
        //TODO(User): Update to the location of your .crt file.
        config.caFile = "C:/curl/bin/curl-ca-bundle.crt";
#endif
        config.region = region;

        TranscribeStreamingServiceClient client(config);
        StartStreamTranscriptionHandler handler;
        handler.SetOnErrorCallback(
            [](const Aws::Client::AWSError<TranscribeStreamingServiceErrors>
&error) {
                std::cerr << "ERROR: " + error.GetMessage() << std::endl;
            });
        //SetTranscriptEventCallback called for every 'chunk' of file transcribed.
        // Partial results are returned in real time.
        handler.SetTranscriptEventCallback([](const TranscriptEvent &ev) {
```

```

        for (auto &&r: ev.GetTranscript().GetResults()) {
            if (r.GetIsPartial()) {
                std::cout << "[partial] ";
            }
            else {
                std::cout << "[Final] ";
            }
            for (auto &&alt: r.GetAlternatives()) {
                std::cout << alt.GetTranscript() << std::endl;
            }
        }
    });

    StartStreamTranscriptionRequest request;
    request.SetMediaSampleRateHertz(SAMPLE_RATE);
    request.SetLanguageCode(LanguageCode::en_US);
    request.SetMediaEncoding(
        MediaEncoding::pcm); // wav and aiff files are PCM formats.
    request.SetEventStreamHandler(handler);

    auto OnStreamReady = [](AudioStream &stream) {
        Aws::FStream file(FILE_NAME, std::ios_base::in |
std::ios_base::binary);
        if (!file.is_open()) {
            std::cerr << "Failed to open " << FILE_NAME << '\n';
        }
        std::array<char, BUFFER_SIZE> buf;
        int i = 0;
        while (file) {
            file.read(&buf[0], buf.size());

            if (!file)
                std::cout << "File: only " << file.gcount() << " could be
read"
                    << std::endl;

            Aws::Vector<unsigned char> bits{buf.begin(), buf.end()};
            AudioEvent event(std::move(bits));
            if (!stream) {
                std::cerr << "Failed to create a stream" << std::endl;
                break;
            }
            //The std::basic_istream::gcount() is used to count the
characters in the given string. It returns

```

```

        //the number of characters extracted by the last read()
operation.
        if (file.gcount() > 0) {
            if (!stream.WriteAudioEvent(event)) {
                std::cerr << "Failed to write an audio event" <<
std::endl;
                break;
            }
        }
        else {
            break;
        }
        std::this_thread::sleep_for(std::chrono::milliseconds(
            25)); // Slow down because we are streaming from a file.
    }
    if (!stream.WriteAudioEvent(
        AudioEvent())) {
        // Per the spec, we have to send an empty event (an event
without a payload) at the end.
        std::cerr << "Failed to send an empty frame" << std::endl;
    }
    else {
        std::cout << "Successfully sent the empty frame" << std::endl;
    }
    stream.flush();
    stream.Close();
};

    Aws::Utils::Threading::Semaphore signaling(0 /*initialCount*/, 1 /
*maxCount*/);
    auto OnResponseCallback = [&signaling](
        const TranscribeStreamingServiceClient * /*unused*/,
        const Model::StartStreamTranscriptionRequest & /*unused*/,
        const Model::StartStreamTranscriptionOutcome &outcome,
        const std::shared_ptr<const Aws::Client::AsyncCallerContext> & /
*unused*/) {

        if (!outcome.IsSuccess()) {
            std::cerr << "Transcribe streaming error "
                << outcome.GetError().GetMessage() << std::endl;
        }

        signaling.Release();
};

```



```
        std::cout << "Starting..." << std::endl;
        client.StartStreamTranscriptionAsync(request, OnStreamReady,
        OnResponseCallback,
                                     nullptr /*context*/);
        signaling.WaitOne(); // Prevent the application from exiting until we're
done.
        std::cout << "Done" << std::endl;
    }

    Aws::ShutdownAPI(options);

    return 0;
}
```

- Para obtener más información sobre la API, consulta [StartStreamTranscription](#) la Referencia AWS SDK para C++ de la API.

Escenarios

Transcripción de un archivo de audio

El siguiente ejemplo de código muestra cómo generar una transcripción de un archivo de audio de origen mediante Amazon Transcribe Streaming.

SDK para C++

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
int main() {
    Aws::SDKOptions options;

    Aws::InitAPI(options);
    {
        //TODO(User): Set to the region of your AWS account.
        const Aws::String region = Aws::Region::US_WEST_2;
```

```
    //Load a profile that has been granted AmazonTranscribeFullAccess AWS
managed permission policy.
    Aws::Client::ClientConfiguration config;
#ifdef _WIN32
    // ATTENTION: On Windows with the AWS C++ SDK, this example only runs if the
SDK is built
    // with the curl library.
    // For more information, see the accompanying ReadMe.
    // For more information, see "Building the SDK for Windows with curl".
    // https://docs.aws.amazon.com/sdk-for-cpp/v1/developer-guide/setup-
windows.html
    //TODO(User): Update to the location of your .crt file.
    config.caFile = "C:/curl/bin/curl-ca-bundle.crt";
#endif

    config.region = region;

    TranscribeStreamingServiceClient client(config);
    StartStreamTranscriptionHandler handler;
    handler.SetOnErrorCallback(
        [](const Aws::Client::AWSError<TranscribeStreamingServiceErrors>
&error) {
            std::cerr << "ERROR: " + error.GetMessage() << std::endl;
        });
    //SetTranscriptEventCallback called for every 'chunk' of file transcribed.
    // Partial results are returned in real time.
    handler.SetTranscriptEventCallback([](const TranscriptEvent &ev) {
        for (auto &&r: ev.GetTranscript().GetResults()) {
            if (r.GetIsPartial()) {
                std::cout << "[partial] ";
            }
            else {
                std::cout << "[Final] ";
            }
            for (auto &&alt: r.GetAlternatives()) {
                std::cout << alt.GetTranscript() << std::endl;
            }
        }
    });

    StartStreamTranscriptionRequest request;
    request.SetMediaSampleRateHertz(SAMPLE_RATE);
    request.SetLanguageCode(LanguageCode::en_US);
    request.SetMediaEncoding(
```

```

        MediaEncoding::pcm); // wav and aiff files are PCM formats.
request.SetEventStreamHandler(handler);

auto OnStreamReady = [](AudioStream &stream) {
    Aws::FStream file(FILE_NAME, std::ios_base::in |
std::ios_base::binary);
    if (!file.is_open()) {
        std::cerr << "Failed to open " << FILE_NAME << '\n';
    }
    std::array<char, BUFFER_SIZE> buf;
    int i = 0;
    while (file) {
        file.read(&buf[0], buf.size());

        if (!file)
            std::cout << "File: only " << file.gcount() << " could be
read"
                << std::endl;

        Aws::Vector<unsigned char> bits{buf.begin(), buf.end()};
        AudioEvent event(std::move(bits));
        if (!stream) {
            std::cerr << "Failed to create a stream" << std::endl;
            break;
        }
        //The std::basic_istream::gcount() is used to count the
characters in the given string. It returns
//the number of characters extracted by the last read()
operation.

        if (file.gcount() > 0) {
            if (!stream.WriteAudioEvent(event)) {
                std::cerr << "Failed to write an audio event" <<
std::endl;
                break;
            }
        }
        else {
            break;
        }
        std::this_thread::sleep_for(std::chrono::milliseconds(
25)); // Slow down because we are streaming from a file.
    }
    if (!stream.WriteAudioEvent(
        AudioEvent())) {

```

```

        // Per the spec, we have to send an empty event (an event
        without a payload) at the end.
        std::cerr << "Failed to send an empty frame" << std::endl;
    }
    else {
        std::cout << "Successfully sent the empty frame" << std::endl;
    }
    stream.flush();
    stream.Close();
};

    Aws::Utils::Threading::Semaphore signaling(0 /*initialCount*/, 1 /*
*maxCount*/);
    auto OnResponseCallback = [&signaling](
        const TranscribeStreamingServiceClient * /*unused*/,
        const Model::StartStreamTranscriptionRequest & /*unused*/,
        const Model::StartStreamTranscriptionOutcome &outcome,
        const std::shared_ptr<const Aws::Client::AsyncCallerContext> & /*
*unused*/) {

        if (!outcome.IsSuccess()) {
            std::cerr << "Transcribe streaming error "
                << outcome.GetError().GetMessage() << std::endl;
        }

        signaling.Release();
    };

    std::cout << "Starting..." << std::endl;
    client.StartStreamTranscriptionAsync(request, OnStreamReady,
    OnResponseCallback,
                                nullptr /*context*/);
    signaling.WaitOne(); // Prevent the application from exiting until we're
done.
    std::cout << "Done" << std::endl;
}

    Aws::ShutdownAPI(options);

    return 0;
}

```

- Para obtener más información sobre la API, consulta [StartStreamTranscription](#) la Referencia AWS SDK para C++ de la API.

Seguridad para AWS SDK para C++

La seguridad en la nube de Amazon Web Services (AWS) es la máxima prioridad. Como cliente de AWS, se beneficia de una arquitectura de red y un centro de datos que se han diseñado para satisfacer los requisitos de seguridad de las organizaciones más exigentes. La seguridad es una responsabilidad compartida entre usted AWS y usted. En el [modelo de responsabilidad compartida](#), se habla de “seguridad de la nube” y “seguridad en la nube”:

Seguridad de la nube: AWS se encarga de proteger la infraestructura en la que se ejecutan todos los servicios que se ofrecen en la AWS nube y de proporcionarle servicios que pueda utilizar de forma segura. Nuestra responsabilidad en materia de seguridad es nuestra máxima prioridad AWS, y auditores externos comprueban y verifican periódicamente la eficacia de nuestra seguridad como parte de los [programas de AWS conformidad](#).

Seguridad en la nube: su responsabilidad viene determinada por el AWS servicio que utilice y otros factores, como la confidencialidad de sus datos, los requisitos de su organización y las leyes y reglamentos aplicables.

Este AWS producto o servicio sigue el [modelo de responsabilidad compartida](#) a través de los servicios específicos de Amazon Web Services (AWS) a los que da soporte. Para obtener información sobre la seguridad de los AWS servicios, consulte la [página de documentación sobre la seguridad del AWS servicio](#) y [AWS los servicios que se encuentran dentro del ámbito de aplicación de AWS las medidas de conformidad establecidas por el programa de conformidad](#).

Temas

- [Protección de datos en AWS SDK for C++](#)
- [Identity and Access Management](#)
- [Validación de la conformidad de este producto o servicio de AWS](#)
- [Resiliencia de este producto o servicio de AWS](#)
- [Seguridad de la infraestructura de este producto o servicio de AWS](#)
- [Aplicación de una versión de TLS mínima en AWS SDK para C++](#)
- [Migración del cliente de cifrado Amazon S3 \(V1 a V2\)](#)
- [Migración del cliente de cifrado Amazon S3 \(V2 a V3\)](#)

Protección de datos en AWS SDK for C++

El [modelo de responsabilidad compartida](#) de AWS se aplica a la protección de datos en AWS SDK for C++. Como se describe en este modelo, AWS es responsable de proteger la infraestructura global que ejecuta toda la Nube de AWS. Eres responsable de mantener el control sobre el contenido alojado en esta infraestructura. También eres responsable de las tareas de administración y configuración de seguridad para los Servicios de AWS que utiliza. Para obtener más información sobre la privacidad de los datos, consulta las [Preguntas frecuentes sobre la privacidad de datos](#). Para obtener información sobre la protección de datos en Europa, consulta la publicación de blog sobre el [Modelo de responsabilidad compartida de AWS y GDPR](#) en el Blog de seguridad de AWS.

Con fines de protección de datos, recomendamos proteger las credenciales de la Cuenta de AWS y configurar cuentas de usuario individuales con AWS IAM Identity Center o AWS Identity and Access Management (IAM). De esta manera, solo se otorgan a cada usuario los permisos necesarios para cumplir sus obligaciones laborales. También recomendamos proteger sus datos de la siguiente manera:

- Utiliza la autenticación multifactor (MFA) en cada cuenta.
- Utiliza SSL/TLS para comunicarse con los recursos de AWS. Se recomienda el uso de TLS 1.2 y recomendamos TLS 1.3.
- Configure los registros de API y de actividad de los usuarios con AWS CloudTrail. Para obtener información sobre cómo utilizar registros de seguimiento de CloudTrail para capturar actividades de AWS, consulta [Working with CloudTrail trails](#) en la Guía del usuario de AWS CloudTrail.
- Utiliza las soluciones de cifrado de AWS, junto con todos los controles de seguridad predeterminados dentro de los servicios de Servicios de AWS.
- Utiliza servicios de seguridad administrados avanzados, como Amazon Macie, que lo ayuden a detectar y proteger los datos confidenciales almacenados en Amazon S3.
- Si necesita módulos criptográficos validados FIPS 140-3 al acceder a AWS a través de una interfaz de línea de comandos o una API, utiliza un punto de conexión de FIPS. Para obtener más información sobre los puntos de conexión de FIPS disponibles, consulta [Estándar de procesamiento de la información federal \(FIPS\) 140-3](#).

Se recomienda encarecidamente no introducir nunca información confidencial o sensible, como por ejemplo, direcciones de correo electrónico de clientes, en etiquetas o campos de formato libre, tales como el campo Nombre. Incluye las situaciones en las que debe trabajar con SDK para C++ u otros Servicios de AWS a través de la consola, la API, la AWS CLI u otros SDK de AWS. Cualquier

dato que ingrese en etiquetas o campos de texto de formato libre utilizados para nombres se puede emplear para los registros de facturación o diagnóstico. Si proporciona una URL a un servidor externo, recomendamos encarecidamente que no incluya información de credenciales en la URL a fin de validar la solicitud para ese servidor.

Identity and Access Management

AWS Identity and Access Management (IAM) es un Servicio de AWS que ayuda a los administradores a controlar de forma segura el acceso a los recursos de AWS. Los administradores de IAM controlan quién está autenticado (ha iniciado sesión) y autorizado (tiene permisos) para utilizar recursos de AWS. IAM es un Servicio de AWS que se puede utilizar sin cargo adicional.

Temas

- [Público](#)
- [Autenticación con identidades](#)
- [Administración de acceso mediante políticas](#)
- [Cómo funcionan los Servicios de AWS con IAM](#)
- [Solución de problemas de identidades y accesos en AWS](#)

Público

La forma en que utilice AWS Identity and Access Management (IAM) difiere en función del trabajo que realice en AWS.

Usuario de servicio: si utiliza Servicios de AWS para realizar el trabajo, el administrador le proporciona las credenciales y los permisos que necesita. A medida que utilice más características de AWS para realizar su trabajo, es posible que necesite permisos adicionales. Entender cómo se administra el acceso puede ayudarle a solicitar los permisos correctos al administrador. Si no puede acceder a una característica en AWS, consulte [Solución de problemas de identidades y accesos en AWS](#) o la Guía del usuario del Servicio de AWS que esté usando.

Administrador de servicio: si está a cargo de los recursos de AWS en su empresa, probablemente tenga acceso completo a AWS. Su trabajo consiste en determinar a qué características y recursos de AWS deben acceder los usuarios del servicio. A continuación, debe enviar solicitudes a su administrador de IAM para cambiar los permisos de los usuarios de sus servicios. Revise la información de esta página para conocer los conceptos básicos de IAM. Para obtener más

información sobre cómo la empresa puede utilizar IAM con AWS, consulte la Guía del usuario del Servicio de AWS que esté usando.

Administrador de IAM: si es un administrador de IAM, es posible que quiera conocer más detalles sobre cómo escribir políticas para administrar el acceso a AWS. Para consultar ejemplos de políticas basadas en la identidad de AWS que puede utilizar en IAM, consulte la Guía del usuario del Servicio de AWS que esté usando.

Autenticación con identidades

La autenticación es la manera de iniciar sesión en AWS mediante credenciales de identidad. Debe estar autenticado como Usuario raíz de la cuenta de AWS, un usuario de IAM o asumiendo un rol de IAM.

Puede iniciar sesión como una identidad federada con las credenciales de un origen de identidad, como AWS IAM Identity Center (IAM Identity Center), la autenticación de inicio de sesión único o las credenciales de Google/Facebook. Para obtener más información sobre el inicio de sesión, consulte [Cómo iniciar sesión en la Cuenta de AWS](#) en la Guía del usuario de AWS Sign-In.

Para el acceso programático, AWS proporciona un SDK y una CLI para firmar criptográficamente las solicitudes. Para obtener más información, consulte [AWS Signature Version 4 para solicitudes de API](#) en la Guía del usuario de IAM.

Usuario raíz de la Cuenta de AWS

Cuando se crea una Cuenta de AWS, se comienza con una identidad de inicio de sesión llamada Cuenta de AWS del usuario raíz que tiene acceso completo a todos los Servicios de AWS y recursos. Recomendamos encarecidamente que no utilice el usuario raíz para las tareas diarias. Para ver las tareas que requieren credenciales de usuario raíz, consulte [Tareas que requieren credenciales de usuario raíz](#) en la Guía del usuario de IAM.

Identidad federada

Como práctica recomendada, exija a los usuarios humanos que utilicen la federación con un proveedor de identidades para acceder a Servicios de AWS con credenciales temporales.

Una identidad federada es un usuario del directorio empresarial, proveedor de identidades web o Directory Service que accede a Servicios de AWS mediante credenciales de un origen de identidad. Las identidades federadas asumen roles que proporcionan credenciales temporales.

Para una administración de acceso centralizada, le recomendamos AWS IAM Identity Center. Para obtener más información, consulte [¿Qué es el Centro de identidades de IAM?](#) en la Guía del usuario de AWS IAM Identity Center.

Usuarios y grupos de IAM

Un [usuario de IAM](#) es una identidad con permisos específicos para una sola persona o aplicación. Recomendamos el uso de credenciales temporales en lugar de usuarios de IAM con credenciales a largo plazo. Para obtener más información, consulte [Solicitar que los usuarios humanos utilicen la federación con un proveedor de identidades para acceder a AWS mediante credenciales temporales](#) en la Guía del usuario de IAM.

Un [grupo de IAM](#) especifica un conjunto de usuarios de IAM y facilita la administración de los permisos para grupos grandes de usuarios. Para obtener más información, consulte [Casos de uso para usuarios de IAM](#) en la Guía del usuario de IAM.

Roles de IAM

Un [rol de IAM](#) es una identidad con permisos específicos que proporciona credenciales temporales. Puede asumir un rol [cambiando de un usuario a un rol de IAM \(consola\)](#) o llamando a una operación de la API de la AWS CLI o AWS. Para obtener más información, consulte [Métodos para asumir un rol](#) en la Guía del usuario de IAM.

Los roles de IAM son útiles para el acceso de usuarios federados, los permisos de usuario de IAM temporales, el acceso entre cuentas, el acceso entre servicios y las aplicaciones que se ejecutan en Amazon EC2. Para obtener más información, consulte [Acceso a recursos entre cuentas en IAM](#) en la Guía del usuario de IAM.

Administración de acceso mediante políticas

Para controlar el acceso en AWS, se crean políticas y se adjuntan a identidades o recursos de AWS. Una política define permisos cuando se asocia con una identidad o recurso. AWS evalúa estas políticas cuando una entidad principal realiza una solicitud. La mayoría de las políticas se almacenan en AWS como documentos JSON. Para obtener más información sobre los documentos de política JSON, consulte [Información general de políticas JSON](#) en la Guía del usuario de IAM.

Mediante políticas, los administradores especifican quién tiene acceso a qué definiendo qué entidad principal puede realizar acciones sobre qué recursos y en qué condiciones.

De forma predeterminada, los usuarios y los roles no tienen permisos. Un administrador de IAM crea políticas de IAM y las agrega a los roles, que luego los usuarios pueden asumir. Las políticas de IAM definen permisos independientemente del método utilizado para realizar la operación.

Políticas basadas en identidades

Las políticas basadas en identidad son documentos JSON de políticas de permisos que asocia a una identidad (usuario, grupo o rol). Estas políticas controlan qué acciones pueden realizar las identidades, en qué recursos y en qué condiciones. Para obtener más información sobre cómo crear una política basada en identidad, consulte [Creación de políticas de IAM](#) en la Guía del usuario de IAM.

Las políticas basadas en la identidad pueden ser políticas integradas (incrustadas directamente en una sola identidad) o políticas administradas (políticas independientes asociadas a varias identidades). Para obtener información sobre cómo elegir entre políticas administradas e insertadas, consulte [Elegir entre políticas administradas y políticas insertadas](#) en la Guía del usuario de IAM.

Políticas basadas en recursos

Las políticas basadas en recursos son documentos de políticas JSON que se asocian a un recurso. Los ejemplos incluyen las políticas de confianza de roles de IAM y las políticas de bucket de Amazon S3. En los servicios que admiten políticas basadas en recursos, los administradores de servicios pueden utilizarlos para controlar el acceso a un recurso específico. Debe [especificar una entidad principal](#) en una política en función de recursos.

Las políticas basadas en recursos son políticas insertadas que se encuentran en ese servicio. No se puede utilizar políticas de IAM administradas de AWS en una política basada en recursos.

Listas de control de acceso (ACL)

Las listas de control de acceso (ACL) controlan qué entidades principales (miembros de cuentas, usuarios o roles) tienen permisos para acceder a un recurso. Las ACL son similares a las políticas basadas en recursos, aunque no utilizan el formato de documento de políticas JSON.

Amazon S3, AWS WAF y Amazon VPC son ejemplos de servicios que admiten las ACL. Para obtener más información sobre las ACL, consulta [Información general de Lista de control de acceso \(ACL\)](#) en la Guía para desarrolladores de Amazon Simple Storage Service.

Otros tipos de políticas

AWS admite tipos de políticas adicionales que pueden establecer el máximo de permisos concedidos por los tipos de políticas más comunes:

- Límites de permisos: establecen los permisos máximos que una política basada en la identidad puede conceder a una entidad de IAM. Para obtener más información, consulte [Límites de permisos para las entidades de IAM](#) en la Guía del usuario de IAM.
- Políticas de control de servicios (SCP): especifican los permisos máximos para una organización o unidad organizativa en AWS Organizations. Para obtener más información, consulte [Políticas de control de servicios](#) en la Guía del usuario de AWS Organizations.
- Políticas de control de recursos (RCP): establecen los permisos máximos disponibles para los recursos de las cuentas. Para obtener más información, consulte [Políticas de control de recursos \(RCP\)](#) en la Guía del usuario de AWS Organizations.
- Políticas de sesión: políticas avanzadas que se pasan como parámetro cuando se crea una sesión temporal para un rol o un usuario federado. Para más información, consulte [Políticas de sesión](#) en la Guía del usuario de IAM.

Varios tipos de políticas

Cuando se aplican varios tipos de políticas a una solicitud, los permisos resultantes son más complicados de entender. Para obtener información acerca de cómo AWS decide si permitir o no una solicitud cuando hay varios tipos de políticas implicados, consulte [Lógica de evaluación de políticas](#) en la Guía del usuario de IAM.

Cómo funcionan los Servicios de AWS con IAM

Para obtener una perspectiva general sobre cómo funcionan los Servicios de AWS con la mayoría de las características de IAM, consulte [Servicios de AWS que funcionan con IAM](#) en la Guía del usuario de IAM.

Para obtener información sobre cómo usar un Servicio de AWS específico con IAM, consulte la sección de seguridad de la Guía del usuario del servicio correspondiente.

Solución de problemas de identidades y accesos en AWS

Utilice la siguiente información para diagnosticar y solucionar los problemas comunes que puedan surgir cuando trabaje con AWS e IAM.

Temas

- [No tengo autorización para realizar una acción en AWS](#)
- [No tengo autorización para realizar la operación iam:PassRole](#)
- [Quiero permitir a personas externas a mi Cuenta de AWS el acceso a mis recursos de AWS](#)

No tengo autorización para realizar una acción en AWS

Si recibe un error que indica que no tiene autorización para realizar una acción, las políticas se deben actualizar para permitirle realizar la acción.

En el siguiente ejemplo, el error se produce cuando el usuario de IAM mateojackson intenta utilizar la consola para consultar los detalles acerca de un recurso ficticio *my-example-widget*, pero no tiene los permisos ficticios *awes:GetWidget*.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
awes:GetWidget on resource: my-example-widget
```

En este caso, la política del usuario mateojackson debe actualizarse para permitir el acceso al recurso *my-example-widget* mediante la acción *awes:GetWidget*.

Si necesita ayuda, póngase en contacto con su gestor de AWS. El gestor es la persona que le proporcionó las credenciales de inicio de sesión.

No tengo autorización para realizar la operación iam:PassRole

Si recibe un error que indica que no tiene autorización para realizar la acción *iam:PassRole*, las políticas deben actualizarse a fin de permitirle pasar un rol a AWS.

Algunos Servicios de AWS le permiten transferir un rol existente a dicho servicio en lugar de crear un nuevo rol de servicio o uno vinculado al servicio. Para ello, debe tener permisos para transferir el rol al servicio.

En el siguiente ejemplo, el error se produce cuando un usuario de IAM denominado marymajor intenta utilizar la consola para realizar una acción en AWS. Sin embargo, la acción requiere que el servicio cuente con permisos que otorguen un rol de servicio. Mary no tiene permisos para transferir el rol al servicio.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
iam:PassRole
```

En este caso, las políticas de Mary se deben actualizar para permitirle realizar la acción `iam:PassRole`.

Si necesita ayuda, póngase en contacto con su gestor de AWS. El gestor es la persona que le proporcionó las credenciales de inicio de sesión.

Quiero permitir a personas externas a mi Cuenta de AWS el acceso a mis recursos de AWS

Puede crear un rol que los usuarios de otras cuentas o las personas externas a la organización puedan utilizar para acceder a sus recursos. Puede especificar una persona de confianza para que asuma el rol. En el caso de los servicios que admitan las políticas basadas en recursos o las listas de control de acceso (ACL), puede utilizar dichas políticas para conceder a las personas acceso a sus recursos.

Para obtener más información, consulte lo siguiente:

- Para obtener información acerca de si AWS admite estas características, consulte [Cómo funcionan los Servicios de AWS con IAM](#).
- Para obtener información acerca de cómo proporcionar acceso a los recursos de las Cuenta de AWS de su propiedad, consulte [Proporcionar acceso a un usuario de IAM a otra Cuentas de AWS de la que es propietario](#) en la Guía del usuario de IAM.
- Para obtener información acerca de cómo proporcionar acceso a tus recursos a Cuentas de AWS de terceros, consulta [Proporcionar acceso a Cuentas de AWS que son propiedad de terceros](#) en la Guía del usuario de IAM.
- Para obtener información sobre cómo proporcionar acceso mediante una federación de identidades, consulta [Proporcionar acceso a usuarios autenticados externamente \(identidad federada\)](#) en la Guía del usuario de IAM.
- Para conocer sobre la diferencia entre las políticas basadas en roles y en recursos para el acceso entre cuentas, consulte [Acceso a recursos entre cuentas en IAM](#) en la Guía del usuario de IAM.

Validación de la conformidad de este producto o servicio de AWS

Para saber si un Servicio de AWS está incluido en el ámbito de programas de conformidad específicos, consulte [Servicios de AWS incluidos por programa de conformidad](#) y escoja el programa de conformidad que le interese. Para obtener información general, consulte [Programas de conformidad de AWS](#).

Puedes descargar los informes de auditoría de terceros utilizando AWS Artifact. Para obtener más información, consulte [Descarga de informes en AWS Artifact](#).

Su responsabilidad de conformidad al utilizar Servicios de AWS se determina en función de la confidencialidad de los datos, los objetivos de conformidad de su empresa, así como de la legislación y los reglamentos aplicables. Para obtener más información sobre la responsabilidad de conformidad al usar Servicios de AWS, consulte la [Documentación de seguridad de AWS](#).

Este producto o servicio de AWS sigue el [modelo de responsabilidad compartida](#) a través de los servicios específicos de Amazon Web Services (AWS) que admite. Para obtener información sobre la seguridad de los servicios de AWS, consulte la [página de documentación sobre la seguridad de los servicios de AWS](#) y los [servicios de AWS sujetos a las medidas de conformidad de AWS de cada programa de conformidad](#).

Resiliencia de este producto o servicio de AWS

La infraestructura global de AWS está conformada por Regiones de AWS y zonas de disponibilidad.

Las Regiones de AWS proporcionan varias zonas de disponibilidad físicamente independientes y aisladas que se encuentran conectadas mediante redes con un alto nivel de rendimiento y redundancia, además de baja latencia.

Con las zonas de disponibilidad, puede diseñar y utilizar aplicaciones y bases de datos que realizan una conmutación por error automática entre las zonas sin interrupciones. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de centros de datos únicos o múltiples.

Para obtener más información sobre las regiones y zonas de disponibilidad de AWS, consulte [Infraestructura global de AWS](#).

Este producto o servicio de AWS sigue el [modelo de responsabilidad compartida](#) a través de los servicios específicos de Amazon Web Services (AWS) que admite. Para obtener información sobre la seguridad de los servicios de AWS, consulte la [página de documentación sobre la seguridad de los servicios de AWS](#) y los [servicios de AWS sujetos a las medidas de conformidad de AWS de cada programa de conformidad](#).

Seguridad de la infraestructura de este producto o servicio de AWS

Este producto o servicio de AWS utiliza servicios administrados y, por lo tanto, está protegido por la seguridad de red global de AWS. Para obtener información sobre los servicios de seguridad de

AWS y cómo AWS protege la infraestructura, consulte [Seguridad en la nube de AWS](#). Para diseñar su entorno de AWS con las prácticas recomendadas de seguridad de la infraestructura, consulte [Protección de la infraestructura](#) en Portal de seguridad de AWS Well-Architected Framework.

Puede utilizar llamadas a la API publicadas en AWS para acceder a este producto o servicio de AWS a través de la red. Los clientes deben admitir lo siguiente:

- Seguridad de la capa de transporte (TLS). Exigimos TLS 1.2 y recomendamos TLS 1.3.
- Conjuntos de cifrado con confidencialidad directa total (PFS) como DHE (Ephemeral Diffie-Hellman) o ECDHE (Elliptic Curve Ephemeral Diffie-Hellman). La mayoría de los sistemas modernos como Java 7 y posteriores son compatibles con estos modos.

Además, las solicitudes deben estar firmadas mediante un ID de clave de acceso y una clave de acceso secreta que esté asociada a una entidad principal de IAM. También puede utilizar [AWS Security Token Service](#) (AWS STS) para generar credenciales de seguridad temporales para firmar solicitudes.

Este producto o servicio de AWS sigue el [modelo de responsabilidad compartida](#) a través de los servicios específicos de Amazon Web Services (AWS) que admite. Para obtener información sobre la seguridad de los servicios de AWS, consulte la [página de documentación sobre la seguridad de los servicios de AWS](#) y los [servicios de AWS sujetos a las medidas de conformidad de AWS de cada programa de conformidad](#).

Aplicación de una versión de TLS mínima en AWS SDK para C++

Para aumentar la seguridad al comunicarse con los servicios de AWS, debe configurar SDK for C++ para usar TLS 1.2 o una versión posterior. Nosotros recomendamos TLS 1.3.

El AWS SDK para C++ es una biblioteca multiplataforma. Puede crear y ejecutar la aplicación en las plataformas que desee. Las diferentes plataformas pueden depender de distintos clientes HTTP subyacentes.

macOS, Linux, Android y otras plataformas distintas de Windows utilizan de forma predeterminada [libcurl](#). Si la versión de libcurl es posterior a la 7.34.0, TLS 1.0 es la versión mínima que utilizan los clientes HTTP subyacentes.

En Windows, la biblioteca predeterminada es [WinHTTP](#). Windows decide el protocolo que se utilizará entre los protocolos TLS 1.0, TLS 1.1, TLS 1.2 y TLS 1.3 disponibles. [WinInet](#) e [IXMLHttpRequest2](#)

son las otras dos opciones disponibles en Windows. Puedes configurar tu aplicación para reemplazar la biblioteca predeterminada durante CMake y en tiempo de ejecución. Para estos dos clientes HTTP, Windows también decide el protocolo seguro.

El AWS SDK para C++ también proporciona la flexibilidad necesaria para anular los clientes HTTP predeterminados. Por ejemplo, puede aplicar libcurl o usar los clientes HTTP que desee utilizando una fábrica de clientes HTTP personalizada. Por lo tanto, para usar TLS 1.2 como versión mínima, debe conocer la biblioteca de clientes HTTP que está usando.

Aplicación de una versión específica de TLS con libcurl en todas las plataformas

En esta sección se asume que el AWS SDK para C++ utiliza libcurl como una dependencia para la compatibilidad con el protocolo HTTP. Para especificar de forma explícita la versión de TLS, necesitará una versión de libcurl mínima de 7.34.0. Además, es posible que tenga que modificar el código fuente del AWS SDK para C++ y volver a crearlo después.

En este procedimiento, se muestra cómo llevar a cabo esas tareas.

Para aplicar TLS 1.2 con libcurl

1. Compruebe que su instalación de libcurl tenga al menos la versión 7.34.0.
2. Descargue el código de fuente para el AWS SDK para C++ en GitHub.
3. Abra `aws-cpp-sdk-core/source/http/curl/CurlHttpClient.cpp` y busque las líneas siguientes.

```
#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. Si es necesario, cambie el último parámetro de la llamada a la función de la siguiente manera.

```
#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. Si ha realizado los cambios de código anteriores, compile e instale el AWS SDK para C++ según las instrucciones de <https://github.com/aws/aws-sdk-cpp#building-the-sdk>.
6. En el cliente de servicio de la aplicación, habilite `verifySSL` en su configuración, si esta opción no se ha habilitado todavía.

Para aplicar TLS 1.3 con libcurl

Para aplicar el protocolo TLS 1.3, siga los pasos de la sección anterior para configurar la opción `CURL_SSLVERSION_TLSv1_3` en lugar de `CURL_SSLVERSION_TLSv1_2`.

Aplicación de una versión específica de TLS en Windows

Los siguientes procedimientos muestran cómo aplicar TLS 1.2 o TLS 1.3 con WinHTTP, WinInet o `IXMLHTTPRequest2`.

Requisito previo: determine la compatibilidad con TLS en Windows

- Determine la versión del protocolo TLS compatible con su sistema siguiendo la descripción incluida en <https://docs.microsoft.com/en-us/windows/win32/secauthn/protocols-in-tls-ssl-schannel-ssp->.
- Si está ejecutando en Windows 7 SP1 o Windows Server 2008 R2 SP1, debe asegurarse de que la compatibilidad con TLS 1.2 esté habilitada en el registro, como se describe en <https://docs.microsoft.com/en-us/windows-server/security/tls/tls-registry-settings#tls-12>. Si está ejecutando una distribución anterior, debe actualizar el sistema operativo.

Para aplicar TLS 1.2 o TLS 1.3 con WinHTTP

WinHTTP proporciona una API para establecer de forma explícita los protocolos seguros aceptables. Sin embargo, para poder configurar esto en tiempo de ejecución, debe modificar el código fuente del AWS SDK para C++ y volver a crearlo después.

1. Descargue el código de fuente para el AWS SDK para C++ en GitHub.
2. Abra `aws-cpp-sdk-core/source/http/windows/WinHttpSyncHttpClient.cpp` y busque las líneas siguientes.

```
#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());
}
```

El indicador de la opción `WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_3` se define si TLS 1.3 está presente en el sistema de compilación actual. Para obtener más información, consulte [WINHTTP_OPTION_SECURE_PROTOCOLS](#) y [Compatibilidad con las versiones del protocolo TLS](#) en el sitio web de Microsoft.

3. Elija una de las opciones siguientes:

- Para aplicar TLS 1.2:

Según la directiva `#else`, cambie el valor de la variable `flags` del siguiente modo.

```
DWORD flags = WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_2;
```

- Para aplicar TLS 1.3:

Según la directiva `#if defined(WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_3)`, cambie el valor de la variable `flags` del siguiente modo.

```
DWORD flags = WINHTTP_FLAG_SECURE_PROTOCOL_TLS1_3;
```

4. Si ha realizado los cambios de código anteriores, compile e instale el AWS SDK para C++ según las instrucciones de <https://github.com/aws/aws-sdk-cpp#building-the-sdk>.
5. En el cliente de servicio de la aplicación, habilite `verifySSL` en su configuración, si esta opción no se ha habilitado todavía.

Para aplicar TLS 1.2 con WinInet e IXMLHTTPRequest2

No hay ninguna API que especifique el protocolo seguro para las bibliotecas WinInet e IXMLHTTPRequest2. Por lo tanto, AWS SDK para C++ usa el protocolo predeterminado para el sistema operativo. Puede actualizar el registro de Windows para forzar el uso de TLS 1.2, como se muestra en el siguiente procedimiento. Sin embargo, tenga en cuenta que el resultado es un cambio global que afecta a todas las aplicaciones que dependen de Schannel.

1. Abra el Editor del Registro y vaya a `Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols`.
2. Cree las siguientes subclaves, si no existen: TLS 1.0, TLS 1.1, y TLS 1.2.
3. En cada una de las subclaves, cree una subclave Client y otra Server.
4. Cree las siguientes claves y valores.

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

Observe que `TLS 1.2\Client\Enabled` es la única clave que está establecida en 1. Si se establece esta clave en 1, hace que TLS 1.2 sea el único protocolo seguro aceptable.

Para aplicar TLS 1.3 con WinInet e IXMLHTTPRequest2

No hay ninguna API que especifique el protocolo seguro para las bibliotecas WinInet e IXMLHTTPRequest2. Por lo tanto, AWS SDK para C++ usa el protocolo predeterminado para el sistema operativo. Puede actualizar el registro de Windows para forzar el uso de TLS 1.3, como se muestra en el siguiente procedimiento. Sin embargo, tenga en cuenta que el resultado es un cambio global que afecta a todas las aplicaciones que dependen de Schannel.

1. Abra el Editor del Registro y vaya a `Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols`.
2. Cree las siguientes subclaves, si no existen: TLS 1.0, TLS 1.1, TLS 1.2 y TLS 1.3.
3. En cada una de las subclaves, cree una subclave Client y otra Server.

4. Cree las siguientes claves y valores.

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

Observe que TLS 1.3\Client\Enabled es la única clave que está establecida en 1. Si se establece esta clave en 1, hace que TLS 1.3 sea el único protocolo seguro aceptable.

Migración del cliente de cifrado Amazon S3 (V1 a V2)

Note

Si utiliza la versión 2 del cliente de cifrado Amazon S3 y desea migrar a la versión 3, consulte [Migración del cliente de cifrado Amazon S3 \(V2 a V3\)](#).

En este tema se muestra cómo migrar las aplicaciones de la versión 1 (V1) del cliente de cifrado Amazon Simple Storage Service (Amazon S3) a la versión 2 (V2) y cómo garantizar la disponibilidad de las aplicaciones durante todo el proceso de migración.

Información general sobre la migración

Esta migración se produce en dos fases:

1. Actualice los clientes existentes para leer nuevos formatos. En primer lugar, implemente una versión actualizada de AWS SDK para C++ en su aplicación. Esto permite a los clientes de cifrado de la versión V1 descifrar los objetos escritos por los nuevos clientes de la versión V2. Si su aplicación usa varios AWS SDKs, debe actualizar cada SDK por separado.

2. Migre los clientes de cifrado y descifrado a la versión V2. Una vez que todos sus clientes de cifrado de la versión 1 puedan leer los nuevos formatos, puede migrar los clientes de cifrado y descifrado existentes a sus respectivas versiones de la versión 2.

Actualizar los clientes existentes para leer nuevos formatos

Primero debe actualizar los clientes existentes a la versión más reciente del SDK. Tras completar este paso, los clientes V1 de su aplicación podrán descifrar los objetos cifrados por los clientes de cifrado V2 sin actualizar la base de código de la aplicación.

Cree e instale la versión más reciente de AWS SDK para C++

Aplicaciones que utilizan el SDK desde el código fuente

Si compila e instalas la fuente AWS SDK para C++ desde la fuente, descarga o clona la fuente del SDK desde [aws/aws-sdk-cpp](#) una GitHub. Luego, repita los pasos de compilación e instalación normales.

Si vas AWS SDK para C++ a actualizar desde una versión anterior a la 1.8.x, consulta este [REGISTRO DE CAMBIOS](#) para ver los [cambios más importantes](#) introducidos en cada versión principal. Para obtener más información sobre cómo compilar e instalar, consulte [AWS SDK para C++ + Obtención del AWS SDK para C++ desde el código fuente](#)

Aplicaciones que utilizan el SDK desde Vcpkg

Si su aplicación utiliza [Vcpkg](#) para realizar un seguimiento de las actualizaciones del SDK, simplemente use su método de actualización de Vcpkg existente para actualizar el SDK a la versión más reciente. Tenga en cuenta que hay un retraso entre el momento en que se publica una versión y el momento en que está disponible a través de un administrador de paquetes. La versión más reciente siempre está disponible en la [instalación desde el código fuente](#).

Ejecute el siguiente comando para actualizar el paquete `aws-sdk-cpp`:

```
vcpkg upgrade aws-sdk-cpp
```

Y compruebe la versión del paquete `aws-sdk-cpp`:

```
vcpkg list aws-sdk-cpp
```

La versión debe ser como mínimo la 1.8.24.

Para obtener más información sobre el uso de Vcpkg con AWS SDK para C++, consulte. [Obtención del AWS SDK para C++ de un administrador de paquetes](#)

Creación, instalación e implementación de sus aplicaciones

Si su aplicación se enlaza de forma estática con el AWS SDK para C++, no es necesario realizar cambios en el código de la aplicación, pero debe volver a compilarla para utilizar los cambios más recientes del SDK. Este paso no es necesario en el caso de la vinculación dinámica.

Tras actualizar la versión dependiente de la aplicación y comprobar su funcionalidad, proceda a implementar la aplicación en su flota. Una vez completada la implementación de la aplicación, puede pasar a la siguiente fase de migración de la aplicación para utilizar la versión V2 de los clientes de cifrado y descifrado.

Migrar clientes de cifrado y descifrado a la versión V2

En los siguientes pasos, se muestra cómo migrar correctamente el código de la versión V1 a la V2 del cliente de cifrado de Amazon S3. Dado que es necesario realizar cambios en el código, tendrá que volver a crear la aplicación, independientemente de si se vincula estática o dinámicamente con ella. AWS SDK para C++

Uso de los nuevos materiales de cifrado

Con los clientes de cifrado de Amazon S3 V2 y la configuración criptográfica V2, los siguientes materiales de cifrado han quedado obsoletos:

- `SimpleEncryptionMaterials`
- `KMSEncryptionMaterials`

Se han sustituido por los siguientes materiales de cifrado seguro:

- `SimpleEncryptionMaterialsWithGCMAAD`
- `KMSWithContextEncryptionMaterials`

Para crear un cliente de cifrado S3 V2, se requieren los siguientes cambios de código:

- Si utiliza **KMSEncryptionMaterials** al crear un cliente de cifrado de S3:
 - Al crear un cliente de cifrado de S3 V2, sustituya `KMSEncryptionMaterials` por `KMSWithContextEncryptionMaterials` y especifíquelo en la configuración criptográfica V2.
 - Al colocar un objeto con clientes de cifrado de Amazon S3 V2, debe proporcionar de manera explícita un mapa de contexto de cadena-cadena como contexto de KMS para cifrar la CEK. Puede ser un mapa vacío.
- Si utiliza **SimpleEncryptionMaterials** al crear un cliente de cifrado de S3:
 - Al crear un cliente de cifrado de Amazon S3 V2, sustituya `SimpleEncryptionMaterials` por `SimpleEncryptionMaterialsWithGCMAAD` y especifíquelo en la configuración criptográfica V2.
 - Al colocar un objeto con clientes de cifrado Amazon S3 V2, debe proporcionar de manera explícita un mapa de contexto de cadena-cadena; de lo contrario, el SDK devolverá un error.

Ejemplo: uso del algoritmo KMS/KMSWithContext Key Wrap

Antes de la migración (encapsulamiento de claves de KMS)

```
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);
```

Tras la migración (empaquetado de claves KMSWith contextuales)

```
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
*/);
```


Ejemplo: uso del algoritmo de encapsulamiento de claves AES/AES-GCM

Antes de la migración (encapsulamiento de claves AES)

```
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);
```

Después de la migración (encapsulamiento de claves AES-GCM)

```
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 */);
```

Ejemplos adicionales

Los ejemplos siguientes muestran cómo abordar casos prácticos específicos relacionados con la migración de la V1 a la V2.

Descifrado de objetos cifrados por clientes de cifrado de Amazon S3 antiguos

De forma predeterminada, no puede usar el cliente de cifrado Amazon S3 V2 para descifrar objetos que se cifraron con algoritmos de ajuste de claves obsoletos o esquemas criptográficos de contenido obsoletos.

Han quedado obsoletos los siguientes algoritmos de empaquetado de claves:

- KMS
- AES_KEY_WRAP

Y han quedado obsoletos los siguientes esquemas criptográficos de contenido:

- CBC

- CTR

Si utiliza clientes de cifrado de Amazon S3 antiguos AWS SDK para C++ para cifrar los objetos, es probable que utilice los métodos obsoletos si:

- Ha utilizado `SimpleEncryptionMaterials` o `KMSEncryptionMaterials`.
- Ha utilizado `ENCRYPTION_ONLY` como `Crypto Mode` en su configuración criptográfica.

Para utilizar el cliente de cifrado Amazon S3 V2 para descifrar objetos cifrados mediante algoritmos de encapsulamiento de claves obsoletos o esquemas criptográficos de contenido obsoletos, debe anular el valor predeterminado de `SecurityProfile` en la configuración criptográfica V2 de V2 a `V2_AND_LEGACY`.

Ejemplo

Antes de la migración

```
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);
```

Después de la migración

```
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);
```

Descifrado de objetos con un rango

Con los clientes de cifrado de Amazon S3 antiguos, puede especificar el rango de bytes que se van a recibir al descifrar un objeto de S3. En el cliente de cifrado Amazon S3 V2, esta función está configurada como `DISABLED` de manera predeterminada. Por lo tanto, debe anular el valor

predeterminado de `RangeGetMode` de `DISABLED` a `ALL` en la configuración criptográfica de la versión 2.

Ejemplo

Antes de la migración

```
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);
```

Después de la migración

```
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);
```

Descifrado de objetos con cualquier CMK

Al descifrar objetos que se han cifrado con `KMSWithContextEncryptionMaterials`, los clientes de cifrado de Amazon S3 V2 pueden permitir que KMS encuentre la CMK adecuada proporcionando una clave maestra vacía. Esta característica está configurada `DISABLED` como de forma predeterminada. Debe configurarla de forma explícita llamando a `SetKMSEncryptWithAnyCMK(true)` para sus materiales de cifrado de KMS.

Ejemplo

Antes de la migración

```
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);
```

Después de la migración

```
auto materials = Aws::MakeShared<KMSWithContextEncryptionMaterials>("s3EncryptionV2",  
    ""/* provide an empty KMS Master Key*/);  
materials.SetKMSThroughAnyCMK(true);  
CryptoConfigurationV2 cryptoConfig(materials);  
S3EncryptionClientV2 encryptionClient(cryptoConfig);  
// Code snippet here to setup the getObjectRequest object.  
encryptionClient.GetObject(getObjectRequest);
```

Para obtener el código completo de todos estos escenarios de migración, consulte el [ejemplo de cifrado de Amazon S3](#) en Github.

Migración del cliente de cifrado Amazon S3 (V2 a V3)

Note

Si utiliza la versión 1 del cliente de cifrado Amazon S3, primero debe migrar a la versión 2 antes de migrar a la versión 3. Consulte [Migración del cliente de cifrado Amazon S3 \(V1 a V2\)](#).

En este tema se muestra cómo migrar las aplicaciones de la versión 2 (V2) a la versión 3 (V3) del cliente de cifrado Amazon Simple Storage Service (Amazon S3) y cómo garantizar la disponibilidad de las aplicaciones durante todo el proceso de migración. La versión 3 presenta el `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` algoritmo y las políticas de compromiso para mejorar la seguridad mediante la protección contra la manipulación de las claves de datos en los archivos de instrucciones.

Información general sobre la migración

Esta migración se produce en dos fases:

1. Actualice los clientes existentes para leer nuevos formatos. En primer lugar, implemente una versión actualizada de AWS SDK para C++ en su aplicación. Esto permite a los clientes de cifrado

V2 existentes descifrar los objetos escritos por los nuevos clientes V3. Si su aplicación usa varios AWS SDKs, debe actualizar cada SDK por separado.

2. Migre los clientes de cifrado y descifrado a la versión 3. Una vez que todos sus clientes de cifrado V2 puedan leer nuevos formatos, puede migrar los clientes de cifrado y descifrado existentes a sus respectivas versiones V3.

Comprensión de los conceptos de la V3

La versión 3 del cliente de cifrado Amazon S3 presenta nuevas características de seguridad que mejoran la protección contra la manipulación de claves de datos. Comprender estos conceptos es esencial para una migración exitosa.

Política de compromiso

Las políticas de compromiso controlan la forma en que el cliente de cifrado gestiona el compromiso de claves durante las operaciones de cifrado y descifrado. La versión 3 ofrece tres opciones de políticas para admitir diferentes escenarios de migración y requisitos de seguridad:

FORBID_ENCRYPT_ALLOW_DECRYPT

Comportamiento de cifrado: cifra los objetos sin compromiso de clave, utilizando los mismos algoritmos que la V2.

Comportamiento de descifrado: permite descifrar objetos cifrados con y sin compromiso de clave.

Implicaciones de seguridad: esta política no impone ningún compromiso con la clave y puede permitir la manipulación de la clave de datos cifrada de los archivos de instrucciones. Utilice esta política solo durante la fase de migración inicial cuando necesite que los clientes de la versión 2 lean los objetos recién cifrados.

Compatibilidad de versiones: los objetos cifrados con esta política se pueden leer en todas las implementaciones de las versiones 2 y 3.

REQUIRE_ENCRYPT_ALLOW_DECRYPT (predeterminado)

Comportamiento de cifrado: cifra los objetos con un compromiso de clave mediante el algoritmo. `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY`

Comportamiento de descifrado: permite descifrar tanto los objetos cifrados con compromiso de clave como los objetos cifrados sin compromiso de clave.

Implicaciones de seguridad: esta política proporciona una seguridad sólida para los objetos recién cifrados y, al mismo tiempo, mantiene la compatibilidad con versiones anteriores para leer objetos antiguos. Esta es la política recomendada para la mayoría de los escenarios de migración.

Compatibilidad de versiones: los objetos cifrados con esta política solo pueden leerse en la versión 3 y en las implementaciones más recientes de la versión 2. Los clientes de la versión 2 no pueden descifrar estos objetos. Sin embargo, los clientes V3 que utilizan esta política aún pueden descifrar los objetos cifrados por los clientes V2.

REQUIRE_ENCRYPT_REQUIRE_DECRYPT

Comportamiento de cifrado: cifra los objetos con un compromiso de clave mediante el algoritmo. `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY`

Comportamiento de descifrado: solo permite descifrar objetos cifrados con compromiso de clave. Rechaza los objetos cifrados sin compromiso de clave.

Implicaciones de seguridad: esta política proporciona el más alto nivel de seguridad al imponer un compromiso clave en todas las operaciones. Utilice esta política solo después de que todos los objetos se hayan vuelto a cifrar con la clave y ya no necesite leer los objetos cifrados V1 o V2 antiguos.

Compatibilidad de versiones: los objetos cifrados con esta política solo pueden leerse en la versión 3 y en las implementaciones más recientes de la versión 2. Además, los clientes que utilizan esta política no pueden descifrar los objetos cifrados por los clientes V1 o V2.

Consideraciones sobre la migración: durante la migración, comience por `FORBID_ENCRYPT_ALLOW_DECRYPT` si necesita que los clientes de la versión 2 lean los objetos nuevos y, a continuación, continúe con esa opción `REQUIRE_ENCRYPT_ALLOW_DECRYPT` una vez que todos los clientes se hayan actualizado a la versión 3. Por último, considérela `REQUIRE_ENCRYPT_REQUIRE_DECRYPT` solo después de volver a cifrar todos los objetos antiguos.

Algoritmo `ALG_AES_256_GCM_HKDF__COMMIT_KEY_SHA512`

El `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` algoritmo es un nuevo algoritmo de cifrado introducido en la versión 3 que proporciona una seguridad mejorada para las claves de datos cifradas almacenadas en los archivos de instrucciones.

Impacto en los archivos de instrucciones: el `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` algoritmo solo afecta a los archivos de instrucciones, que son objetos S3 independientes que

almacenan metadatos de cifrado, incluida la clave de datos cifrados. Este cambio de algoritmo no afecta a los objetos que almacenan metadatos de cifrado en metadatos de objetos (el método de almacenamiento predeterminado).

Protección contra la manipulación: el `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` algoritmo protege contra la manipulación de las claves de datos al vincular criptográficamente la clave de datos cifrada al contexto de cifrado. Esto evita que los atacantes sustituyan una clave de datos cifrados diferente en el archivo de instrucciones, lo que podría provocar el descifrado por una clave no deseada.

Compatibilidad de versiones: los objetos cifrados con el `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` algoritmo solo se pueden descifrar mediante las implementaciones de la versión 3 y las versiones de transición más recientes del SDK, que incluyen soporte para el descifrado de la versión 3.

Warning

Importante: Antes de habilitar el cifrado con el `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY` algoritmo (mediante políticas `REQUIRE_ENCRYPT_ALLOW_DECRYPT` o políticas de `REQUIRE_ENCRYPT_REQUIRE_DECRYPT` compromiso), debe asegurarse de que todos los clientes que lean estos objetos se hayan actualizado a la V3 o a la última versión de transición a la V2 que admita el descifrado de la V3. Si no se actualizan primero todos los lectores, se producirán errores al descifrar los objetos recién cifrados.

Actualizar los clientes existentes para leer nuevos formatos

Primero debe actualizar los clientes existentes a la versión más reciente del SDK. Tras completar este paso, los clientes V2 de su aplicación podrán descifrar los objetos cifrados por los clientes de cifrado V3 sin actualizar la base de código de la aplicación.

Cree e instale la última versión de AWS SDK para C++

Aplicaciones que utilizan el SDK desde el código fuente

Si compila e instalas la fuente AWS SDK para C++ desde la fuente, descarga o clona la fuente del SDK desde [aws/aws-sdk-cpp](https://github.com/aws/aws-sdk-cpp) una GitHub. Luego, repita los pasos de compilación e instalación normales.

Si vas a actualizar el AWS SDK para C++ desde una versión anterior a la 1.11.x, consulta este [REGISTRO DE CAMBIOS](#) para ver los [cambios más importantes](#) introducidos en cada versión principal. Para obtener más información sobre cómo compilar e instalar, consulte [AWS SDK para C++ + Obtención del AWS SDK para C++ desde el código fuente](#)

Aplicaciones que utilizan el SDK desde Vcpkg

Si su aplicación utiliza [Vcpkg](#) para realizar un seguimiento de las actualizaciones del SDK, simplemente use su método de actualización de Vcpkg existente para actualizar el SDK a la versión más reciente. Tenga en cuenta que hay un retraso entre el momento en que se publica una versión y el momento en que está disponible a través de un administrador de paquetes. La versión más reciente siempre está disponible en la [instalación desde el código fuente](#).

Ejecute el siguiente comando para actualizar el paquete `aws-sdk-cpp`:

```
vcpkg upgrade aws-sdk-cpp
```

Y compruebe la versión del paquete `aws-sdk-cpp`:

```
vcpkg list aws-sdk-cpp
```

La versión debe ser al menos la 1.11.x para poder descifrar objetos cifrados en la versión 3.

Para obtener más información sobre el uso de Vcpkg con, consulte [AWS SDK para C++ Obtención del AWS SDK para C++ de un administrador de paquetes](#)

Creación, instalación e implementación de sus aplicaciones

Si su aplicación se enlaza de forma estática con el AWS SDK para C++, no es necesario realizar cambios en el código de la aplicación, pero debe volver a compilarla para utilizar los cambios más recientes del SDK. Este paso no es necesario en el caso de la vinculación dinámica.

Tras actualizar la versión dependiente de la aplicación y comprobar su funcionalidad, proceda a implementar la aplicación en su flota. Una vez completada la implementación de la aplicación, puede continuar con la siguiente fase de migración de la aplicación para utilizar los clientes de cifrado y descifrado V3.

Migre los clientes de cifrado y descifrado a la V3

Los siguientes pasos le muestran cómo migrar correctamente el código de la V2 a la V3 del cliente de cifrado Amazon S3. Dado que es necesario realizar cambios en el código, tendrá que volver a

crear la aplicación, independientemente de si se vincula estática o dinámicamente con ella. AWS SDK para C++

Uso de clientes de cifrado V3

La V3 introduce la `S3EncryptionClientV3` clase y reemplaza `CryptoConfigurationV3` a los equivalentes de la V2. Las principales diferencias en la V3 son:

- La V3 usa `KMSWithContextEncryptionMaterials` (igual que la V2) pero requiere una configuración explícita en `CryptoConfigurationV3`
- Todas `PutObject` las operaciones requieren un mapa de contexto de cifrado (puede estar vacío).
- La versión 3 introduce políticas de compromiso para controlar el comportamiento de cifrado y descifrado.
- De forma predeterminada, la V3 cifra con clave de compromiso mediante el algoritmo. `ALG_AES_256_GCM_HKDF_SHA512_COMMIT_KEY`
- La API de configuración de descifrado del algoritmo antiguo cambia de a.
`config.SetSecurityProfile(SecurityProfile::V2_AND_LEGACY);`
`config.AllowLegacy();`

Ejemplo: migración de la V2 a la V3 con cifrado KMS

Antes de la migración (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);
```

Durante la migración (versión 3 con compatibilidad con versiones anteriores)

```
// 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);
```

Después de la migración (versión 3 con un compromiso clave)

```
// 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);
```

Ejemplos adicionales

En esta sección se proporcionan ejemplos adicionales para configurar las opciones del cliente de cifrado de la V3 para que sean compatibles con diversos escenarios y requisitos de migración.

Habilitación de Legacy Support

Los clientes de la versión 3 pueden descifrar los objetos cifrados por los clientes de la versión 2 solo si utilizan las políticas `REQUIRE_ENCRYPT_ALLOW_DECRYPT` o las políticas de `FORBID_ENCRYPT_ALLOW_DECRYPT` compromiso. Sin embargo, si necesita descifrar los objetos cifrados por los clientes de la versión 1, debe habilitar explícitamente la compatibilidad con versiones antiguas mediante este método. `AllowLegacy()`

Cuándo utilizar el soporte antiguo:

- Tiene objetos en S3 que se cifraron con la versión 1 del cliente de cifrado de S3.
- Debe leer estos objetos cifrados en la versión 1 con el cliente de la versión 3 durante el proceso de migración.
- Está utilizando la política de compromiso `REQUIRE_ENCRYPT_ALLOW_DECRYPT` o `FORBID_ENCRYPT_ALLOW_DECRYPT`

Warning

El soporte antiguo solo debe habilitarse temporalmente durante la migración. Una vez que todos los objetos de la V1 se hayan vuelto a cifrar con la V2 o la V3, deshabilite la compatibilidad con la versión anterior para garantizar la máxima seguridad.

Ejemplo: activación de Legacy Support

```
// 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);
```

Configuración del método de almacenamiento

El cliente de cifrado de S3 puede almacenar los metadatos de cifrado de dos maneras: como metadatos de objetos (opción predeterminada) o en un archivo de instrucciones independiente. Puede configurar el método de almacenamiento mediante el `SetStorageMethod()` método `activoCryptoConfigurationV3`.

Opciones de método de almacenamiento:

METADATA (predeterminado)

Los metadatos de cifrado se almacenan en los encabezados de metadatos del objeto. Este es el método más común y práctico, ya que toda la información de cifrado se almacena con el propio objeto.

Cuándo usarlo: utilice este método en la mayoría de los escenarios. Simplifica la administración de objetos, ya que los metadatos de cifrado viajan con el objeto.

INSTRUCTION_FILE

Los metadatos de cifrado se almacenan en un objeto S3 independiente (el archivo de instrucciones) con el sufijo `.instruction`.

Cuándo usarlo: utilice este método cuando el tamaño de los metadatos del objeto sea un problema o cuando necesite separar los metadatos de cifrado del objeto cifrado. Tenga en cuenta que el uso de archivos de instrucciones requiere administrar dos objetos de S3 (el objeto cifrado y su archivo de instrucciones) en lugar de uno.

Ejemplo: configuración del método de almacenamiento

```
// 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

Cuando utilice el método `INSTRUCTION_FILE` de almacenamiento, recuerde que al eliminar el objeto cifrado no se elimina automáticamente el archivo de instrucciones. Debe gestionar ambos objetos por separado.

Historial de documentos de la Guía para AWS SDK para C++ desarrolladores

En este tema se enumeran los cambios importantes en la Guía para AWS SDK para C++ desarrolladores. Para obtener notificaciones sobre las actualizaciones de esta documentación, puede suscribirse a una [fuente RSS](#).

Cambio	Descripción	Fecha
Opciones de configuración del cliente	Se ha actualizado la lista de variables de configuración del cliente.	8 de septiembre de 2025
Actualizaciones de la administración de memoria y del índice	Se han actualizado los parámetros de administración de memoria a la versión más reciente. Tabla de contenido estandarizada para que sea más coherente en todas partes AWS SDKs.	14 de marzo de 2025
Migración del cliente de cifrado Amazon S3 V3	Se agregó información sobre la migración de la V2 a la V3 del cliente de cifrado Amazon S3.	2 de diciembre de 2024
libcrypto personalizado	Se ha agregado contenido para el uso personalizado de libcrypto. Se ha eliminado la limitación CMake máxima. Se actualizaron CMake los parámetros disponibles.	20 de febrero de 2024
Índice	Se ha actualizado el índice para que los ejemplos de código sean más accesibles.	1 de junio de 2023

[Actualizaciones de las prácticas recomendadas de IAM](#)

Se ha actualizado la guía para implementar las prácticas recomendadas de IAM. Para obtener más información, consulta [prácticas recomendadas de seguridad en IAM](#).

1 de marzo de 2023

[Eliminación de nuget](#)

Se elimina la mención de nuget como una opción viable del administrador de paquetes porque la última versión disponible es demasiado antigua.

2 de diciembre de 2022

[Actualizaciones de la sección de introducción](#)

Se actualizó el contenido de vcpkg para comunicar claramente lo que no es compatible con una opción externa AWS y que es una opción externa. Se han actualizado las instrucciones sobre cómo compilar el SDK para Windows con curl.

18 de octubre de 2022

[Actualizaciones de ClientConfiguration](#)

Se ha actualizado la estructura de ClientConfiguration para reflejar con precisión la API más reciente.

22 de septiembre de 2022

[Mejoras en la sección de introducción](#)

Se ha mejorado la claridad de las instrucciones en la sección de introducción sobre cómo compilar el SDK en Windows y Linux.

24 de junio de 2022

Compatibilidad de curl con Windows	Se han agregado notas sobre cómo compilar el SDK para Windows con curl.	15 de junio de 2022
Retirarse: clásico EC2	Se agregaron notas sobre la jubilación de -Classic EC2.	13 de abril de 2022
Activación de métricas de SDK	Se ha eliminado la información sobre cómo habilitar métricas de SDK, que ha quedado en desuso.	20 de enero de 2022
Trabajando con servicios AWS	Se incluyen listas de los ejemplos de código que están disponibles GitHub en el repositorio de ejemplos de código.	11 de enero de 2022
Mejoras	Mejoras en la sección de introducción, en la sección de ejemplos de código y en la estandarización general.	9 de junio de 2021
Actualización de la versión	Se ha reflejado la actualización de la versión de lanzamiento general del SDK a la 1.9.	20 de abril de 2021
Cómo empezar a usar el AWS SDK para C++	Sección actualizada con una nueva organización y detalles.	17 de marzo de 2021
Migración de clientes de cifrado a Amazon S3	Se ha agregado información sobre cómo migrar las aplicaciones desde la versión 1 a la versión 2 del cliente de cifrado de Amazon S3.	7 de agosto de 2020

<u>Contenido de seguridad</u>	Se ha añadido contenido de seguridad.	6 de febrero de 2020
<u>Creación, listado y eliminación de buckets</u>	Se actualizó el CreateBucket ejemplo de Amazon S3 para que sea compatible Regiones de AWS.	20 de junio de 2019
<u>Instrucciones de compilación</u>	Se han actualizado las instrucciones de compilación del SDK.	16 de abril de 2019
<u>Métodos asíncronos</u>	Se ha agregado una nueva sección.	16 de abril de 2019
<u>Clases de clientes de servicio</u>	Varias actualizaciones.	5 de abril de 2019
<u>Administración de permisos de acceso de Amazon S3</u>	Varias actualizaciones.	3 de abril de 2019
<u>Actualizaciones relacionadas con la compilación y las variables de configuración</u>	Se han actualizado las instrucciones para compilar el SDK. Se actualizaron las variables de configuración de AWS cliente disponibles.	1 de marzo de 2019
<u>Gestor de paquetes C++ vcpkg</u>	Se han actualizado las instrucciones para configurar el administrador de paquetes de C++ vcpkg.	19 de enero de 2019

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la versión original de inglés, prevalecerá la versión en inglés.