



Guía para desarrolladores

AWS SDK para Go v2



AWS SDK para Go v2: 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 AWS SDK para Go v2?	1
Mantenimiento y compatibilidad de las versiones principales del SDK	1
Introducción	2
Obtención de una cuenta de Amazon	2
Instalación de AWS SDK para Go v2	2
Obtener las claves de acceso de AWS	3
Para obtener su ID de clave de acceso y clave de acceso secreta	3
Temas relacionados	4
Invocación de una operación	4
Configurar el SDK	6
Cargando archivos de configuración AWS compartidos	6
Especificar la AWS región	7
Configuración de la región con una variable de entorno	7
Especificación de la región mediante programación	7
Especificación de credenciales	7
Roles de IAM para las tareas	9
Funciones de IAM para Amazon Instances EC2	9
Credenciales y configuración compartidas	9
Variables de entorno	12
Especificación de credenciales mediante programación	13
Autenticación	17
Definiciones	17
Flujo de trabajo de resolución de esquemas de autenticación	20
AuthScheme compatibles de forma nativa	21
Puntos de conexión del cliente	24
Personalización	24
V2: EndpointResolverV2 + BaseEndpoint	25
V1: EndpointResolver	29
Migración	32
Cliente HTTP	36
Reemplazo durante la carga de la configuración	36
Tiempo de espera	36
Dialer	37
Transporte	38

Interceptores	41
Interceptores frente a middleware	41
Enlaces de interceptor disponibles	42
Registro de interceptores	43
Configuración global del interceptor	43
Registro	44
Logger	44
ClientLogMode	45
Reintentos y tiempos de espera	45
Elemento de reintento estándar	46
NopRetryer	46
Personalización del comportamiento	46
Tiempos de espera	51
Uso del SDK	53
Creación de un cliente de servicio	53
NewFromConfig	53
New	54
Llamada a operaciones de servicio	56
Transferencia de parámetros a una operación de servicio	56
Reemplazo de opciones de cliente para la llamada a operaciones	57
Gestión de respuestas de operación	58
Uso simultáneo de clientes de servicio	60
Uso de paginadores de operaciones	62
Uso de esperadores	64
Reemplazo de la configuración de un esperador	65
Reemplazo avanzado de la configuración de un esperador	67
Gestión de errores en el SDK	68
Registro de errores	68
Errores del cliente de servicio	69
Recuperación de identificadores de solicitud	70
Uso de los servicios de AWS	72
Sumas de comprobación de Amazon S3	72
Cargar un objeto	73
Descargar un objeto	76
Utilidades del SDK	78
Utilidades de Amazon RDS	78

IAM Authentication	78
Utilidades de Amazon CloudFront	79
Firmante de URL de Amazon CloudFront	79
Servicio de metadatos de instancias de Amazon EC2	80
Utilidades de Amazon S3	81
Administradores de transferencias de Amazon S3	81
Entrada de transmisión no buscable	91
Middleware	93
Escritura de un middleware personalizado	94
Cómo asociar middleware a todos los clientes	96
Cómo asociar middleware a una operación específica	96
Transferencia de metadatos descendente en la pila	97
Metadatos proporcionados por el SDK	99
Transferencia de metadatos ascendente en la pila	99
Preguntas frecuentes	102
¿Cómo se configura el cliente HTTP de mi SDK? ¿Existen directrices o prácticas recomendadas?	102
¿Cómo se configuran los tiempos de espera de las operaciones?	102
Las solicitudes que realiza el SDK agotan el tiempo de espera o tardan demasiado, ¿cómo lo soluciono?	103
¿Cómo puedo solucionar un error <code>read: connection reset</code> ?	103
¿Por qué recibo errores de “firma no válida” al usar un proxy HTTP con el SDK?	104
Tiempos de las operaciones del SDK	104
Pruebas unitarias	111
Simulación de operaciones de cliente	111
Simulación de paginadores	113
Ejemplos de código	116
API Gateway	117
AWS contribuciones de la comunidad	117
Aurora	118
Introducción	118
Conceptos básicos	120
Acciones	138
Amazon Bedrock	158
Introducción	118
Acciones	138

Amazon Bedrock Runtime	162
Introducción	118
Escenarios	165
Amazon Titan Image Generator	168
Anthropic Claude	171
CloudFormation	178
Acciones	138
CloudWatch Registros	180
Acciones	138
Amazon Cognito Identity Provider	183
Introducción	118
Acciones	138
Escenarios	165
Amazon DocumentDB	265
Ejemplos de tecnología sin servidor	265
DynamoDB	267
Conceptos básicos	120
Acciones	138
Escenarios	165
Ejemplos de tecnología sin servidor	265
AWS contribuciones de la comunidad	117
IAM	340
Introducción	118
Conceptos básicos	120
Acciones	138
Kinesis	402
Ejemplos de tecnología sin servidor	265
Lambda	405
Introducción	118
Conceptos básicos	120
Acciones	138
Escenarios	165
Ejemplos de tecnología sin servidor	265
AWS contribuciones de la comunidad	117
Amazon MSK	517
Ejemplos de tecnología sin servidor	265

Centro de socios	519
Acciones	138
Amazon RDS	522
Introducción	118
Conceptos básicos	120
Acciones	138
Ejemplos de tecnología sin servidor	265
Amazon Redshift	558
Introducción	118
Conceptos básicos	120
Acciones	138
Amazon S3	577
Introducción	118
Conceptos básicos	120
Acciones	138
Escenarios	165
Ejemplos de tecnología sin servidor	265
Amazon SNS	668
Introducción	118
Acciones	138
Escenarios	165
Ejemplos de tecnología sin servidor	265
Amazon SQS	696
Introducción	118
Acciones	138
Escenarios	165
Ejemplos de tecnología sin servidor	265
Migre a la v2	729
Versión mínima de Go	729
Modularización	729
Carga de la configuración	730
Ejemplos	27
Simulación e *iface	733
Credenciales y proveedores de credenciales	734
Credenciales estáticas	14
Credenciales de rol de Amazon EC2 IAM	736

Credenciales de los puntos de conexión	737
Credenciales de proceso	737
AWS Security Token Service Credenciales	738
Clientes de servicio	740
Creación de clientes	741
Puntos de conexión	743
Autenticación	744
Invocación de operaciones de la API	744
Tipos de datos de servicio	745
Parámetros de puntero	746
Tipos de errores	746
Paginadores	748
Waiters	749
Solicitudes prefiradas	750
Personalización de solicitudes	752
Entrada/salida de la operación	752
Solicitud/respuesta HTTP	756
Fases del controlador	758
Características	760
Servicio de metadatos de EC2 instancias de Amazon	760
Amazon S3 Transfer Manager	761
Utilidades de CloudFront firma de Amazon	762
Cliente de cifrado de Amazon S3	762
Cambios en las personalizaciones de servicios	762
Amazon S3	762
Seguridad	765
Protección de los datos	765
Validación de conformidad	767
Resiliencia	767
Historial de revisión	768
.....	dcclxix

¿Qué es AWS SDK para Go v2?

AWS SDK para Go v2 proporciona diversas API y utilidades que los desarrolladores pueden emplear para crear aplicaciones Go que utilicen servicios de AWS, como por ejemplo, Amazon Elastic Compute Cloud (Amazon EC2) y Amazon Simple Storage Service (Amazon S3).

El SDK elimina la complejidad de escribir directamente el código en una interfaz de servicio web. Oculta muchas de las operaciones básicas, como la autenticación, los reintentos de solicitud y la gestión de errores.

El SDK también incluye utilidades eficaces. Por ejemplo, el administrador de cargas y descargas de Amazon S3 puede dividir automáticamente objetos grandes en varias partes y transferirlas en paralelo.

Use la Guía para desarrolladores de AWS SDK para Go como ayuda para instalar, configurar y usar el SDK. En esta guía se proporcionan información de configuración, código de muestra y una presentación de las utilidades del SDK.

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](#)

Empiece a utilizar AWS SDK para Go

AWS SDK para Go requiere una versión mínima de Go 1.23. Para ver su versión actual de Go, ejecute el siguiente comando:

```
go version
```

Para obtener información sobre cómo instalar o actualizar su versión de Go, consulte [Download and install](#) en la documentación de Go.

Obtención de una cuenta de Amazon

Para poder utilizar AWS SDK para Go v2, antes debe tener una cuenta de Amazon. Consulte [Creación de una cuenta de AWS](#) para obtener más información.

Instalación de AWS SDK para Go v2

AWS SDK para Go v2 usa los módulos de Go, una característica que se incorporó en Go 1.11. Inicialice el proyecto local mediante la ejecución del siguiente comando de Go:

```
go mod init example
```

Tras inicializar el proyecto del módulo de Go, podrá recuperar el SDK y sus dependencias necesarias mediante el comando `go get`. Estas dependencias se registrarán en el archivo `go.mod` que creó el comando anterior.

Los siguientes comandos muestran cómo recuperar el conjunto estándar de módulos del SDK para usarlos en la aplicación.

```
go get github.com/aws/aws-sdk-go-v2
go get github.com/aws/aws-sdk-go-v2/config
```

De este modo se recuperarán tanto el módulo del SDK principal como el módulo de configuración que se utiliza para cargar la configuración compartida de AWS.

A continuación, puede instalar uno o varios clientes de API de servicio de AWS necesarios para la aplicación. Todos los clientes de API se encuentran en la jerarquía de importación `github.com/`

`aws/aws-sdk-go-v2/service`. [Aquí](#) encontrará un conjunto completo de los clientes de API actualmente compatibles. Para instalar un cliente de servicio, ejecute el comando siguiente a fin de recuperar el módulo y registrar la dependencia en el archivo `go.mod`. En este ejemplo recuperamos el cliente de API de Amazon S3.

```
go get github.com/aws/aws-sdk-go-v2/service/s3
```

Obtener las claves de acceso de AWS

Las claves de acceso constan de un ID de clave de acceso y una clave de acceso secreta, que se utilizan para firmar las solicitudes de programación que se realizan a AWS. Si no tiene claves de acceso, puede crearlas mediante la [Consola de administración de AWS](#). Es recomendable que utilice las claves de acceso de IAM en lugar de las claves de acceso de la cuenta raíz de AWS. IAM le permite controlar de forma segura el acceso a los servicios de AWS y a los recursos de su cuenta de AWS.

Note

Para crear claves de acceso, debe tener permisos para realizar las acciones de IAM requeridas. Para obtener más información, consulte [Permisos obligatorios para obtener acceso a recursos de IAM](#) de la Guía del usuario de IAM.

Para obtener su ID de clave de acceso y clave de acceso secreta

1. Abra la [consola de IAM](#).
2. En el menú de navegación, elija Users (Usuarios).
3. Seleccione su nombre de usuario de IAM (no la casilla de verificación).
4. Abra la pestaña Security credentials (Credenciales de seguridad) y, a continuación, seleccione Create access key (Crear clave de acceso).
5. Para ver la nueva clave de acceso, elija Show (Mostrar). Sus credenciales serán similares a las siguientes:
 - ID de clave de acceso: AKIAIOSFODNN7EXAMPLE
 - Clave de acceso secreta: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
6. Para descargar el par de claves, elija Descargar archivo .csv. Almacene las claves en un lugar seguro.

 Warning

Mantenga la confidencialidad de las claves para proteger su cuenta de AWS y no las comparta nunca con nadie ajeno a su organización.

Temas relacionados

- [¿Qué es IAM?](#) de la Guía del usuario de IAM.
- [Credenciales de seguridad de AWS](#) de la Referencia general de Amazon Web Services.

Invocación de una operación

Una vez que haya instalado el SDK, importará paquetes de AWS a sus aplicaciones Go para usar el SDK, como se muestra en el siguiente ejemplo, que importa las bibliotecas de AWS, Config y Amazon S3. Tras importar los paquetes del SDK, se carga la configuración compartida del SDK de AWS, se crea un cliente y se invoca una operación de la API.

```
package main

import (
    "context"
    "log"
    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"
)

func main() {
    // Load the Shared AWS Configuration (~/.aws/config)
    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        log.Fatal(err)
    }

    // Create an Amazon S3 service client
    client := s3.NewFromConfig(cfg)

    // Get the first page of results for ListObjectsV2 for a bucket
    output, err := client.ListObjectsV2(context.TODO(), &s3.ListObjectsV2Input{
```

```
        Bucket: aws.String("amzn-s3-demo-bucket"),
    })
    if err != nil {
        log.Fatal(err)
    }

    log.Println("first page results")
    for _, object := range output.Contents {
        log.Printf("key=%s size=%d", aws.ToString(object.Key), *object.Size)
    }
}
```

Configurar el SDK

En la AWS SDK para Go versión 2, puede configurar los ajustes comunes para los clientes de servicio, como el registrador, el nivel de registro y la configuración de reintentos. La mayoría de los ajustes son opcionales. Sin embargo, para cada cliente de servicio, debe especificar una AWS región y sus credenciales. El SDK usa estos valores para enviar solicitudes a la región correcta y firmar las solicitudes con las credenciales correctas. Puede especificar estos valores mediante programación en el código o a través del entorno de ejecución.

Cargando archivos de configuración AWS compartidos

Existen varias formas de inicializar un cliente de la API de servicio, pero el siguiente patrón es el más común que se recomienda a los usuarios.

Para configurar el SDK para que utilice los archivos de configuración AWS compartidos, utilice el siguiente código:

```
import (
    "context"
    "log"
    "github.com/aws/aws-sdk-go-v2/config"
)

// ...

cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    log.Fatalf("failed to load configuration, %v", err)
}
```

`config.LoadDefaultConfig(context.TODO())` construirá un [AWS.config](#) utilizando las fuentes de configuración AWS compartidas. Esto incluye configurar un proveedor de credenciales, configurar la AWS región y cargar la configuración específica del servicio. Los clientes de servicio se pueden crear mediante la instancia de `aws.Config` cargada, lo que proporciona un patrón coherente para la creación de clientes.

Para obtener más información sobre los archivos de configuración AWS compartidos, consulte [Configuración](#) en la Guía de referencia de herramientas AWS SDKs y herramientas.

Especificar la AWS región

A especificar la región, se especifica dónde enviar las solicitudes, por ejemplo, `us-west-2` o `us-east-2`. Para ver una lista de regiones para cada servicio, consulte [Puntos de conexión de servicio y cuotas](#) en la Referencia general de Amazon Web Services.

El SDK no tiene ninguna región predeterminada. Para especificar una región:

- Establezca la variable de entorno `AWS_REGION` en la región predeterminada.
- Establezca la región de forma explícita mediante la [configuración. `WithRegion`](#) como argumento `config.LoadDefaultConfig` al cargar la configuración.

Revisión: Si establece una región mediante todas estas técnicas, el SDK utilizará la región que haya especificado de forma explícita.

Configuración de la región con una variable de entorno

Linux, macOS o Unix

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

Windows

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

Especificación de la región mediante programación

```
cfg, err := config.LoadDefaultConfig(context.TODO(), config.WithRegion("us-west-2"))
```

Especificación de credenciales

AWS SDK para Go Requiere credenciales (una clave de acceso y una clave de acceso secreta) para firmar las solicitudes AWS. Puede especificar sus credenciales en varias ubicaciones, en función de su caso de uso concreto. Para obtener más información sobre cómo obtener las credenciales, consulte [Empiece a utilizar AWS SDK para Go](#).

Al inicializar una `aws.Config` instancia mediante `config.LoadDefaultConfig`, el SDK utiliza su cadena de credenciales predeterminada para buscar AWS las credenciales. Esta cadena de credenciales predeterminada busca las credenciales en este orden:

1. Variables de entorno.
 1. Credenciales estáticas (`AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY` y `AWS_SESSION_TOKEN`)
 2. Token de identidad web (`AWS_WEB_IDENTITY_TOKEN_FILE`)
2. Archivos de configuración compartida
 1. El SDK tiene como valor predeterminado el archivo `credentials` de la carpeta `.aws` que se encuentra en la carpeta de inicio de su computadora.
 2. El SDK tiene como valor predeterminado el archivo `config` de la carpeta `.aws` que se encuentra en la carpeta de inicio de su computadora.
3. Si su aplicación utiliza una definición de tareas o una operación de `RunTask API` de Amazon ECS, la función de IAM para las tareas.
4. Si tu aplicación se ejecuta en una EC2 instancia de Amazon, rol de IAM para Amazon EC2.

El SDK detecta y utiliza los proveedores integrados de forma automática, sin necesidad de realizar configuraciones manuales. Por ejemplo, si utilizas funciones de IAM para las EC2 instancias de Amazon, tus aplicaciones utilizarán automáticamente las credenciales de la instancia. No es necesario configurar manualmente las credenciales en la aplicación.

Como práctica recomendada, se AWS recomienda especificar las credenciales en el siguiente orden:

1. Utilice funciones de IAM para las tareas si su aplicación utiliza una definición de tarea o una operación de `RunTask API` de Amazon ECS.
2. Usa funciones de IAM para Amazon EC2 (si tu aplicación se ejecuta en una EC2 instancia de Amazon).

Los roles de IAM proporcionan a las aplicaciones de la instancia credenciales de seguridad temporales para realizar AWS llamadas. Las funciones de IAM proporcionan una forma sencilla de distribuir y gestionar las credenciales en varias EC2 instancias de Amazon.

3. Utilice archivos de configuración o credenciales compartidos.

Las credenciales y los archivos de configuración se comparten entre otras AWS SDKs y AWS CLI. Como práctica recomendada de seguridad, recomendamos utilizar un archivo de credenciales

para configurar valores confidenciales, como la clave de acceso IDs y las claves secretas. Estos son los [requisitos de formato](#) para cada uno de estos archivos.

4. Utilice variables de entorno.

Establecer variables de entorno es útil si realizas trabajos de desarrollo en una máquina que no sea una EC2 instancia de Amazon.

Roles de IAM para las tareas

Si su aplicación utiliza una definición de tarea de Amazon ECS o una operación RunTask, utilice [roles de IAM para las tareas](#) a fin de especificar un rol de IAM que los contenedores puedan utilizar en una tarea.

Funciones de IAM para Amazon Instances EC2

Si ejecutas tu aplicación en una EC2 instancia de Amazon, usa la [función de IAM](#) de la instancia para obtener credenciales de seguridad temporales a las que realizar llamadas. AWS

Si ha configurado la instancia para que utilice roles de IAM, el SDK utiliza estas credenciales para su aplicación de forma automática. No es necesario especificar las credenciales manualmente.

Credenciales y configuración compartidas

Las credenciales compartidas y los archivos de configuración se pueden usar para compartir configuraciones comunes entre AWS SDKs otras herramientas. Si utiliza credenciales distintas para cada herramienta o aplicación, puede utilizar perfiles para configurar varias claves de acceso en el mismo archivo de configuración.

Puede proporcionar varias ubicaciones de archivos de credenciales y de configuración mediante `config.LoadOptions`. De forma predeterminada, el SDK carga los archivos almacenados en las ubicaciones predeterminadas que se indican en [Especificación de credenciales](#).

```
import (  
    "context"  
    "github.com/aws/aws-sdk-go-v2/config"  
)  
  
// ...
```

```
cfg , err := config.LoadDefaultConfig(context.TODO(),
    config.WithSharedCredentialsFiles(
        []string{"test/credentials", "data/credentials"},
    ),
    config.WithSharedConfigFiles(
        []string{"test/config", "data/config"},
    )
)
```

Al trabajar con archivos de credenciales y configuración compartidas, en caso de especificarse perfiles duplicados estos se fusionan para resolver un perfil. En caso de conflicto al fusionar:

1. Si se especifican perfiles duplicados en un mismo credentials/config archivo, prevalecen las propiedades del perfil especificadas en este último perfil.
2. Si se especifican perfiles duplicados en varios archivos de credenciales o de configuración, las propiedades de perfil se resolverán según el orden de entrada del archivo en `config.LoadOptions`. Las propiedades de perfil de los últimos archivos tienen prioridad.
3. Si existe un perfil tanto en el archivo de credenciales como en el de configuración, las propiedades del archivo de credenciales tendrán prioridad.

Si es necesario, puede habilitar `LogConfigurationWarnings` en `config.LoadOptions` y registrar los pasos de resolución de perfiles.

Creación del archivo de credenciales

Si no tiene un archivo de credenciales compartido (`.aws/credentials`), puede usar cualquier editor de texto para crear uno en su directorio principal. Añada el siguiente contenido a su archivo de credenciales, sustituyendo `<YOUR_ACCESS_KEY_ID>` y `<YOUR_SECRET_ACCESS_KEY>` por sus credenciales.

```
[default]
aws_access_key_id = <YOUR_ACCESS_KEY_ID>
aws_secret_access_key = <YOUR_SECRET_ACCESS_KEY>
```

El encabezado `[default]` define las credenciales para el perfil predeterminado, que el SDK utilizará a menos que lo configure para usar otro perfil.

También puede utilizar credenciales de seguridad temporales mediante la adición de los tokens de sesión a su perfil, como se muestra en el ejemplo siguiente:

```
[temp]
aws_access_key_id = <YOUR_TEMP_ACCESS_KEY_ID>
aws_secret_access_key = <YOUR_TEMP_SECRET_ACCESS_KEY>
aws_session_token = <YOUR_SESSION_TOKEN>
```

El nombre de la sección correspondiente a un perfil no predeterminado en un archivo de credenciales no debe empezar por la palabra `profile`. Puede obtener más información en la [Guía AWS SDKs de referencia de herramientas](#).

Creación del archivo de configuración

Si no tiene un archivo de credenciales compartido (`.aws/config`), puede usar cualquier editor de texto para crear uno en su directorio principal. Añada el siguiente contenido a su archivo de configuración y `<REGION>` sustitúyalo por la región deseada.

```
[default]
region = <REGION>
```

El encabezado `[default]` define la configuración para el perfil predeterminado, que el SDK utilizará a menos que lo configure para usar otro perfil.

Puede utilizar perfiles con nombre, como se muestra en el ejemplo siguiente:

```
[profile named-profile]
region = <REGION>
```

El nombre de sección de un perfil no predeterminado de un archivo de configuración siempre debe empezar por la palabra `profile`, seguida del nombre de perfil previsto. Puede obtener más información en la [Guía de referencia de herramientas AWS SDKs y herramientas](#).

Especificación de perfiles

Puede incluir varias claves de acceso en el mismo archivo de configuración; para ello, asocie cada conjunto de claves de acceso a un perfil. Por ejemplo, en el archivo de credenciales, puede declarar varios perfiles, de la siguiente manera.

```
[default]
aws_access_key_id = <YOUR_DEFAULT_ACCESS_KEY_ID>
aws_secret_access_key = <YOUR_DEFAULT_SECRET_ACCESS_KEY>
```

```
[test-account]
aws_access_key_id = <YOUR_TEST_ACCESS_KEY_ID>
aws_secret_access_key = <YOUR_TEST_SECRET_ACCESS_KEY>

[prod-account]
; work profile
aws_access_key_id = <YOUR_PROD_ACCESS_KEY_ID>
aws_secret_access_key = <YOUR_PROD_SECRET_ACCESS_KEY>
```

De forma predeterminada, el SDK comprueba la variable de entorno `AWS_PROFILE` para determinar qué perfil se va a utilizar. Si no se ha establecido ninguna variable `AWS_PROFILE`, el SDK usa el perfil `default`.

Puede que a veces quiera utilizar un perfil diferente con su aplicación. Por ejemplo, desea usar las credenciales `test-account` con su aplicación `myapp`. Puede usar este perfil mediante el siguiente comando:

```
$ AWS_PROFILE=test-account myapp
```

También puede indicar al SDK que seleccione un perfil mediante una llamada a `os.Setenv("AWS_PROFILE", "test-account")` antes de llamar a `config.LoadDefaultConfig`, o mediante la transferencia de un perfil explícito como argumento, como se muestra en el ejemplo siguiente:

```
cfg, err := config.LoadDefaultConfig(context.TODO(),
    config.WithSharedConfigProfile("test-account"))
```

Note

Si especifica las credenciales en variables de entorno, el SDK siempre usará esas credenciales, independientemente del perfil que especifique.

Variables de entorno

De forma predeterminada, el SDK detecta AWS las credenciales configuradas en su entorno y las utiliza para firmar solicitudes AWS. De ese modo, no es necesario administrar las credenciales en las aplicaciones.

El SDK busca credenciales en las siguientes variables de entorno:

- `AWS_ACCESS_KEY_ID`
- `AWS_SECRET_ACCESS_KEY`
- `AWS_SESSION_TOKEN` (opcional)

En los ejemplos siguientes se muestra cómo configurar las variables de entorno:

Linux, OS X o Unix

```
$ export AWS_ACCESS_KEY_ID=YOUR_AKID
$ export AWS_SECRET_ACCESS_KEY=YOUR_SECRET_KEY
$ export AWS_SESSION_TOKEN=TOKEN
```

Windows

```
> set AWS_ACCESS_KEY_ID=YOUR_AKID
> set AWS_SECRET_ACCESS_KEY=YOUR_SECRET_KEY
> set AWS_SESSION_TOKEN=TOKEN
```

Especificación de credenciales mediante programación

`config.LoadDefaultConfig` permite proporcionar una [AWS explícita. `CredentialProvider`](#) al cargar las fuentes de configuración compartidas. Para pasar un proveedor de credenciales explícito al cargar la configuración compartida, utilice [`config.WithCredentialsProvider`](#).

Por ejemplo, si `customProvider` hace referencia a una instancia de implementación de `aws.CredentialProvider`, se puede pasar durante la carga de la configuración de la siguiente manera:

```
cfg, err := config.LoadDefaultConfig(context.TODO(),
    config.WithCredentialsProvider(customProvider))
```

Si proporciona credenciales de forma explícita, como en este ejemplo, el SDK solo usa esas credenciales.

Note

Todos los proveedores de credenciales transferidos o devueltos `LoadDefaultConfig` se incluyen [`CredentialsCache`](#) automáticamente. Esto permite un almacenamiento en caché y una rotación de credenciales seguros para el uso simultáneo. Si configuras explícitamente

un proveedor `aws.Config` directamente, también debes empaquetar explícitamente el proveedor con este tipo mediante [NewCredentialsCache](#).

Credenciales estáticas

Puede codificar de forma rígida las credenciales en su aplicación mediante las [credenciales](#). [NewStaticCredentialsProvider](#) proveedor de credenciales para establecer de forma explícita las claves de acceso que se van a utilizar. Por ejemplo:

```
cfg, err := config.LoadDefaultConfig(context.TODO(),
    config.WithCredentialsProvider(credentials.NewStaticCredentialsProvider("AKID",
    "SECRET_KEY", "TOKEN")),
)
```

Warning

No incruste credenciales dentro de una aplicación. Use este método solo con fines de prueba.

Credenciales de inicio de sesión único

El SDK proporciona un proveedor de credenciales para recuperar credenciales temporales AWS mediante. AWS IAM Identity Center Con el AWS CLI, se autentica en el portal de acceso y se autoriza el AWS acceso a las credenciales temporales. AWS A continuación, configuras la aplicación para que cargue el perfil de inicio de sesión único (SSO) y el SDK utiliza tus credenciales de SSO para recuperar las AWS credenciales temporales que se renovarán automáticamente si caducan. Si sus credenciales de SSO caducan, debe renovarlas de forma explícita iniciando sesión de nuevo en su cuenta del IAM Identity Center mediante. AWS CLI

Por ejemplo, puede crear un perfil `dev-profile`, autenticar y autorizar ese perfil mediante la AWS CLI aplicación y configurar su aplicación como se muestra a continuación.

1. Primero, cree los elementos `profile` y `sso-session`.

```
[profile dev-profile]
sso_session = dev-session
```

```
sso_account_id = 012345678901
sso_role_name = Developer
region = us-east-1

[sso-session dev-session]
sso_region = us-west-2
sso_start_url = https://company-sso-portal.awsapps.com/start
sso_registration_scopes = sso:account:access
```

2. Inicie sesión con el AWS CLI para autenticar y autorizar el perfil de inicio de sesión único.

```
$ aws --profile dev-profile sso login
Attempting to automatically open the SSO authorization page in your default browser.
If the browser does not open or you wish to use a different device to authorize this
request, open the following URL:

https://device.sso.us-west-2.amazonaws.com/

Then enter the code:

ABCD-EFGH
Successfully logged into Start URL: https://company-sso-portal.awsapps.com/start
```

3. A continuación, configure la aplicación para que utilice el perfil de SSO.

```
import "github.com/aws/aws-sdk-go-v2/config"

// ...

cfg, err := config.LoadDefaultConfig(
    context.Background(),
    config.WithSharedConfigProfile("dev-profile"),
)
if err != nil {
    return err
}
```

Para obtener más información sobre la configuración de los perfiles de SSO y la autenticación mediante el uso, AWS CLI consulte [Configuración del uso AWS IAM Identity Center en la AWS CLI Guía del](#) usuario. AWS CLI Para obtener más información sobre cómo crear el proveedor de

credenciales de SSO mediante programación, consulte la documentación de referencia de la API [ssocreds](#).

Credenciales de acceso

Puede usar sus credenciales de inicio de sesión AWS de Management Console actuales 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.

1. Inicie el flujo de inicio de sesión mediante la AWS CLI y siga las instrucciones del navegador. En este ejemplo, guardamos la sesión de inicio de sesión en un perfil `nuevodev-profile`, pero esto es opcional.

```
$ aws --profile dev-profile login
```

2. (Opcional) Inspeccione el archivo de configuración AWS compartido para comprobar que la sesión se ha establecido.

```
[profile dev-profile]
login_session = arn:aws:sts::account_id>:role
```

3. A continuación, configure la aplicación para que utilice el perfil de inicio de sesión.

```
import "github.com/aws/aws-sdk-go-v2/config"

// ...

cfg, err := config.LoadDefaultConfig(
    context.Background(),
    // only necessary if login session is saved to a non-default profile
    config.WithSharedConfigProfile("dev-profile"),
)
if err != nil {
    return err
}
```


Para obtener más información sobre la configuración de los perfiles de inicio de sesión y la autenticación mediante la AWS CLI, consulte [Inicio de sesión para el desarrollo AWS local con credenciales de consola](#).

Otros proveedores de credenciales

El SDK proporciona otros métodos para recuperar credenciales en el módulo [credentials](#). Por ejemplo, puede recuperar credenciales de seguridad temporales AWS Security Token Service o credenciales de un almacenamiento cifrado.

Proveedores de credenciales disponibles:

- [ec2rolecreds: recupere las](#) credenciales de los EC2 roles de Amazon Instances mediante Amazon IMDS. EC2
- [endpointcreds](#): recupere credenciales de un punto de conexión HTTP arbitrario.
- [processcreds](#): recupere credenciales de un proceso externo que invocará el intérprete de comandos del entorno de host.
- [stscreds](#) — Recupera credenciales de AWS STS

Configuración de la autenticación

AWS SDK para Go ofrece la posibilidad de configurar el servicio de comportamiento de autenticación. En la mayoría de los casos, la configuración predeterminada será suficiente, pero la configuración de autenticación personalizada permite un comportamiento adicional, como trabajar con características del servicio previas al lanzamiento.

Definiciones

En esta sección se proporciona una descripción detallada de los componentes de autenticación de AWS SDK para Go.

AuthScheme

[AuthScheme](#) es la interfaz que define el flujo de trabajo a través del cual el SDK recupera la identidad del intermediario y la asocia a una solicitud de operación.

Un esquema de autenticación utiliza los componentes siguientes, que se describen en detalle más adelante:

- Un ID único que identifica el esquema
- Un solucionador de identidades, que devuelve la identidad del intermediario utilizada en el proceso de firma (por ejemplo, sus credenciales de AWS)
- Un firmante, que es el que realmente inyecta la identidad del intermediario en la solicitud de transporte de la operación (por ejemplo, el encabezado HTTP `Authorization`)

Cada opción de cliente de servicio incluye un campo `AuthSchemes` que, de forma predeterminada, se rellena con la lista de esquemas de autenticación compatibles con ese servicio.

AuthSchemeResolver

Cada opción de cliente de servicio incluye un campo `AuthSchemeResolver`. Esta interfaz, definida por servicio, es la API a la que llama el SDK para determinar las opciones de autenticación posibles para cada operación.

Important

El solucionador de esquemas de autenticación NO dicta qué esquema de autenticación se usa. Devuelve una lista de esquemas que se pueden usar (“opciones”); el esquema final se selecciona a través de un algoritmo fijo que se describe [aquí](#).

Opción

Un valor [Option](#), devuelto de una llamada a `ResolverAuthSchemes`, representa una posible opción de autenticación.

Una opción consta de tres conjuntos de información:

- Un identificador que representa el posible esquema
- Un conjunto opaco de propiedades que se proporcionará al solucionador de identidades del esquema
- Un conjunto opaco de propiedades que se proporcionará al firmante del esquema

Nota sobre las propiedades

En el 99 % de los casos de uso, los intermediarios no tienen por qué preocuparse por las propiedades opacas para la resolución de identidades y la firma. El SDK extraerá las propiedades

necesarias para cada esquema y las pasará a las interfaces con establecimiento inflexible de tipos expuestas en el SDK. Por ejemplo, el solucionador de autenticación predeterminado para servicios codifica la opción SigV4 de modo que tenga propiedades del firmante para la región y el nombre de la firma, cuyos valores se pasan a la implementación de [v4.HTTPSigner](#) configurada del cliente al seleccionarse SigV4.

Identidad

[Identity](#) es una representación abstracta de quién es el intermediario del SDK.

El tipo de identidad más común que se utiliza en el SDK es un conjunto de `aws.Credentials`. En la mayoría de los casos de uso, el intermediario no tiene por qué preocuparse de `Identity` como abstracción y puede trabajar directamente con los tipos concretos.

Note

Para conservar la compatibilidad con versiones anteriores y evitar confusiones en relación con la API, el tipo de identidad `aws.Credentials` específico del SDK de AWS no cumple directamente con la interfaz `Identity`. Esta asignación se gestiona de forma interna.

IdentityResolver

[IdentityResolver](#) es la interfaz a través de la cual se recupera un valor `Identity`.

Existen versiones concretas de `IdentityResolver` en el SDK con establecimiento inflexible de tipos (por ejemplo, [aws.CredentialsProvider](#)). El SDK gestiona esta asignación de forma interna.

Un intermediario solo tendrá que implementar directamente la interfaz `IdentityResolver` al definir un esquema de autenticación externo.

Signer

[Signer](#) es la interfaz a través de la cual se complementa una solicitud con el valor `Identity` del intermediario recuperado.

Existen versiones concretas de `Signer` en el SDK con establecimiento inflexible de tipos (por ejemplo, [v4.HTTPSigner](#)). El SDK gestiona esta asignación de forma interna.

Un intermediario solo tendrá que implementar directamente la interfaz `Signer` al definir un esquema de autenticación externo.

AuthResolverParameters

Cada servicio toma un conjunto específico de entradas que se pasan a su función de resolución, definida en cada paquete de servicios como `AuthResolverParameters`.

Los parámetros base del solucionador son los siguientes:

nombre	tipo	description
Operation	string	El nombre de la operación que se invoca.
Region	string	La región de AWS del cliente. Solo está presente para los servicios que utilizan SigV4[A].

Si implementa su propio solucionador, no debería tener que crear una instancia propia de sus parámetros en ningún caso. El SDK obtendrá estos valores por solicitud y los pasará a su implementación.

Flujo de trabajo de resolución de esquemas de autenticación

Cuando llama a una operación de servicio de AWS a través del SDK, la siguiente secuencia de acciones se produce una vez serializada la solicitud:

1. El SDK llama a la API `AuthSchemeResolver.ResolveAuthSchemes()` del cliente y obtiene los parámetros de entrada según sea necesario, a fin de conseguir una lista de posibles valores [Option](#) para la operación.
2. El SDK recorre en iteración esa lista y selecciona el primer esquema que cumple con las siguientes condiciones:
 - En la propia lista `AuthSchemes` del cliente hay un esquema con un ID coincidente.
 - El solucionador de identidades del esquema existe (no es `nil`) en las opciones del cliente (si se comprueba mediante el método `GetIdentityResolver` del esquema, la asignación a los tipos de solucionador de identidades concretos descritos anteriormente se gestiona de forma interna). (1)
3. Suponiendo que se haya seleccionado un esquema viable, el SDK invoca su API `GetIdentityResolver()` para recuperar la identidad del intermediario. Por ejemplo,

el esquema de autenticación SigV4 integrado se asignará internamente al proveedor de Credentials del cliente de forma interna.

4. El SDK llama a `GetIdentity()` del solucionador de identidades (por ejemplo, `aws.CredentialProvider.Retrieve()` para SigV4).
5. El SDK llama a `ResolveEndpoint()` del solucionador de puntos de conexión para encontrar el punto de conexión de la solicitud. El punto de conexión puede incluir metadatos adicionales que influyan en el proceso de firma (por ejemplo, un nombre de firma único para S3 Object Lambda).
6. El SDK llama a la API `Signer()` del esquema de autenticación para recuperar su firmante y usa su API `SignRequest()` para firmar la solicitud con la identidad del intermediario recuperada anteriormente.

(1) Si el SDK encuentra la opción anónima (ID `smithy.api#noAuth`) en la lista, se selecciona de forma automática, ya que no existe un solucionador de identidades correspondiente.

AuthScheme compatibles de forma nativa

Los esquemas de autenticación siguientes son compatibles de forma nativa con AWS SDK para Go.

Nombre	ID de esquema	Solucionador de identidades	Signer	Notas
SigV4	<code>aws.auth#sigv4</code>	aws.CredentialsProvider	v4.HTTPSigner	El valor predeterminado actual de la mayoría de las operaciones de servicio de AWS.
SigV4A	<code>aws.auth#sigv4a</code>	<code>aws.CredentialsProvider</code>	n/a	El uso de SigV4A es limitado en este momento, la implementación del firmante es interna. Consulte este anuncio para ver un

Nombre	ID de esquema	Solucionador de identidades	Signer	Notas
				nuevo módulo de inclusión aws-http-auth que expone las API de uso general para firmar solicitudes HTTP.
SigV4Express	com.amazonaws.s3#sigv4express	s3.ExpressCredentialsProvider	v4.HTTPSigner	Se utiliza para Express One Zone .
HTTP Bearer	smithy.api#httpBearerAuth	smithybearer.TokenProvider	smithybearer.Signer	Utilizado por codecatalyst .
Anonymous	smithy.api#noAuth	n/a	n/a	Sin autenticación: no se requiere identidad y la solicitud no se firma ni se autentica.

Configuración de identidades

En AWS SDK para Go, los componentes de identidad de un esquema de autenticación se configuran en `Options` del cliente del SDK. El SDK recogerá y utilizará de forma automática los valores de estos componentes para el esquema que seleccione cuando se llame a una operación.

 Note

Por motivos de compatibilidad con versiones anteriores, el SDK permite de forma implícita el uso del esquema de autenticación anónimo si no se ha configurado ningún solucionador de identidades. Esto se puede lograr de forma manual mediante el establecimiento de todos los solucionadores de identidades del elemento `Options` de un cliente en `nil` (el solucionador de identidades `sigv4` también se puede establecer en `aws.AnonymousCredentials{}`).

Configuración del firmante

En AWS SDK para Go, los componentes del firmante correspondientes a un esquema de autenticación se configuran en `Options` en el cliente del SDK. El SDK recogerá y utilizará de forma automática los valores de estos componentes para el esquema que seleccione cuando se llame a una operación. No es necesaria ninguna configuración adicional.

Esquema de autenticación personalizado

Para definir un esquema de autenticación personalizado y configurarlo para su uso, el intermediario debe hacer lo siguiente:

1. Definir una implementación [AuthScheme](#)
2. Registrar el esquema en la lista `AuthSchemes` del cliente del SDK
3. Instrumentar el `AuthSchemeResolver` del cliente del SDK para que devuelva un valor `Option` de autenticación con el ID del esquema, cuando proceda.

 Warning

Los servicios siguientes tienen un comportamiento de autenticación único o personalizado. Recomendamos que delegue en la implementación predeterminada y la encapsule en consecuencia si necesita un comportamiento de autenticación personalizado:

Servicio	Notas
S3	Uso condicional de <code>SigV4A</code> y <code>SigV4Express</code> en función de la entrada de la operación.

Servicio	Notas
EventBridge	Uso condicional de SigV4A en función de la entrada de la operación.
Cognito	Algunas operaciones son únicamente anónimas.
SSO	Algunas operaciones son únicamente anónimas.
STS	Algunas operaciones son únicamente anónimas.

Configuración de puntos de conexión del cliente

Warning

La resolución de puntos de conexión es un tema avanzado del SDK. Al cambiar esta configuración, corre el riesgo de invalidar el código. La configuración predeterminada debe poder aplicarse a la mayoría de los usuarios en entornos de producción.

AWS SDK para Go Ofrece la posibilidad de configurar un punto final personalizado para usarlo en un servicio. En muchos casos, la configuración predeterminada será suficiente. La configuración de puntos de conexión personalizados permite un comportamiento adicional, como trabajar con versiones previas al lanzamiento de un servicio.

Personalización

Existen dos “versiones” de la configuración de resolución de puntos de conexión en el SDK:

- v2, publicada en el tercer trimestre de 2023, configurada mediante:
 - `EndpointResolverV2`
 - `BaseEndpoint`
- v1, publicada junto con el SDK, configurada mediante:
 - `EndpointResolver`

Recomendamos a los usuarios de la resolución de puntos de conexión v1 que migren a la versión 2 para obtener acceso a características del servicio más recientes relacionadas con estos.

V2: **EndpointResolverV2 + BaseEndpoint**

En la versión 2 de la resolución, `EndpointResolverV2` es el mecanismo definitivo a través del cual se produce la resolución de puntos de conexión. El método `ResolveEndpoint` del solucionador se invoca como parte del flujo de trabajo para cada solicitud que realice en el SDK. El nombre de host del `Endpoint` devuelto por el solucionador se usa tal cual al realizar la solicitud (sin embargo, los serializadores de operaciones pueden seguir asociando elementos a la ruta HTTP).

En la resolución v2 se incluye una configuración adicional de nivel de cliente, `BaseEndpoint`, que se utiliza a fin de especificar un nombre de host “base” para la instancia de su servicio. El valor establecido aquí no es definitivo: se transfiere en última instancia como parámetro al `EndpointResolverV2` del cliente cuando se produce la resolución final (siga leyendo para obtener más información sobre los parámetros `EndpointResolverV2`). La implementación del solucionador tiene entonces la oportunidad de inspeccionar y, posiblemente, modificar ese valor para determinar el punto de conexión final.

Por ejemplo, si realiza una solicitud `GetObject` de S3 en un bucket determinado con un cliente en el que ha especificado un `BaseEndpoint`, el solucionador predeterminado inyectará el bucket en el nombre del host si es compatible con el host virtual (suponiendo que no haya desactivado el alojamiento virtual en la configuración del cliente).

En la práctica, lo más probable es que `BaseEndpoint` se utilice para dirigir al cliente a una instancia de desarrollo o de vista previa de un servicio.

Parámetros **EndpointResolverV2**

Cada servicio toma un conjunto específico de entradas que se transfieren a su función de resolución, definida en cada paquete de servicios como `EndpointParameters`.

Cada servicio incluye los siguientes parámetros base, que se utilizan para facilitar la resolución de puntos de conexión general en AWS:

name	type	description
Region	string	La AWS región del cliente

name	type	description
Endpoint	string	Indica el valor establecido para <code>BaseEndpoint</code> en la configuración del cliente.
UseFips	bool	Indica si los puntos de conexión de FIPS están habilitados en la configuración del cliente.
UseDualStack	bool	Indica si los puntos de conexión de doble pila están habilitados en la configuración del cliente.

Los servicios pueden especificar parámetros adicionales necesarios para la resolución. Por ejemplo, los `EndpointParameters` de S3 incluyen el nombre del bucket, así como varios ajustes de características específicos de S3 (por ejemplo, si el direccionamiento de host virtual está habilitado).

Si implementa su propio `EndpointResolverV2`, no debería tener que crear una instancia propia de `EndpointParameters` en ningún caso. El SDK obtendrá los valores por solicitud y los transferirá a su implementación.

Nota acerca de Amazon S3

Amazon S3 es un servicio complejo con muchas de sus características modeladas a través de complejas personalizaciones de puntos de conexión, como el alojamiento virtual de buckets o el punto de acceso multirregional de S3, entre otras.

Por ello, recomendamos que no reemplace la implementación de `EndpointResolverV2` en su cliente de S3. Si necesita ampliar su comportamiento de resolución (por ejemplo, mediante el envío de solicitudes a una pila de desarrollo local con consideraciones adicionales sobre los puntos de conexión), recomendamos encapsular la implementación predeterminada de forma que se delegue de nuevo en esta como alternativa (como se muestra en los ejemplos que aparecen a continuación).

Ejemplos

Con **BaseEndpoint**

El fragmento de código siguiente muestra cómo apuntar su cliente de S3 a la instancia local de un servicio, que en este ejemplo se aloja en el dispositivo de bucle invertido en el puerto 8080.

```
client := s3.NewFromConfig(cfg, func (o *svc.Options) {
    o.BaseEndpoint = aws.String("https://localhost:8080/")
})
```

Con **EndpointResolverV2**

El fragmento de código siguiente muestra cómo inyectar un comportamiento personalizado en la resolución de puntos de conexión de S3 mediante `EndpointResolverV2`.

```
import (
    "context"
    "net/url"

    "github.com/aws/aws-sdk-go-v2/service/s3"
    smithyendpoints "github.com/aws/smithy-go/endpoints"
)

type resolverV2 struct {
    // you could inject additional application context here as well
}

func (*resolverV2) ResolveEndpoint(ctx context.Context, params s3.EndpointParameters) (
    smithyendpoints.Endpoint, error,
) {
    if /* input params or caller context indicate we must route somewhere */ {
        u, err := url.Parse("https://custom.service.endpoint/")
        if err != nil {
            return smithyendpoints.Endpoint{}, err
        }
        return smithyendpoints.Endpoint{
            URI: *u,
        }, nil
    }

    // delegate back to the default v2 resolver otherwise
}
```

```

    return s3.NewDefaultEndpointResolverV2().ResolveEndpoint(ctx, params)
}

func main() {
    // load config...

    client := s3.NewFromConfig(cfg, func (o *s3.Options) {
        o.EndpointResolverV2 = &resolverV2{
            // ...
        }
    })
}

```

Con ambos

El programa de ejemplo siguiente demuestra la interacción entre `BaseEndpoint` y `EndpointResolverV2`. Este es un caso de uso avanzado:

```

import (
    "context"
    "fmt"
    "log"
    "net/url"

    "github.com/aws/aws-sdk-go-v2"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    smithyendpoints "github.com/aws/smithy-go/endpoints"
)

type resolverV2 struct {}

func (*resolverV2) ResolveEndpoint(ctx context.Context, params s3.EndpointParameters) (
    smithyendpoints.Endpoint, error,
) {
    // s3.Options.BaseEndpoint is accessible here:
    fmt.Printf("The endpoint provided in config is %s\n", *params.Endpoint)

    // fallback to default
    return s3.NewDefaultEndpointResolverV2().ResolveEndpoint(ctx, params)
}

func main() {

```

```
cfg, err := config.LoadDefaultConfig(context.Background())
if (err != nil) {
    log.Fatal(err)
}

client := s3.NewFromConfig(cfg, func (o *s3.Options) {
    o.BaseEndpoint = aws.String("https://endpoint.dev/")
    o.EndpointResolverV2 = &resolverV2{}
})

// ignore the output, this is just for demonstration
client.ListBuckets(context.Background(), nil)
}
```

Cuando se ejecuta, el programa anterior genera lo siguiente:

```
The endpoint provided in config is https://endpoint.dev/
```

V1: EndpointResolver

Warning

La resolución de puntos de conexión v1 se conserva por motivos de compatibilidad con versiones anteriores y está aislada del comportamiento moderno de la resolución de puntos de conexión v2. Solo se usará si el intermediario establece el campo `EndpointResolver`. Lo más probable es que el uso de la versión 1 le impida acceder a las características del servicio relacionadas con los puntos de conexión que se introdujeron con el lanzamiento de la resolución v2 o después de este. Consulte “Migración” para obtener instrucciones sobre cómo actualizar.

A se [EndpointResolver](#) puede configurar para proporcionar una lógica de resolución de puntos finales personalizada para los clientes del servicio. Puede utilizar un solucionador de puntos de conexión personalizado para reemplazar la lógica de resolución de puntos de conexión de un servicio para todos los puntos de conexión o solo para un punto de conexión regional específico. La resolución de puntos finales personalizada puede activar la lógica de resolución de puntos finales del servicio como alternativa si una resolución personalizada no desea resolver un punto final solicitado. [EndpointResolverWithOptionsFunc](#) se puede utilizar para agrupar fácilmente las funciones a fin de adaptarlas a la `EndpointResolverWithOptions` interfaz.

A se `EndpointResolver` puede configurar fácilmente pasando el solucionador incluido en el [WithEndpointResolverWithOptions](#) paquete [LoadDefaultConfig](#), lo que permite anular los puntos finales al cargar las credenciales, así como configurar el resultado `aws.Config` con un solucionador de puntos final personalizado.

El solucionador de puntos de conexión recibe el servicio y la región en forma de cadena, lo que le permite controlar su comportamiento de forma dinámica. Cada paquete de clientes de servicio tiene una constante `ServiceID` exportada que se puede utilizar para determinar qué cliente de servicio invoca el solucionador de puntos de conexión.

Un solucionador de terminales puede utilizar el valor del error [EndpointNotFoundError](#) centinela para activar la resolución alternativa según la lógica de resolución predeterminada del cliente del servicio. Esto permite reemplazar de forma selectiva uno o varios puntos de conexión sin problemas y sin tener que gestionar una lógica alternativa.

Si la implementación del solucionador de puntos de conexión devuelve un error distinto de `EndpointNotFoundError`, la resolución de puntos de conexión se detiene y la operación del servicio devuelve un error a la aplicación.

Ejemplos

Con alternativa

El fragmento de código siguiente muestra cómo se puede reemplazar un único punto de conexión de servicio para DynamoDB con un comportamiento alternativo para otros puntos de conexión:

```
customResolver := aws.EndpointResolverWithOptionsFunc(func(service, region string,
    options ...interface{}) (aws.Endpoint, error) {
    if service == dynamodb.ServiceID && region == "us-west-2" {
        return aws.Endpoint{
            PartitionID: "aws",
            URL:        "https://test.us-west-2.amazonaws.com",
            SigningRegion: "us-west-2",
        }, nil
    }
    // returning EndpointNotFoundError will allow the service to fallback to it's
    default resolution
    return aws.Endpoint{}, &aws.EndpointNotFoundError{}
})
```

```
cfg, err := config.LoadDefaultConfig(context.TODO(),
    config.WithEndpointResolverWithOptions(customResolver))
```

Sin alternativa

El fragmento de código siguiente muestra cómo se puede reemplazar un único punto de conexión de servicio para DynamoDB sin un comportamiento alternativo para otros puntos de conexión:

```
customResolver := aws.EndpointResolverWithOptionsFunc(func(service, region string,
    options ...interface{}) (aws.Endpoint, error) {
    if service == dynamodb.ServiceID && region == "us-west-2" {
        return aws.Endpoint{
            PartitionID: "aws",
            URL:         "https://test.us-west-2.amazonaws.com",
            SigningRegion: "us-west-2",
        }, nil
    }
    return aws.Endpoint{}, fmt.Errorf("unknown endpoint requested")
})

cfg, err := config.LoadDefaultConfig(context.TODO(),
    config.WithEndpointResolverWithOptions(customResolver))
```

Puntos de conexión inmutables

Warning

Establecer un punto de conexión como inmutable puede impedir que algunas de las características del cliente de servicio funcionen correctamente, lo que podría provocar un comportamiento indefinido. Se debe tener cuidado a la hora de definir un punto de conexión como inmutable.

Algunos clientes de servicio, como Amazon S3, pueden modificar el punto de conexión devuelto por el solucionador para determinadas operaciones de servicio. Por ejemplo, Amazon S3 gestionará de forma automática el [direccionamiento virtual de buckets](#) al mutar el punto de conexión resuelto. Puedes evitar que el SDK modifique tus puntos de conexión personalizados configurándolos en [HostnameImmutable](#) `true` Por ejemplo:

```
customResolver := aws.EndpointResolverWithOptionsFunc(func(service, region string,
    options ...interface{}) (aws.Endpoint, error) {
```

```
if service == dynamodb.ServiceID && region == "us-west-2" {
    return aws.Endpoint{
        PartitionID: "aws",
        URL:         "https://test.us-west-2.amazonaws.com",
        SigningRegion: "us-west-2",
        HostnameImmutable: true,
    }, nil
}
return aws.Endpoint{}, fmt.Errorf("unknown endpoint requested")
})

cfg, err := config.LoadDefaultConfig(context.TODO(),
    config.WithEndpointResolverWithOptions(customResolver))
```

Migración

Al migrar de la versión 1 a la versión 2 de la resolución de puntos de conexión, se aplican los siguientes principios generales:

- Devolver un [punto final](#) con el valor [HostnameImmutable](#) establecido en `false` equivale aproximadamente `BaseEndpoint` a configurarlo en la URL devuelta originalmente desde la versión 1 y dejarlo `EndpointResolverV2` como predeterminado.
- Devolver un punto final con el valor `HostnameImmutable` establecido en `true` equivale aproximadamente a implementar un punto final `EndpointResolverV2` que devuelva la URL devuelta originalmente desde la versión 1.
 - La excepción principal se da en el caso de las operaciones con prefijos de punto de conexión modelados. Más adelante se incluye una nota al respecto.

A continuación, se proporcionan ejemplos de estos casos.

Warning

Los puntos de conexión inmutables de la versión 1 y la resolución de la versión 2 no tienen un comportamiento equivalente. Por ejemplo, los reemplazos de firma para características personalizadas como S3 Object Lambda seguirían estableciéndose para los puntos de conexión inmutables devueltos mediante código de la versión 1, pero no se haría lo mismo en el caso de la versión 2.

Nota sobre los prefijos de host

Algunas operaciones se modelan con prefijos de host que se anteponen al punto de conexión resuelto. Este comportamiento debe funcionar en conjunto con el resultado de la `ResolveEndpoint V2` y, por lo tanto, el prefijo de host se seguirá aplicando a ese resultado.

Puede deshabilitar el prefijo de host del punto de conexión de forma manual mediante la aplicación de un middleware (consulte la sección de ejemplos).

Ejemplos

Punto de conexión mutable

En el ejemplo de código siguiente se muestra cómo migrar un solucionador de puntos de conexión básico de la versión 1 que devuelve un punto de conexión modificable:

```
// v1
client := svc.NewFromConfig(cfg, func (o *svc.Options) {
    o.EndpointResolver = svc.EndpointResolverFromURL("https://custom.endpoint.api/")
})

// v2
client := svc.NewFromConfig(cfg, func (o *svc.Options) {
    // the value of BaseEndpoint is passed to the default EndpointResolverV2
    // implementation, which will handle routing for features such as S3 accelerate,
    // MRAP, etc.
    o.BaseEndpoint = aws.String("https://custom.endpoint.api/")
})
```

Punto de conexión inmutable

```
// v1
client := svc.NewFromConfig(cfg, func (o *svc.Options) {
    o.EndpointResolver = svc.EndpointResolverFromURL("https://custom.endpoint.api/",
    func (e *aws.Endpoint) {
        e.HostnameImmutable = true
    })
})

// v2
import (
    smithyendpoints "github.com/aws/smithy-go/endpoints"
```

```

)

type staticResolver struct {}

func (*staticResolver) ResolveEndpoint(ctx context.Context, params
    svc.EndpointParameters) (
    smithyendpoints.Endpoint, error,
    ) {
    // This value will be used as-is when making the request.
    u, err := url.Parse("https://custom.endpoint.api/")
    if err != nil {
        return smithyendpoints.Endpoint{}, err
    }
    return smithyendpoints.Endpoint{
        URI: *u,
    }, nil
}

client := svc.NewFromConfig(cfg, func (o *svc.Options) {
    o.EndpointResolverV2 = &staticResolver{}
})

```

Deshabilitación del prefijo de host

```

import (
    "context"
    "fmt"
    "net/url"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/<service>"
    smithyendpoints "github.com/aws/smithy-go/endpoints"
    "github.com/aws/smithy-go/middleware"
    smithyhttp "github.com/aws/smithy-go/transport/http"
)

// disableEndpointPrefix applies the flag that will prevent any
// operation-specific host prefix from being applied
type disableEndpointPrefix struct{}

func (disableEndpointPrefix) ID() string { return "disableEndpointPrefix" }

```

```

func (disableEndpointPrefix) HandleInitialize(
    ctx context.Context, in middleware.InitializeInput, next
    middleware.InitializeHandler,
) (middleware.InitializeOutput, middleware.Metadata, error) {
    ctx = smithyhttp.SetHostnameImmutable(ctx, true)
    return next.HandleInitialize(ctx, in)
}

func addDisableEndpointPrefix(o *<service>.Options) {
    o.APIOptions = append(o.APIOptions, (func(stack *middleware.Stack) error {
        return stack.Initialize.Add(disableEndpointPrefix{}, middleware.After)
    })))
}

type staticResolver struct{}

func (staticResolver) ResolveEndpoint(ctx context.Context, params
    <service>.EndpointParameters) (
    smithyendpoints.Endpoint, error,
) {
    u, err := url.Parse("https://custom.endpoint.api/")
    if err != nil {
        return smithyendpoints.Endpoint{}, err
    }

    return smithyendpoints.Endpoint{URI: *u}, nil
}

func main() {
    cfg, err := config.LoadDefaultConfig(context.Background())
    if err != nil {
        panic(err)
    }

    svc := <service>.NewFromConfig(cfg, func(o *<service>.Options) {
        o.EndpointResolverV2 = staticResolver{}
    })

    _, err = svc.<Operation>(context.Background(), &<service>.<OperationInput>{ /* ...
    */ },
        addDisableEndpointPrefix)
    if err != nil {
        panic(err)
    }
}

```

```
}  
}
```

Personalización del cliente HTTP

AWS SDK para Go utiliza un cliente HTTP predeterminado con valores de configuración predeterminados. Si bien puede cambiar algunos de estos valores de configuración, el cliente y el transporte HTTP predeterminados no están suficientemente configurados para aquellos clientes que usan AWS SDK para Go en un entorno con requisitos de alto rendimiento y baja latencia. Para obtener más información, consulte la sección [Preguntas frecuentes](#), ya que las recomendaciones de configuración varían en función de las cargas de trabajo específicas. En esta sección se describe cómo configurar un cliente HTTP personalizado y cómo usarlo para crear llamadas a AWS SDK para Go.

Para ayudarle a crear un cliente HTTP personalizado, en esta sección se describe cómo usar [NewBuildableClient](#) a fin de configurar ajustes personalizados y cómo utilizar ese cliente con un cliente de servicio de AWS SDK para Go.

Vamos a definir qué queremos personalizar.

Reemplazo durante la carga de la configuración

Se pueden proporcionar clientes HTTP personalizados al llamar a [LoadDefaultConfig](#) mediante la encapsulación del cliente con [WithHTTPClient](#) y la transferencia del valor resultante a `LoadDefaultConfig`. Por ejemplo, para pasar `customClient` como nuestro cliente:

```
cfg, err := config.LoadDefaultConfig(context.TODO(),  
    config.WithHTTPClient(customClient))
```

Tiempo de espera

El objeto `BuildableHTTPClient` se puede configurar con un límite de tiempo de espera para las solicitudes. En este tiempo de espera se incluye el tiempo necesario para conectarse, procesar cualquier redireccionamiento y leer el cuerpo de respuesta completo. Por ejemplo, para modificar el tiempo de espera del cliente:

```
import "github.com/aws/aws-sdk-go-v2/aws/transport/http"
```

```
// ...  
  
httpClient := http.NewBuildableClient().WithTimeout(time.Second*5)
```

Dialer

El objeto `BuildableHTTPClient` proporciona una mecánica de creación para crear clientes con opciones [Dialer](#) modificadas. En el ejemplo siguiente se muestra cómo establecer la configuración `Dialer` de un cliente:

```
import awshttp "github.com/aws/aws-sdk-go-v2/aws/transport/http"  
import "net"  
  
// ...  
  
httpClient := awshttp.NewBuildableClient().WithDialerOptions(func(d *net.Dialer) {  
    d.KeepAlive = -1  
    d.Timeout = time.Millisecond*500  
})
```

Configuración

Dialer.KeepAlive

Esta configuración representa el período de conservación de una conexión de red activa.

Establézcala en un valor negativo para deshabilitar estos períodos.

Establézcala en 0 para habilitar los períodos de conservación si el protocolo y el sistema operativo los admiten.

Los protocolos de red o sistemas operativos que no admiten los períodos de conservación omiten este campo. De forma predeterminada, TCP habilita el período de conservación.

Consulte <https://golang.org/pkg/net/#Dialer.KeepAlive>.

Establezca `KeepAlive` como `time.Duration`.

Dialer.Timeout

Esta configuración representa el tiempo máximo de espera de marcado para la creación de una conexión.

El valor predeterminado es de 30 segundos.

Consulte <https://golang.org/pkg/net/#Dialer.Timeout>

Establezca Timeout como time.Duration.

Transporte

El objeto `BuildableHTTPClient` proporciona una mecánica de creación destinada a crear clientes con opciones [Transport](#) modificadas.

Configuración de un proxy

Si no puede conectarse directamente a Internet, puede utilizar variables de entorno compatibles con Go (`HTTP_PROXY/HTTPS_PROXY`) o crear un cliente HTTP personalizado para configurar el proxy. En el ejemplo siguiente se configura el cliente para usar `PROXY_URL` como punto de conexión del proxy:

```
import awshttp "github.com/aws/aws-sdk-go-v2/aws/transport/http"
import "net/http"

// ...

httpClient := awshttp.NewBuildableClient().WithTransportOptions(func(tr
    *http.Transport) {
    proxyURL, err := url.Parse("PROXY_URL")
    if err != nil {
        log.Fatal(err)
    }
    tr.Proxy = http.ProxyURL(proxyURL)
})
```

Otras configuraciones

A continuación, se muestran otras configuraciones de `Transport` que se pueden modificar para ajustar el cliente HTTP. En la documentación del tipo [Transport](#) encontrará configuraciones adicionales que no se describen aquí. Estas configuraciones se pueden aplicar como se muestra en el ejemplo siguiente:

```
import awshttp "github.com/aws/aws-sdk-go-v2/aws/transport/http"
import "net/http"
```

```
// ...

httpClient := awshttp.NewBuildableClient().WithTransportOptions(func(tr
    *http.Transport) {
    tr.ExpectContinueTimeout = 0
    tr.MaxIdleConns = 10
})
```

Transport.ExpectContinueTimeout

Si la solicitud tiene el encabezado “Expect: 100-continue”, esta configuración representa el tiempo máximo de espera para los primeros encabezados de respuesta de un servidor después de escribir por completo los encabezados de solicitud. En este tiempo no se incluye el tiempo necesario para enviar el encabezado de solicitud. El cliente HTTP envía su carga útil después de agotarse este tiempo de espera.

El valor predeterminado es de un segundo.

Establézcala en 0 para no definir ningún tiempo de espera y enviar la carga útil de la solicitud sin esperar. Un caso de uso se produce cuando tiene problemas con proxies o servicios de terceros que toman una sesión similar al uso de Amazon S3 en la función que se muestra más adelante.

Consulte <https://golang.org/pkg/net/http/#Transport.ExpectContinueTimeout>.

Establezca ExpectContinue como time.Duration.

Transport.IdleConnTimeout

Esta configuración representa el tiempo máximo para mantener activa una conexión de red inactiva entre solicitudes HTTP.

Establézcala en 0 para que no haya límite.

Consulte <https://golang.org/pkg/net/http/#Transport.IdleConnTimeout>.

Establezca IdleConnTimeout como time.Duration.

Transport.MaxIdleConns

Esta configuración representa el número máximo de conexiones inactivas (con mantenimiento activo) en todos los hosts. Un caso de uso para aumentar este valor es cuando se dan muchas conexiones de los mismos clientes en un período breve.

0 significa que no hay límite.

Consulte <https://golang.org/pkg/net/http/#Transport.MaxIdleConns>.

Establezca `MaxIdleConns` como `int`.

`Transport.MaxIdleConnsPerHost`

Esta configuración representa el número máximo de conexiones inactivas (con mantenimiento activo) que se conservan por host. Un caso de uso para aumentar este valor es cuando se dan muchas conexiones de los mismos clientes en un período breve.

El valor predeterminado es de dos conexiones inactivas por host.

Establézcala en 0 para usar `DefaultMaxIdleConnsPerHost` (2).

Consulte <https://golang.org/pkg/net/http/#Transport.MaxIdleConnsPerHost>.

Establezca `MaxIdleConnsPerHost` como `int`.

`Transport.ResponseHeaderTimeout`

Esta configuración representa el tiempo máximo de espera para que un cliente lea el encabezado de respuesta.

Si el cliente no puede leer el encabezado de la respuesta dentro de este período, se producirá un error de tiempo de espera de la solicitud.

Tenga cuidado al establecer este valor cuando utilice funciones de Lambda de ejecución prolongada, ya que la operación no devolverá ningún encabezado de respuesta hasta que la función de Lambda haya finalizado o se haya agotado su tiempo de espera. Sin embargo, puede seguir usando esta opción con la operación de la API `** InvokeAsync**`.

El valor predeterminado es no establecer un tiempo de espera, es decir, una espera indefinida.

Consulte <https://golang.org/pkg/net/http/#Transport.ResponseHeaderTimeout>.

Establezca `ResponseHeaderTimeout` como `time.Duration`.

`Transport.TLSHandshakeTimeout`

Esta configuración representa el tiempo máximo de espera para que se complete un establecimiento de comunicación TLS.

El valor predeterminado es de diez segundos.

Cero significa que no hay tiempo de espera.

Consulte <https://golang.org/pkg/net/http/#Transport.TLSHandshakeTimeout>.

Establezca `TLSHandshakeTimeout` como `time.Duration`.

Interceptores HTTP

Puede usar interceptores para conectarse a la ejecución de las solicitudes y respuestas de la API. Los interceptores son mecanismos abiertos en los que el SDK llama al código que escribe para inyectar un comportamiento en el ciclo de vida de las solicitudes o respuestas. De esta forma puede modificar una solicitud en tránsito, depurar el procesamiento de las solicitudes, ver las excepciones y mucho más.

Interceptores frente a middleware

AWS SDK para Go v2 proporciona tanto interceptores como middleware para personalizar el procesamiento de las solicitudes. Si bien ambos cumplen propósitos similares, están pensados para diferentes audiencias y casos de uso:

- Los interceptores están diseñados para los usuarios del SDK que desean personalizar el procesamiento de solicitudes y respuestas con una API sencilla y centrada en HTTP. Proporcionan puntos de enlace específicos en el ciclo de vida de las solicitudes y trabajan directamente con las solicitudes y las respuestas HTTP.
- El middleware es un sistema más avanzado e independiente del transporte que el SDK utiliza principalmente de forma interna. Si bien es potente, el middleware requiere un conocimiento más profundo de los aspectos internos del SDK e implica interfaces más complejas.

Las principales ventajas de los interceptores respecto al middleware para los casos de uso habituales son las siguientes:

- Centrados en HTTP: los interceptores trabajan directamente con las solicitudes y las respuestas HTTP, lo que elimina la necesidad de comprobar el tipo de transporte que requiere el middleware.
- Interfaces más sencillas: cada enlace de interceptor tiene una interfaz específica y centrada, en lugar del patrón de middleware genérico.

- Modelo de ejecución más claro: los interceptores se ejecutan en puntos bien definidos del ciclo de vida de las solicitudes sin necesidad de conocer la ordenación de pila del middleware.

Note

Los interceptores se basan en el sistema de middleware existente, por lo que ambos pueden coexistir en la misma aplicación. El middleware sigue estando disponible para casos de uso avanzados que requieren un comportamiento independiente del transporte o una manipulación compleja de la pila.

Enlaces de interceptor disponibles

AWS SDK para Go v2 proporciona enlaces de interceptor en varias etapas del ciclo de vida de las solicitudes. Cada enlace corresponde a una interfaz específica que puede implementar:

- `BeforeExecution`: primer enlace al que se llama durante la ejecución de la operación
- `BeforeSerialization`: antes de que el mensaje de entrada se serialice en una solicitud de transporte
- `AfterSerialization`: después de que el mensaje de entrada se serialice en una solicitud de transporte
- `BeforeRetryLoop`: antes de entrar en el bucle de reintentos
- `BeforeAttempt`: primer enlace al que se llama dentro del bucle de reintentos
- `BeforeSigning`: antes de firmarse la solicitud de transporte
- `AfterSigning`: después de firmarse la solicitud de transporte
- `BeforeTransmit`: antes de enviarse la solicitud de transporte
- `AfterTransmit`: después de recibir la respuesta del transporte
- `BeforeDeserialization`: antes de deserializar la respuesta del transporte
- `AfterDeserialization`: después de deserializar la respuesta del transporte
- `AfterAttempt`: último enlace al que se llama dentro del bucle de reintentos
- `AfterExecution`: último enlace al que se llama durante la ejecución de la operación

Puede implementar varias interfaces en un solo interceptor para conectarse a varias etapas del ciclo de vida de las solicitudes.

Registro de interceptores

Los interceptores se registran cuando se crea un cliente de servicio o cuando se anula la configuración de una operación específica. El registro varía en función de si desea que el interceptor se aplique a todas las operaciones de su cliente o solo a operaciones específicas.

Los interceptores se administran a través de un registro de interceptores en el que se proporcionan métodos para agregar y eliminar interceptores. En el siguiente ejemplo se muestra un interceptor sencillo que agrega un encabezado de ID de rastro AWS X-Ray a las solicitudes salientes antes del proceso de firma:

```
type recursionDetection struct{}

func (recursionDetection) BeforeSigning(ctx context.Context, in
    *smithyhttp.InterceptorContext) error {
    if traceID := os.Getenv("_X_AMZN_TRACE_ID"); traceID != "" {
        in.Request.Header.Set("X-Amzn-Trace-Id", traceID)
    }
    return nil
}

// use it on the client
svc := s3.NewFromConfig(cfg, func(o *s3.Options) {
    o.Interceptors.AddBeforeSigning(recursionDetection{})
})
```

El registro de interceptores se agrega a las opciones de cliente, lo que permite la configuración del interceptor por operación:

```
// ... or use it per-operation
s3.ListBuckets(context.Background(), &s3.ListBucketsInput{
}, func(o *s3.Options) {
    o.Interceptors.AddBeforeSigning(recursionDetection{})
})
```

Configuración global del interceptor

También puede registrar interceptores a nivel global mediante la función `config.LoadDefaultConfig` con las opciones `With*` adecuadas para cada tipo de interceptor. Esto aplica el interceptor a todos los clientes de servicio de AWS creados a partir de esa configuración:

```
type myExecutionInterceptor struct{}

func (*myExecutionInterceptor) AfterExecution(ctx context.Context, in
    *smithyhttp.InterceptorContext) error {
    // Add your custom logic here
    return nil
}

cfg, err := config.LoadDefaultConfig(context.Background(),
    config.WithAfterExecution(&myExecutionInterceptor{}))
if err != nil {
    panic(err)
}

// every service client created from the above config
// will include this interceptor
svc := s3.NewFromConfig(cfg)
```

Registro

AWS SDK para Go cuenta con servicios de registro que permiten a la aplicación habilitar información de depuración a fin de depurar y diagnosticar problemas o errores relacionados con las solicitudes. La interfaz [Logger](#) y [ClientLogMode](#) son los principales componentes disponibles para determinar cómo y qué deben registrar los clientes.

Logger

Al crear un objeto [Config](#) mediante [LoadDefaultConfig](#), se configura un objeto `Logger` predeterminado para enviar mensajes de registro al error estándar del proceso (`stderr`). Un registrador que cumpla con la interfaz [Logger](#) se puede pasar como argumento a `LoadDefaultConfig` mediante su encapsulación con [config.WithLogger](#).

Por ejemplo, para configurar nuestros clientes de forma que usen nuestro `applicationLogger`:

```
cfg, err := config.LoadDefaultConfig(context.TODO(),
    config.WithLogger(applicationLogger))
```

Ahora, los clientes configurados mediante la instancia de `aws.Config` creada enviarán mensajes de registro a `applicationLogger`.

Registradores dependientes del contexto

Una implementación de registrador puede implementar la interfaz [ContextLogger](#) opcional. Los métodos `WithContext` de los registradores que implementen esta interfaz se invocan con el contexto actual. Esto permite que sus implementaciones de registro devuelvan un `Logger` nuevo que puede escribir metadatos de registro adicionales en función de los valores presentes en el contexto.

ClientLogMode

De forma predeterminada, los clientes de servicio no generan mensajes de registro. Para configurar los clientes de forma que envíen mensajes de registro con fines de depuración, utilice el miembro [ClientLogMode](#) en `Config`. `ClientLogMode` se puede configurar para habilitar la mensajería de depuración para:

- Firma de Signature Version 4 (SigV4)
- Reintentos de solicitud
- Solicitudes HTTP
- Respuestas HTTP

Por ejemplo, para habilitar el registro de solicitudes y reintentos HTTP:

```
cfg, err := config.LoadDefaultConfig(context.TODO(),
    config.WithClientLogMode(aws.LogRetries | aws.LogRequest))
```

Consulte [ClientLogMode](#) para ver los diferentes modos de registro de clientes disponibles.

Reintentos y tiempos de espera

AWS SDK para Go Le permite configurar el comportamiento de reintento de las solicitudes a los servicios HTTP. De forma predeterminada, los clientes de servicio utilizan [retry.Standard](#) como elemento de reintento predeterminado. Si la configuración o el comportamiento predeterminado no cumple con los requisitos de la aplicación, puede ajustar la configuración del elemento de reintento o proporcionar su propia implementación del mismo.

AWS SDK para Go Proporciona una interfaz [AWS.Retryer](#) que define el conjunto de métodos que debe implementar una implementación de reintento. [El SDK proporciona dos implementaciones para los reintentos: `retry.standard` y `aws.NoOpRetryer`.](#)

Elemento de reintento estándar

El elemento de reintento [retry.Standard](#) es la implementación de `aws.Retryer` predeterminada que utilizan los clientes del SDK. El elemento de reintento estándar tiene ciertas limitaciones, con un número máximo de intentos configurable y la posibilidad de ajustar la política de retroceso de las solicitudes.

En la tabla siguiente se definen los valores predeterminados de este elemento de reintento:

Propiedad	Predeterminado
Número máximo de intentos	3
Retardo de retroceso máximo	20 segundos

Cuando se produzca un error reintentable al invocar la solicitud, el elemento de reintento estándar utilizará la configuración proporcionada a fin de retrasar la solicitud y reintentarla más adelante. Los reintentos se agregan a la latencia general de la solicitud y debe configurar el elemento de reintento si la configuración predeterminada no cumple los requisitos de la aplicación.

Consulte la documentación del paquete [retry](#) para obtener más información sobre los errores que la implementación del elemento de reintento estándar considera reintentables.

NopRetryer

Las [leyes. NopRetryer](#) es una `aws.Retryer` implementación que se proporciona si desea deshabilitar todos los reintentos. Al invocar una operación de cliente de servicio, este elemento de reintento solo permite que la solicitud se intente una vez y cualquier error resultante se devolverá a la aplicación que realiza la llamada.

Personalización del comportamiento

El SDK proporciona un conjunto de utilidades auxiliares que encapsulan una implementación de `aws.Retryer` y devuelve el elemento de reintento que se proporciona, encapsulado con el comportamiento de reintento deseado. Puede reemplazar el elemento de reintento predeterminado para todos los clientes, por cliente o por operación, en función de los requisitos de las aplicaciones. Para ver más ejemplos en los que se muestra cómo realizar esta acción, consulte los ejemplos de la documentación del paquete [retry](#).

⚠ Warning

Si especifica una implementación de `aws.Retryer` global mediante `config.WithRetryer`, debe asegurarse de que se devuelva una nueva instancia de `aws.Retryer` en cada invocación. De esta forma, se garantiza que no se cree un bucket de tokens de reintentos global en todos los clientes de servicio.

Limitación del número máximo de intentos

Usas [`retry.AddWithMaxAttempts`](#) para ajustar una `aws.Retryer` implementación a fin de establecer el número máximo de intentos en el valor deseado.

⚠ Warning

Si se usa `retry.AddWithMaxAttempts` con un valor de cero, el SDK podrá volver a intentar todos los errores reintentables hasta que la solicitud se realice correctamente o se devuelva un error que no sea reintentable. Permitir que el SDK realice reintentos de forma infinita puede provocar cargas de trabajo fuera de control y ciclos de facturación desorbitados.

```
import "context"
import "github.com/aws/aws-sdk-go-v2/aws/retry"
import "github.com/aws/aws-sdk-go-v2/config"
import "github.com/aws/aws-sdk-go-v2/service/s3"

// ...

// MaxAttempts will be infinite (will retry indefinitely)
cfg, err := config.LoadDefaultConfig(context.TODO(), config.WithRetryer(func()
    aws.Retryer {
        return retry.AddWithMaxAttempts(retry.NewStandard(), 0)
    })
if err != nil {
    return err
}

client := s3.NewFromConfig(cfg)
```

Tenga en cuenta que crear una instancia de un retryer utilizando opciones funcionales para configurarlo `MaxAttempts` directamente tendrá un comportamiento ligeramente diferente. En concreto, si se establece un valor menor que cero o cero, el elemento de reintento utilizará el máximo predeterminado de tres intentos, en lugar de hacer reintentos de forma infinita:

```
import "context"
import "github.com/aws/aws-sdk-go-v2/aws/retry"
import "github.com/aws/aws-sdk-go-v2/config"
import "github.com/aws/aws-sdk-go-v2/service/s3"

// ...

// MaxAttempts will default to 3
cfg, err := config.LoadDefaultConfig(context.TODO(), config.WithRetryer(func()
    aws.Retryer {
        return retry.NewStandard(func(o *retry.StandardOptions) {
            o.MaxAttempts = 0
        })
    })
if err != nil {
    return err
}

client := s3.NewFromConfig(cfg)
```

Limitación del retardo de retroceso máximo

[Usas `retry.AddWithMaxBackoffDelay`](#) para completar una `aws.Retryer` implementación y limitar el tiempo máximo de espera que se puede producir entre el reintento de una solicitud fallida.

Por ejemplo, puede usar el código siguiente para encapsular el elemento de reintento del cliente estándar con un retraso máximo deseado de cinco segundos:

```
import "context"
import "time"
import "github.com/aws/aws-sdk-go-v2/aws/retry"
import "github.com/aws/aws-sdk-go-v2/config"
import "github.com/aws/aws-sdk-go-v2/service/s3"

// ...
```



```
cfg, err := config.LoadDefaultConfig(context.TODO(), config.WithRetryer(func()
    aws.Retryer {
        return retry.AddWithMaxBackoffDelay(retry.NewStandard(), time.Second*5)
    })
if err != nil {
    return err
}

client := s3.NewFromConfig(cfg)
```

Reintento de códigos de error adicionales de la API

Se usa [reintentar. AddWithErrorCodes](#) para empaquetar una `aws.Retryer` implementación e incluir códigos de error de API adicionales que deberían considerarse reintentables.

Por ejemplo, puede usar el código siguiente para encapsular el elemento de reintento del cliente estándar e incluir la excepción `NoSuchBucketException` de Amazon S3 como reintentable.

```
import "context"
import "time"
import "github.com/aws/aws-sdk-go-v2/aws/retry"
import "github.com/aws/aws-sdk-go-v2/config"
import "github.com/aws/aws-sdk-go-v2/service/s3"
import "github.com/aws/aws-sdk-go-v2/service/s3/types"

// ...

cfg, err := config.LoadDefaultConfig(context.TODO(), config.WithRetryer(func()
    aws.Retryer {
        return retry.AddWithErrorCodes(retry.NewStandard(), (*types.NoSuchBucketException)
        (nil).ErrorCode())
    })
if err != nil {
    return err
}

client := s3.NewFromConfig(cfg)
```

Limitación de la tasa del cliente

AWS SDK para Go Introduce un nuevo mecanismo de limitación de la velocidad por parte del cliente en la política de reintentos estándar para adaptarlo al comportamiento de la tecnología moderna.

SDKs [Este comportamiento lo controla el RateLimiter campo de las opciones de un retryer.](#)

A RateLimiter funciona como un depósito de fichas con una capacidad determinada, en el que los intentos fallidos de operación consumen fichas. Si se vuelve a intentar consumir más fichas de las disponibles, se produce un error en la operación con un [QuotaExceededError](#).

La implementación predeterminada está parametrizada de la siguiente forma (cómo modificar cada configuración):

- una capacidad de 500 (establece el valor de RateLimiter al StandardOptions usarla [NewTokenRateLimit](#))
- un reintento provocado por un tiempo de espera cuesta 10 fichas (un número fijo RetryTimeoutCost) StandardOptions
- un reintento provocado por otros errores cuesta 5 fichas (activado) RetryCost StandardOptions
- una operación que se realiza correctamente en el primer intento añade 1 ficha (activada NoRetryIncrement) StandardOptions
 - Las operaciones que se realizan correctamente en el segundo intento o en un intento posterior no agregan tokens

Si considera que el comportamiento predeterminado no se ajusta a las necesidades de su aplicación, puede deshabilitarlo con [ratelimit.None](#).

Ejemplo: Limitador de tasa modificado

```
import (  
    "context"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/aws/ratelimit"  
    "github.com/aws/aws-sdk-go-v2/aws/retry"  
    "github.com/aws/aws-sdk-go-v2/config"  
)  
  
// ...
```

```

cfg, err := config.LoadDefaultConfig(context.Background(), config.WithRetryer(func()
aws.Retryer {
    return retry.NewStandard(func(o *retry.StandardOptions) {
        // Makes the rate limiter more permissive in general. These values are
        // arbitrary for demonstration and may not suit your specific
        // application's needs.
        o.RateLimiter = ratelimit.NewTokenRateLimit(1000)
        o.RetryCost = 1
        o.RetryTimeoutCost = 3
        o.NoRetryIncrement = 10
    })
}))

```

Ejemplo: Sin límite de tasa con ratelimit.None

```

import (
    "context"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/aws/ratelimit"
    "github.com/aws/aws-sdk-go-v2/aws/retry"
    "github.com/aws/aws-sdk-go-v2/config"
)

// ...

cfg, err := config.LoadDefaultConfig(context.Background(), config.WithRetryer(func()
aws.Retryer {
    return retry.NewStandard(func(o *retry.StandardOptions) {
        o.RateLimiter = ratelimit.None
    })
}))

```

Tiempos de espera

El paquete [context](#) se utiliza para establecer tiempos de espera o plazos al invocar una operación de cliente de servicio. Usa el [contexto. WithDeadline](#) para resumir el contexto de su solicitud y establecer una fecha límite en un momento específico para completar la operación invocada. Para establecer un tiempo de espera después de un determinado [contexto de time.Duration uso. WithTimeout](#). El SDK pasa el elemento `context.Context` proporcionado al cliente de transporte HTTP al invocar una API de servicio. Si el contexto que se pasa al SDK se cancela, o se produce su cancelación durante la invocación de la operación, el SDK no reintentará realizar la solicitud y volverá a la

aplicación que realiza la llamada. La cancelación del contexto debe gestionarse de forma adecuada en la aplicación en los casos en los que se haya cancelado el contexto proporcionado al SDK.

Establecimiento de un tiempo de espera

En el ejemplo siguiente se muestra cómo establecer un tiempo de espera para una operación de cliente de servicio.

```
import "context"
import "time"

// ...

ctx := context.TODO() // or appropriate context.Context value for your application

client := s3.NewFromConfig(cfg)

// create a new context from the previous ctx with a timeout, e.g. 5 seconds
ctx, cancel := context.WithTimeout(ctx, 5*time.Second)
defer cancel()

resp, err := client.GetObject(ctx, &s3.GetObjectInput{
    // input parameters
})
if err != nil {
    // handle error
}
```

Uso de AWS SDK para Go

Aprenda formas comunes y recomendadas de programar con AWS SDK para Go en las aplicaciones.

Temas

- [Creación de un cliente de servicio](#)
- [Llamada a operaciones de servicio](#)
- [Uso simultáneo de clientes de servicio](#)
- [Uso de paginadores de operaciones](#)
- [Uso de esperadores](#)
- [Gestión de errores en AWS SDK para Go V2](#)

Creación de un cliente de servicio

Los clientes de servicio se pueden crear mediante las funciones `New` o `NewFromConfig` disponibles en el paquete Go del cliente de servicio. Cada función devolverá un tipo de estructura `Client` que contiene los métodos para invocar las API de servicio. `New` y `NewFromConfig` proporcionan el mismo conjunto de opciones configurables para crear un cliente de servicio, pero ofrecen patrones de creación distintos que veremos en las secciones posteriores.

NewFromConfig

La función `NewFromConfig` proporciona una interfaz coherente para crear clientes de servicio mediante [aws.Config](#). `aws.Config` se puede cargar mediante [config.LoadDefaultConfig](#). Para obtener más información sobre cómo crear una instancia de `aws.Config`, consulte [Configurar el SDK](#). En el siguiente ejemplo se muestra cómo crear un cliente de servicio Amazon S3 mediante `aws.Config` y la función `NewFromConfig`:

```
import "context"
import "github.com/aws/aws-sdk-go-v2/config"
import "github.com/aws/aws-sdk-go-v2/service/s3"

// ...
```

```
cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    panic(err)
}

client := s3.NewFromConfig(cfg)
```

Reemplazo de la configuración

`NewFromConfig` puede tomar uno o más argumentos funcionales que pueden modificar la estructura `Options` de configuración de un cliente. Esto le permite realizar reemplazos específicos, como cambiar la región o modificar opciones específicas del servicio (por ejemplo, la opción `UseAccelerate` de Amazon S3). Por ejemplo:

```
import "context"
import "github.com/aws/aws-sdk-go-v2/config"
import "github.com/aws/aws-sdk-go-v2/service/s3"

// ...

cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    panic(err)
}

client := s3.NewFromConfig(cfg, func(o *s3.Options) {
    o.Region = "us-west-2"
    o.UseAccelerate = true
}))
```

Los reemplazos del valor `Options` del cliente vienen determinados por el orden en el que se asignan los argumentos funcionales a `NewFromConfig`.

New

Note

`New` se considera una forma más avanzada de creación de clientes. Le recomendamos que utilice `NewFromConfig` para la creación de clientes, ya que permite crear con la estructura

de `aws.Config`. Esto elimina la necesidad de crear una instancia de estructura `Options` para cada cliente de servicio que requiera la aplicación.

`New` es una función creadora de clientes que proporciona una interfaz de creación de clientes con el uso exclusivo de la estructura `Options` de los paquetes de clientes para definir las opciones de configuración del cliente. Por ejemplo, para crear un cliente de Amazon S3 mediante `New`:

```
import "github.com/aws/aws-sdk-go-v2/aws"
import "github.com/aws/aws-sdk-go-v2/credentials"
import "github.com/aws/aws-sdk-go-v2/service/s3"

// ...

client := s3.New(s3.Options{
    Region:      "us-west-2",
    Credentials:
aws.NewCredentialsCache(credentials.NewStaticCredentialsProvider(accessKey, secretKey,
"")),
})
```

Reemplazo de la configuración

`New` puede tomar uno o más argumentos funcionales que pueden modificar la estructura `Options` de configuración de un cliente. Esto le permite realizar reemplazos específicos, como cambiar la región o modificar opciones específicas del servicio (por ejemplo, la opción `UseAccelerate` de Amazon S3). Por ejemplo:

```
import "github.com/aws/aws-sdk-go-v2/aws"
import "github.com/aws/aws-sdk-go-v2/credentials"
import "github.com/aws/aws-sdk-go-v2/service/s3"

// ...

options := s3.Options{
    Region:      "us-west-2",
    Credentials:
aws.NewCredentialsCache(credentials.NewStaticCredentialsProvider(accessKey, secretKey,
"")),
}
```

```
client := s3.New(options, func(o *s3.Options) {
    o.Region = "us-east-1"
    o.UseAccelerate = true
})
```

Los reemplazos del valor `Options` del cliente vienen determinados por el orden en el que se asignan los argumentos funcionales a `New`.

Llamada a operaciones de servicio

Una vez que tenga una instancia de cliente de servicio, puede usarla para llamar a las operaciones de un servicio. Por ejemplo, para llamar a la operación `GetObject` de Amazon S3:

```
response, err := client.GetObject(context.TODO(), &s3.GetObjectInput{
    Bucket: aws.String("amzn-s3-demo-bucket"),
    Key:    aws.String("obj-key"),
})
```

Al llamar a una operación de servicio, el SDK valida de forma sincrónica la entrada, serializa la solicitud, la firma con sus credenciales, la envía a AWS y, a continuación, deserializa una respuesta o un error. En la mayoría de los casos, puede llamar a las operaciones de servicio de forma directa. Cada método de cliente para operaciones de servicio devolverá una estructura de respuesta de la operación y un tipo de interfaz de error. Compruebe siempre el tipo `error` para determinar si se ha producido un error antes de intentar acceder a la estructura de respuesta de la operación de servicio.

Transferencia de parámetros a una operación de servicio

Cada método de operación de servicio toma un valor [context.Context](#) que se puede utilizar para establecer plazos de solicitud que el SDK cumplirá. Además, cada operación de servicio tomará una estructura `<OperationName>Input` que se encuentra en el paquete Go respectivo del servicio. Para transferir los parámetros de entrada de la API use la estructura de entrada de la operación.

Las estructuras de entrada de la operación pueden tener parámetros de entrada como los tipos numérico, booleano, de cadena, de mapa y de lista Go estándar. En operaciones de la API más complejas, un servicio puede tener un modelado más complejo de los parámetros de entrada. Estos

otros tipos, como valores de enumeración y estructuras específicas del servicio, se encuentran en el paquete Go `types` del servicio.

Además, los servicios pueden distinguir entre el valor predeterminado de un tipo Go y si el valor lo ha establecido el usuario o no. En estos casos, puede que los parámetros de entrada requieran que pases una referencia del puntero al tipo en cuestión. En el caso de los tipos Go estándar (por ejemplo, numéricos, booleanos y de cadena), existen las funciones prácticas `<Type>` y `From<Type>` disponibles en [aws](#) para facilitar esta conversión. Por ejemplo, [aws.String](#) se puede usar para convertir `string` en un tipo `*string` para los parámetros de entrada que requieren un puntero a una cadena. Por el contrario, [aws.ToString](#) se puede usar para transformar un elemento `*string` en `string`, al tiempo que se brinda protección frente a la cancelación de referencias a un puntero nulo. Las funciones `To<Type>` son útiles a la hora de gestionar las respuestas del servicio.

Veamos un ejemplo de cómo podemos usar un cliente de Amazon S3 para llamar a la API `GetObject` y crear nuestra entrada mediante el paquete `types` y los ayudantes `aws.<Type>`.

```
import "context"
import "github.com/aws/aws-sdk-go-v2/config"
import "github.com/aws/aws-sdk-go-v2/service/s3"
import "github.com/aws/aws-sdk-go-v2/service/s3/types"

// ...

cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    panic(err)
}

client := s3.NewFromConfig(cfg)

resp, err := client.GetObject(context.TODO(), &s3.GetObjectInput{
    Bucket:      aws.String("amzn-s3-demo-bucket"),
    Key:         aws.String("keyName"),
    RequestPayer: types.RequestPayerRequester,
})
```

Reemplazo de opciones de cliente para la llamada a operaciones

De forma similar a la que se usa para modificar las opciones de operación de cliente durante la creación de un cliente con argumentos funcionales, las opciones de cliente pueden modificarse en

el momento en el que se llama al método de operación al proporcionar uno o varios argumentos funcionales al método de operación de servicio. Esta acción se puede ejecutar a la vez que otras de forma segura y no afectará a otras operaciones simultáneas del cliente.

Por ejemplo, para reemplazar la región del cliente “us-west-2” por “us-east-1”:

```
cfg, err := config.LoadDefaultConfig(context.TODO(), config.WithRegion("us-west-2"))
if err != nil {
    log.Printf("error: %v", err)
    return
}

client := s3.NewFromConfig(cfg)

params := &s3.GetObjectInput{
    // ...
}

resp, err := client.GetObject(context.TODO(), params, func(o *Options) {
    o.Region = "us-east-1"
})
```

Gestión de respuestas de operación

Cada operación de servicio tiene una estructura de salida asociada que contiene los miembros de respuesta de la operación del servicio. La estructura de salida sigue el patrón de nomenclatura <OperationName>Output. Puede que algunas operaciones no tengan ningún miembro definido para su salida de operación. Tras llamar a una operación de servicio, siempre se debe comprobar el tipo de argumento error devuelto para determinar si se ha producido un error al invocar la operación de servicio. Los errores devueltos pueden abarcar desde errores de validación de entradas del cliente hasta respuestas de error del servicio devueltas al cliente. No se debe acceder a la estructura de salida de la operación en caso de que el cliente devuelva un error que no sea nulo.

Por ejemplo, para registrar un error de operación y volver de forma prematura de la función de llamada:

```
response, err := client.GetObject(context.TODO())
if err != nil {
    log.Printf("GetObject error: %v", err)
}
```

```
    return  
}
```

Para obtener más información sobre la gestión de errores, incluida la forma de inspeccionar tipos de error específicos, consulte TODO.

Respuestas con **io.ReadCloser**

Algunas operaciones de la API devuelven una estructura de respuesta que contiene un miembro de salida que es `io.ReadCloser`. Este será el caso de las operaciones de la API que expongan algún elemento de su salida en el cuerpo de la propia respuesta HTTP.

Por ejemplo, la operación `GetObject` de Amazon S3 devuelve una respuesta cuyo miembro `Body` es un objeto `io.ReadCloser` para acceder a la carga útil del objeto.

Warning

SIEMPRE DEBE implementar `Close()` en cualquier miembro de salida `io.ReadCloser`, independientemente de si ha consumido su contenido. De lo contrario, se pueden filtrar recursos y crear posibles problemas al leer los cuerpos de las respuestas en operaciones a las que se llame más adelante.

```
resp, err := s3svc.GetObject(context.TODO(), &s3.GetObjectInput{...})  
if err != nil {  
    // handle error  
    return  
}  
// Make sure to always close the response Body when finished  
defer resp.Body.Close()  
  
decoder := json.NewDecoder(resp.Body)  
if err := decoder.Decode(&myStruct); err != nil {  
    // handle error  
    return  
}
```

Metadatos de respuesta

Todas las estructuras de salida de las operaciones de servicio incluyen un miembro `ResultMetadata` de tipo [middleware.Metadata](#). El middleware del SDK utiliza `middleware.Metadata` para proporcionar información adicional a partir de una respuesta del servicio no modelada por este. Esto incluye metadatos como el valor `RequestID`. Por ejemplo, para recuperar el valor `RequestID` asociado a una respuesta del servicio para ayudar a AWS Support a solucionar los problemas de una solicitud:

```
import "fmt"
import "log"
import "github.com/aws/aws-sdk-go-v2/aws/middleware"
import "github.com/aws/aws-sdk-go-v2/service/s3"

// ..

resp, err := client.GetObject(context.TODO(), &s3.GetObjectInput{
    // ...
})
if err != nil {
    log.Printf("error: %v", err)
    return
}

requestID, ok := middleware.GetRequestIDMetadata(resp.ResultMetadata)
if !ok {
    fmt.Println("RequestID not included with request")
}

fmt.Printf("RequestID: %s\n", requestID)
```

Uso simultáneo de clientes de servicio

Puede crear rutinas de Go que utilicen el mismo cliente de servicio de forma simultánea para enviar varias solicitudes. Un cliente de servicio se puede usar con tantas rutinas de Go como desee.

En el siguiente ejemplo, se utiliza un cliente de servicio Amazon S3 en varias rutinas de Go. En este ejemplo se cargan dos objetos en un bucket de Amazon S3 de forma simultánea.

```
import "context"
```

```

import "log"
import "strings"
import "github.com/aws/aws-sdk-go-v2/config"
import "github.com/aws/aws-sdk-go-v2/service/s3"

// ...

cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    log.Printf("error: %v", err)
    return
}

client := s3.NewFromConfig(cfg)

type result struct {
    Output *s3.PutObjectOutput
    Err    error
}

results := make(chan result, 2)

var wg sync.WaitGroup
wg.Add(2)

go func() {
    defer wg.Done()
    output, err := client.PutObject(context.TODO(), &s3.PutObjectInput{
        Bucket: aws.String("amzn-s3-demo-bucket"),
        Key:    aws.String("foo"),
        Body:   strings.NewReader("foo body content"),
    })
    results <- result{Output: output, Err: err}
}()

go func() {
    defer wg.Done()
    output, err := client.PutObject(context.TODO(), &s3.PutObjectInput{
        Bucket: aws.String("amzn-s3-demo-bucket"),
        Key:    aws.String("bar"),
        Body:   strings.NewReader("bar body content"),
    })
    results <- result{Output: output, Err: err}
}()

```

```
wg.Wait()

close(results)

for result := range results {
    if result.Err != nil {
        log.Printf("error: %v", result.Err)
        continue
    }
    fmt.Printf("etag: %v", aws.ToString(result.Output.ETag))
}
```

Uso de paginadores de operaciones

Por lo general, al recuperar una lista de elementos, es posible que tenga que comprobar la estructura de salida en busca de un token o marcador para confirmar si el servicio de AWS ha devuelto todos los resultados de su solicitud. Si el token o marcador está presente, se utiliza para solicitar la siguiente página de resultados. En lugar de administrar estos tokens o marcadores, puede usar los tipos de paginadores disponibles del paquete de servicios.

Hay ayudantes de paginadores disponibles para las operaciones de servicio admitidas y los encontrará en el paquete Go del cliente de servicio. A fin de crear un paginador para una operación admitida, utilice la función `New<OperationName>Paginator`. Las funciones de creación de paginadores toman el valor de `Client` del servicio, los parámetros de entrada `<OperationName>Input` de la operación y un conjunto opcional de argumentos funcionales que permiten configurar otros ajustes opcionales del paginador.

El tipo de paginador de operaciones devuelto ofrece una forma cómoda de recorrer en iteración una operación paginada hasta que llegue a la última página o encuentre los elementos que buscaba la aplicación. Un tipo de paginador tiene dos métodos: `HasMorePages` y `NextPage`. `HasMorePages` devuelve un valor booleano de `true` si no se ha recuperado la primera página o si hay páginas adicionales disponibles para su recuperación mediante la operación. Para recuperar la primera página de la operación o páginas posteriores, se debe llamar a la operación `NextPage`. `NextPage` toma `context.Context` y devuelve la salida de la operación y cualquier error correspondiente. Del mismo modo que se hace con los parámetros devueltos del método de operación de cliente, el error devuelto debe comprobarse siempre antes de intentar usar la estructura de respuesta devuelta. Consulte [Gestión de respuestas de operación](#).

En el siguiente ejemplo se utiliza el paginador `ListObjectsV2` para enumerar hasta tres páginas de claves de objeto de la operación `ListObjectV2`. Cada página consta de un máximo de diez claves, que se define mediante la opción de paginador `Limit`.

```
import "context"
import "log"
import "github.com/aws/aws-sdk-go-v2/config"
import "github.com/aws/aws-sdk-go-v2/aws"
import "github.com/aws/aws-sdk-go-v2/service/s3"

// ...

cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    log.Printf("error: %v", err)
    return
}

client := s3.NewFromConfig(cfg)

params := &s3.ListObjectsV2Input{
    Bucket: aws.String("amzn-s3-demo-bucket"),
}

paginator := s3.NewListObjectsV2Paginator(client, params, func(o
*s3.ListObjectsV2PaginatorOptions) {
    o.Limit = 10
})

pageNum := 0
for paginator.HasMorePages() && pageNum < 3 {
    output, err := paginator.NextPage(context.TODO())
    if err != nil {
        log.Printf("error: %v", err)
        return
    }
    for _, value := range output.Contents {
        fmt.Println(*value.Key)
    }
    pageNum++
}
```

Al igual que sucede con el método de operación de cliente, se pueden modificar opciones de cliente como la región de la solicitud. Para ello, se proporcionan uno o varios argumentos funcionales a `NextPage`. Para obtener más información sobre cómo reemplazar las opciones de cliente al llamar a una operación, consulte [Reemplazo de opciones de cliente para la llamada a operaciones](#).

Uso de esperadores

Al interactuar con las API de AWS que son asíncronas, a menudo es necesario esperar a que un recurso concreto esté disponible para poder realizar más acciones en este.

Por ejemplo, la API `CreateTable` de Amazon DynamoDB devuelve una respuesta inmediata con el valor `CREATING` para `TableStatus` y el usuario no puede invocar operaciones de lectura o escritura hasta que el estado de la tabla haya cambiado a `ACTIVE`.

Escribir lógica para sondear continuamente el estado de la tabla puede resultar engorroso y aumentar la probabilidad de errores. Los esperadores contribuyen a simplificar este proceso y son API sencillas que gestionan la tarea de sondeo de forma automática.

Por ejemplo, puede utilizar esperadores para sondear si se ha creado una tabla de DynamoDB y si está lista para una operación de escritura.

```
import "context"
import "fmt"
import "log"
import "time"
import "github.com/aws/aws-sdk-go-v2/aws"
import "github.com/aws/aws-sdk-go-v2/config"
import "github.com/aws/aws-sdk-go-v2/service/dynamodb"

// ...

cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    log.Printf("error: %v", err)
    return
}

client := dynamodb.NewFromConfig(cfg)

// we create a waiter instance by directly passing in a client
// that satisfies the waiters client Interface.
```



```
waiter := dynamodb.NewTableExistsWaiter(client)

// params is the input to api operation used by the waiter
params := &dynamodb.DescribeTableInput {
    TableName: aws.String("test-table")
}

// maxWaitTime is the maximum wait time, the waiter will wait for
// the resource status.
maxWaitTime := 5 * time.Minutes

// Wait will poll until it gets the resource status, or max wait time
// expires.
err := waiter.Wait(context.TODO(), params, maxWaitTime)
if err != nil {
    log.Printf("error: %v", err)
    return
}
fmt.Println("Dynamodb table is now ready for write operations")
```

Reemplazo de la configuración de un esperador

De forma predeterminada, el SDK utiliza el valor de retraso mínimo y retraso máximo configurado con valores óptimos definidos por los servicios de AWS para diferentes API. Puede reemplazar la configuración del esperador; para ello, proporcione opciones funcionales durante el proceso de creación de este o al invocar una operación de esperador.

Por ejemplo, para reemplazar la configuración del esperador durante el proceso de creación de este:

```
import "context"
import "fmt"
import "log"
import "time"
import "github.com/aws/aws-sdk-go-v2/aws"
import "github.com/aws/aws-sdk-go-v2/config"
import "github.com/aws/aws-sdk-go-v2/service/dynamodb"

// ...

cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    log.Printf("error: %v", err)
```

```

    return
}

client := dynamodb.NewFromConfig(cfg)

// we create a waiter instance by directly passing in a client
// that satisfies the waiters client Interface.
waiter := dynamodb.NewTableExistsWaiter(client, func (o
*dynamodb.TableExistsWaiterOptions) {

    // override minimum delay to 10 seconds
    o.MinDelay = 10 * time.Second

    // override maximum default delay to 300 seconds
    o.MaxDelay = 300 * time.Second
})

```

La función `Wait` de cada esperador también incluye opciones funcionales. Al igual que en el ejemplo anterior, puede reemplazar la configuración del esperador por solicitud `Wait`.

```

// params is the input to api operation used by the waiter
params := &dynamodb.DescribeTableInput {
    TableName: aws.String("test-table")
}

// maxWaitTime is the maximum wait time, the waiter will wait for
// the resource status.
maxWaitTime := 5 * time.Minutes

// Wait will poll until it gets the resource status, or max wait time
// expires.
err := waiter.Wait(context.TODO(), params, maxWaitTime, func (o
*dynamodb.TableExistsWaiterOptions) {

    // override minimum delay to 5 seconds
    o.MinDelay = 5 * time.Second

    // override maximum default delay to 120 seconds
    o.MaxDelay = 120 * time.Second
})
if err != nil {
    log.Printf("error: %v", err)
}

```

```
    return
}
fmt.Println("Dynamodb table is now ready for write operations")
```

Reemplazo avanzado de la configuración de un esperador

También puede personalizar el comportamiento predeterminado del esperador mediante una función reintentable personalizada. Las opciones específicas del esperador también proporcionan `APIOptions` para [personalizar los middlewares de las operaciones](#).

Por ejemplo, para configurar reemplazos avanzados del esperador:

```
import "context"
import "fmt"
import "log"
import "time"
import "github.com/aws/aws-sdk-go-v2/aws"
import "github.com/aws/aws-sdk-go-v2/config"
import "github.com/aws/aws-sdk-go-v2/service/dynamodb"
import "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
// ...

cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    log.Printf("error: %v", err)
    return
}

client := dynamodb.NewFromConfig(cfg)

// custom retryable defines if a waiter state is retryable or a terminal state.
// For example purposes, we will configure the waiter to not wait
// if table status is returned as `UPDATING`
customRetryable := func(ctx context.Context, params *dynamodb.DescribeTableInput,
    output *dynamodb.DescribeTableOutput, err error) (bool, error) {
    if output.Table != nil {
        if output.Table.TableStatus == types.TableStatusUpdating {
            // if table status is `UPDATING`, no need to wait
            return false, nil
        }
    }
}

}
```

```
// we create a waiter instance by directly passing in a client
// that satisfies the waiters client Interface.
waiter := dynamodb.NewTableExistsWaiter(client, func (o
*dynamodb.TableExistsWaiterOptions) {

    // override the service defined waiter-behavior
    o.Retryable = customRetryable
})
```

Gestión de errores en AWS SDK para Go V2

AWS SDK para Go devuelve errores que cumplen con el tipo de interfaz `error` de Go. Puede usar el método `Error()` para obtener una cadena con formato del mensaje de error del SDK sin ningún tratamiento especial. Los errores que devuelve el SDK pueden implementar un método `Unwrap`. El SDK utiliza el método `Unwrap` para proporcionar información contextual adicional sobre los errores y, al mismo tiempo, proporcionar acceso al error o la cadena de errores subyacentes. El método `Unwrap` debe usarse con [Errors.as](#) para gestionar la desencapsulación de las cadenas de error.

Es importante que la aplicación compruebe si se ha producido un error después de invocar una función o un método que pueda devolver un tipo de interfaz `error`. La forma más básica de gestión de errores es similar al ejemplo siguiente:

```
if err != nil {
    // Handle error
    return
}
```

Registro de errores

Tradicionalmente, la forma más sencilla de gestionar errores consiste en registrar o imprimir el mensaje de error antes de volver a la aplicación o salir de ella.

```
import "log"

// ...

if err != nil {
    log.Printf("error: %s", err.Error())
}
```

```
    return
}
```

Errores del cliente de servicio

El SDK encapsula todos los errores devueltos por clientes de servicio con el tipo de error [smithy.OperationError](#). `OperationError` proporciona información contextual sobre el nombre del servicio y la operación asociada a un error subyacente. Esta información puede resultar útil para las aplicaciones que realizan lotes de operaciones en uno o varios servicios, con un mecanismo de gestión de errores centralizado. Su aplicación puede utilizar `errors.As` para acceder a estos metadatos `OperationError`.

```
import "log"
import "github.com/aws/smithy-go"

// ...

if err != nil {
    var oe *smithy.OperationError
    if errors.As(err, &oe) {
        log.Printf("failed to call service: %s, operation: %s, error: %v",
            oe.Service(), oe.Operation(), oe.Unwrap())
    }
    return
}
```

Respuestas de error de la API

Las operaciones de servicio pueden devolver tipos de errores modelados para indicar errores específicos. Estos tipos modelados se pueden utilizar con `errors.As` para desencapsular y determinar si el fallo de la operación se ha debido a un error específico. Por ejemplo, el uso de `CreateBucket` de Amazon S3 puede devolver un error [BucketAlreadyExists](#) si existe un bucket con el mismo nombre.

Por ejemplo, para comprobar si un error fue del tipo `BucketAlreadyExists`:

```
import "log"
import "github.com/aws/aws-sdk-go-v2/service/s3/types"

// ...
```

```
if err != nil {
    var bne *types.BucketAlreadyExists
    if errors.As(err, &bne) {
        log.Println("error:", bne)
    }
    return
}
```

Todos los errores de respuesta de la API de servicio implementan el tipo de interfaz [smithy.APIError](#). Esta interfaz se puede utilizar para gestionar tanto las respuestas de error de servicio modeladas como las no modeladas. Este tipo proporciona acceso al código y al mensaje de error que devuelve el servicio. Además, este tipo indica si el error fue responsabilidad del cliente o del servidor, en caso de conocerse.

```
import "log"
import "github.com/aws/smithy-go"

// ...

if err != nil {
    var ae smithy.APIError
    if errors.As(err, &ae) {
        log.Printf("code: %s, message: %s, fault: %s", ae.ErrorCode(),
ae.ErrorMessage(), ae.ErrorFault().String())
    }
    return
}
```

Recuperación de identificadores de solicitud

Al trabajar con AWS Support, es posible que se le pida que proporcione el identificador de la solicitud cuyos errores intenta solucionar. Puede usar [http.ResponseError](#) y el método `ServiceRequestID()` para recuperar el identificador de solicitud asociado a la respuesta de error.

```
import "log"
import awshttp "github.com/aws/aws-sdk-go-v2/aws/transport/http"

// ...

if err != nil {
```

```
var re *awshttp.ResponseError
if errors.As(err, &re) {
    log.Printf("requestID: %s, error: %v", re.ServiceRequestID(), re.Unwrap());
}
return
}
```

Identificadores de solicitud de Amazon S3

Las solicitudes de Amazon S3 contienen identificadores adicionales que se pueden utilizar para ayudar a AWS Support a solucionar los errores de su solicitud. Puede usar [s3.ResponseError](#) y llamar a `ServiceRequestID()` y `ServiceHostID()` para recuperar el ID de solicitud y el de host.

```
import "log"
import "github.com/aws/aws-sdk-go-v2/service/s3"

// ...

if err != nil {
    var re s3.ResponseError
    if errors.As(err, &re) {
        log.Printf("requestID: %s, hostID: %s request failure", re.ServiceRequestID(),
re.ServiceHostID());
    }
    return
}
```

Uso de AWS SDK para Go v2 con los servicios de AWS

Para realizar llamadas a un servicio de AWS, primero debe crear una instancia del cliente de servicio. Un cliente de servicio proporciona acceso de bajo nivel a todas las acciones de la API para ese servicio. Por ejemplo, se crea un cliente de servicio de Amazon S3 para realizar llamadas a las API de Amazon S3.

Cuando llama a operaciones de servicio, transfiere los parámetros de entrada como estructura. Una llamada correcta dará como resultado una estructura de salida que contiene la respuesta de la API de servicio. Por ejemplo, después de haber llamado correctamente a una acción de creación de buckets de Amazon S3, la acción devuelve una estructura de salida con la ubicación del bucket.

Para ver la lista de clientes de servicio, incluidos sus métodos y parámetros, consulte la [referencia de la API de AWS SDK para Go](#).

Protección de la integridad de los datos con sumas de comprobación

Amazon Simple Storage Service (Amazon S3) permite especificar una suma de comprobación al cargar un objeto. Cuando se especifica una suma de comprobación, esta se almacena con el objeto y se puede validar cuando se descarga el objeto.

Las sumas de comprobación proporcionan un nivel adicional de integridad de los datos al transferir archivos. Con las sumas de comprobación, puede comprobar la coherencia de datos verificando que el archivo recibido coincide con el archivo original. Para obtener más información sobre las sumas de comprobación con Amazon S3, consulte la [Guía del usuario de Amazon Simple Storage Service](#), que incluye los [algoritmos compatibles](#).

Puede elegir el algoritmo que mejor se adapte a sus necesidades y dejar que el SDK calcule la suma de comprobación. También puede especificar un valor de suma de comprobación precalculado mediante uno de los algoritmos compatibles.

Note

A partir de la [versión 1.74.1 del módulo de Amazon S3](#), el SDK ofrece protecciones de integridad predeterminadas mediante el cálculo automático de una suma de comprobación de CRC32 para las cargas. El SDK calcula esta suma de comprobación si no se proporciona

un valor de suma de comprobación precalculado o si no se especifica un algoritmo que el SDK deba usar para calcular una suma de comprobación.

El SDK también proporciona una configuración global para las protecciones de integridad de datos que puede establecer de forma externa y sobre la que puede obtener más información en la [Guía de referencia de los SDK y las herramientas de AWS](#).

Analizaremos las sumas de comprobación en dos fases de solicitud: carga del objeto y descarga del objeto.

Cargar un objeto

Al cargar un objeto con el método `putObject` y proporcionar un algoritmo de suma de comprobación, el SDK calcula la suma de comprobación del algoritmo especificado.

El siguiente fragmento de código muestra una solicitud para cargar un objeto con una suma de comprobación de CRC32. Cuando el SDK envía la solicitud, calcula la suma de comprobación de CRC32 y carga el objeto. Amazon S3 valida la integridad del contenido mediante el cálculo de la suma de comprobación y su comparación con la suma de comprobación proporcionada por el SDK. A continuación, Amazon S3 almacena la suma de comprobación con el objeto.

```
out, err := s3Client.PutObject(context.Background(), &s3.PutObjectInput{
    Bucket:      aws.String("bucket"),
    Key:         aws.String("key"),
    ChecksumAlgorithm: types.ChecksumAlgorithmCrc32,
    Body:        strings.NewReader("Hello World"),
})
```

Si no proporciona un algoritmo de suma de comprobación con la solicitud, el comportamiento de la suma de comprobación varía en función de la versión del SDK que utilice, tal y como se muestra en la siguiente tabla.

Comportamiento de la suma de comprobación cuando no se proporciona ningún algoritmo de suma de comprobación

Versión del módulo de Amazon S3 de AWS SDK para Go	Comportamiento de la suma de comprobación
Antes de v1.74.1	El SDK no calcula automáticamente una suma de comprobación basada en redundancia cíclica (CRC) y la proporciona en la solicitud.
v 1.74.1 o posterior	El SDK usa el algoritmo CRC32 para calcular la suma de comprobación y la proporciona en la solicitud. Amazon S3 valida la integridad de la transferencia mediante el cálculo de la suma de comprobación CRC32 propia y la compara con la suma de comprobación proporcionada por el SDK. Si las sumas de comprobación coinciden, la suma de comprobación se guarda con el objeto.

Utilizar un valor de suma de comprobación calculado previamente

Un valor de suma de comprobación precalculado proporcionado con la solicitud desactiva el cálculo automático por parte del SDK y utiliza el valor proporcionado en su lugar.

En el siguiente ejemplo se muestra una solicitud con una suma de comprobación SHA256 precalculada.

```
out, err := s3Client.PutObject(context.Background(), &s3.PutObjectInput{
    Bucket:      aws.String("bucket"),
    Key:         aws.String("key"),
    ChecksumCRC32: aws.String("checksumvalue"),
    Body:        strings.NewReader("Hello World"),
})
```

Si Amazon S3 determina que el valor de la suma de comprobación es incorrecto para el algoritmo especificado, el servicio devuelve una respuesta de error.

Cargas multiparte

También puede utilizar sumas de comprobación en las cargas multiparte.

AWS SDK para Go proporciona dos opciones para usar sumas de comprobación en cargas multiparte. En la primera opción se utiliza el administrador de transferencias que especifica el algoritmo de CRC32 para la carga.

```
s3Client := s3.NewFromConfig(cfg)
transferManager := manager.NewUploader(s3Client)
out, err := transferManager.Upload(context.Background(), &s3.PutObjectInput{
    Bucket:      aws.String("bucket"),
    Key:         aws.String("key"),
    Body:        large file to trigger multipart upload,
    ChecksumAlgorithm: types.ChecksumAlgorithmCrc32,
})
```

Si no se proporciona un algoritmo de suma de comprobación al utilizar el administrador de transferencias para cargas, el SDK calcula automáticamente una suma de comprobación basada en el algoritmo de CRC32. El SDK realiza este cálculo para todas sus versiones.

La segunda opción usa el cliente de [Amazon S3](#) para llevar a cabo la carga multiparte. Si especifica una suma de comprobación con este método, debe especificar el algoritmo que se utilizará al iniciar la carga. También debe especificar el algoritmo para cada solicitud de parte y proporcionar la suma de comprobación calculada para cada parte una vez cargada.

```
s3Client := s3.NewFromConfig(cfg)
createMultipartUploadOutput, err :=
s3Client.CreateMultipartUpload(context.Background(), &s3.CreateMultipartUploadInput{
    Bucket:      aws.String("bucket"),
    Key:         aws.String("key"),
    ChecksumAlgorithm: types.ChecksumAlgorithmCrc32,
})
if err != nil {
    log.Fatal("err create multipart upload ", err)
}

var partsBody []io.Reader // this is just an example parts content, you should
load your target file in your code
partNum := int32(1)
var completedParts []types.CompletedPart
for _, body := range partsBody {
    uploadPartOutput, err := s3Client.UploadPart(context.Background(),
&s3.UploadPartInput{
        Bucket:      aws.String("bucket"),
        Key:         aws.String("key"),
```

```

        ChecksumAlgorithm: types.ChecksumAlgorithmCrc32,
        Body:                body,
        PartNumber:          aws.Int32(partNum),
        UploadId:            createMultipartUploadOutput.UploadId,
    })
    if err != nil {
        log.Fatal("err upload part ", err)
    }

    completedParts = append(completedParts, types.CompletedPart{
        PartNumber:    aws.Int32(partNum),
        ETag:          uploadPartOutput.ETag,
        ChecksumCRC32: uploadPartOutput.ChecksumCRC32,
    })
    partNum++
}

completeMultipartUploadOutput, err :=
s3Client.CompleteMultipartUpload(context.Background(),
&s3.CompleteMultipartUploadInput{
    Bucket:    aws.String("bucket"),
    Key:       aws.String("key"),
    UploadId:  createMultipartUploadOutput.UploadId,
    MultipartUpload: &types.CompletedMultipartUpload{
        Parts: completedParts,
    },
})
if err != nil {
    log.Fatal("err complete multipart upload ", err)
}

```

Descargar un objeto

Cuando se utiliza el método [GetObject](#) para descargar un objeto, el SDK valida automáticamente la suma de comprobación cuando el campo `ChecksumMode` de `GetObjectInput` se establece en `types.ChecksumModeEnabled`.

La solicitud del siguiente fragmento indica al SDK que valide la suma de comprobación de la respuesta calculándola y comparando los valores.

```

out, err := s3Client.GetObject(context.Background(), &s3.GetObjectInput{
    Bucket:    aws.String("bucket"),
    Key:       aws.String("key"),

```

```
ChecksumMode: types.ChecksumModeEnabled,  
}))
```

Si el objeto no se cargó con una suma de comprobación, no se realizará ninguna validación.

Uso de las utilidades de AWS SDK para Go

AWS SDK para Go incluye las siguientes utilidades para ayudarle a usar los servicios de AWS de forma más fácil. Encuentre las utilidades del SDK en el paquete de servicios de AWS relacionado.

Utilidades de Amazon RDS

IAM Authentication

El paquete [auth](#) proporciona utilidades para generar tokens de autenticación a fin de conectarse a instancias de bases de datos MySQL y PostgreSQL de Amazon RDS. Mediante el método [BuildAuthToken](#), se genera un token de autorización de base de datos al proporcionar el punto de conexión de la base de datos, la región de AWS, el nombre de usuario y una implementación de [aws.CredentialProvider](#) que devuelve credenciales de IAM con permiso para conectarse a la base de datos mediante la autenticación de base de datos de IAM. Para obtener más información sobre la configuración de Amazon RDS con la autenticación de IAM, consulte los siguientes recursos de la Guía para desarrolladores de Amazon RDS:

- [Activación y desactivación de la autenticación de bases de datos de IAM](#)
- [Creación y uso de una política de IAM para el acceso a bases de datos de IAM](#)
- [Creación de cuentas de base de datos utilizando autenticación de IAM](#)

En el ejemplo siguiente se muestra cómo generar un token de autenticación para conectarse a una base de datos de Amazon RDS:

```
import "context"
import "github.com/aws/aws-sdk-go-v2/config"
import "github.com/aws/aws-sdk-go-v2/feature/rds/auth"

// ...

cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    panic("configuration error: " + err.Error())
}

authenticationToken, err := auth.BuildAuthToken(
```

```
context.TODO(),
    "mydb.123456789012.us-east-1.rds.amazonaws.com:3306", // Database Endpoint (With
    Port)
    "us-east-1", // AWS Region
    "jane_doe", // Database Account
    cfg.Credentials,
)
if err != nil {
    panic("failed to create authentication token: " + err.Error())
}
```

Utilidades de Amazon CloudFront

Firmante de URL de Amazon CloudFront

El firmante de URL de Amazon CloudFront simplifica el proceso de creación de direcciones URL firmadas. Una dirección URL firmada incluye información, como la fecha y hora de caducidad, que permite controlar el acceso a su contenido. Las URL firmadas son útiles cuando desea distribuir contenido a través de Internet, pero quiere restringir el acceso a algunos usuarios (por ejemplo, a aquellos que han pagado una tarifa).

Para firmar una URL, cree una instancia de `URLSigner` con el ID de par de claves de CloudFront y la clave privada asociada. A continuación, llame al método `Sign` o `SignWithPolicy` e incluya la URL que se va a firmar. Para obtener más información sobre los pares de claves de Amazon CloudFront, consulte el apartado sobre la [creación de pares de claves de CloudFront para los firmantes de confianza](#) de la Guía para desarrolladores de CloudFront.

En el ejemplo siguiente, se crea una URL firmada que es válida durante una hora tras su creación.

```
import "github.com/aws/aws-sdk-go-v2/feature/cloudfront/sign"

// ...

signer := sign.NewURLSigner(keyID, privKey)

signedURL, err := signer.Sign(rawURL, time.Now().Add(1*time.Hour))
if err != nil {
    log.Fatalf("Failed to sign url, err: %s\n", err.Error())
    return
}
```

Para obtener más información sobre la utilidad de firma, consulte el paquete [sign](#) de la referencia de la API de AWS SDK para Go.

Servicio de metadatos de instancias de Amazon EC2

Puede utilizar AWS SDK para Go para acceder al [servicio de metadatos de instancias de Amazon EC2](#). El paquete Go [feature/ec2/imds](#) proporciona un tipo [Client](#) que se puede utilizar para acceder al servicio de metadatos de instancias de Amazon EC2. Tanto `Client` como sus operaciones asociadas se pueden utilizar de forma similar a los demás clientes del servicio de AWS que proporciona el SDK. Para obtener más información sobre cómo configurar el SDK y utilizar los clientes de servicio, consulte [Configurar el SDK](#) y [Uso de AWS SDK para Go v2 con los servicios de AWS](#).

El cliente puede ayudarle a recuperar fácilmente información sobre las instancias en las que se ejecutan sus aplicaciones, como su región de AWS o dirección IP local. Por lo general, debe crear y enviar solicitudes HTTP para recuperar metadatos de instancias. En su lugar, cree un `imds.Client` para acceder al servicio de metadatos de instancias de Amazon EC2 mediante un cliente programático, como otros servicios de AWS.

Por ejemplo, para crear un cliente:

```
import "context"
import "github.com/aws/aws-sdk-go-v2/config"
import "github.com/aws/aws-sdk-go-v2/feature/ec2/imds"

// ...

cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    log.Printf("error: %v", err)
    return
}

client := imds.NewFromConfig(cfg)
```

A continuación, utilice el cliente del servicio para recuperar información de una categoría de metadatos como `local-ipv4` (la dirección IP privada de la instancia).

```
localIp, err := client.GetMetadata(context.TODO(), &imds.GetMetadataInput{
```



```
    Path: "local-ipv4",
  })
  if err != nil {
    log.Printf("Unable to retrieve the private IP address from the EC2 instance: %s\n",
      err)
    return
  }
  content, _ := io.ReadAll(localIp.Content)
  fmt.Printf("local-ip: %v\n", string(content))
}
```

Para obtener una lista de todas las categorías de metadatos, consulte [Categorías de metadatos de instancia](#) de la Guía del usuario de Amazon EC2.

Utilidades de Amazon S3

Administradores de transferencias de Amazon S3

Los administradores de carga y descarga de Amazon S3 pueden dividir objetos grandes para transferirlos en varias partes, en paralelo. Esto facilita la reanudación de transferencias interrumpidas.

Administrador de cargas de Amazon S3

El administrador de cargas de Amazon S3 determina si un archivo puede dividirse en partes más pequeñas y cargarse en paralelo. Puede personalizar el número de cargas paralelas y el tamaño de las partes que se cargan.

En el ejemplo siguiente se utiliza el elemento `Uploader` de Amazon S3 para cargar un archivo: El uso de `Uploader` es similar a la operación `s3.PutObject()`.

```
import "context"
import "github.com/aws/aws-sdk-go-v2/config"
import "github.com/aws/aws-sdk-go-v2/service/s3"
import "github.com/aws/aws-sdk-go-v2/feature/s3/manager"

// ...

cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
  log.Printf("error: %v", err)
}
```

```
    return
}

client := s3.NewFromConfig(cfg)

uploader := manager.NewUploader(client)
result, err := uploader.Upload(context.TODO(), &s3.PutObjectInput{
    Bucket: aws.String("amzn-s3-demo-bucket"),
    Key:    aws.String("my-object-key"),
    Body:   uploadFile,
})
```

Opciones de configuración

Al crear una `Uploader` instancia mediante [NewUploader](#), puede especificar varias opciones de configuración para personalizar la forma en que se cargan los objetos. Las opciones se reemplazan al proporcionar uno o varios argumentos a `NewUploader`. Entre estas opciones se incluyen las siguientes:

- `PartSize`: especifica el tamaño del búfer, en bytes, de cada una de las partes que se van a cargar. El tamaño mínimo por parte es de 5 MiB.
- `Concurrency`: especifica el número de partes que se van a cargar en paralelo.
- `LeavePartsOnError`: indica si las partes cargadas correctamente se deben dejar en Amazon S3.

El valor `Concurrency` limita el número simultáneo de cargas de partes que pueden producirse durante una llamada a `Upload` determinada. No se trata de un límite de simultaneidad de clientes global. Ajuste los valores de configuración `PartSize` y `Concurrency` para encontrar la configuración óptima. Por ejemplo, los sistemas con conexiones de gran ancho de banda pueden enviar partes más grandes y más cargas en paralelo.

Por ejemplo, la aplicación configura `Uploader` con un ajuste de `Concurrency` establecido en 5. Si, a continuación, la aplicación llama a `Upload` desde dos rutinas de Go diferentes, el resultado son 10 cargas de partes simultáneas (2 rutinas de Go por 5 `Concurrency`).

Warning

Se espera que su aplicación limite las llamadas simultáneas a `Upload` para evitar el agotamiento de los recursos de la aplicación.

A continuación, se muestra un ejemplo para establecer el tamaño predeterminado de las partes durante la creación de `Uploader`:

```
uploader := manager.NewUploader(client, func(u *Uploader) {
    u.PartSize = 10 * 1024 * 1024, // 10 MiB
})
```

Para obtener más información `Uploader` y sus configuraciones, consulta [Uploader](#) en la referencia de la AWS SDK para Go API.

PutObjectInput Body Field (io. Reader contra io.Reader)

El campo `Body` de la estructura `s3.PutObjectInput` es un tipo `io.Reader`. Sin embargo, este campo se puede rellenar con un tipo que cumpla tanto con la interfaz `io.ReadSeeker` como con `io.ReaderAt` para mejorar el uso de los recursos de las aplicaciones en el entorno de host. En el ejemplo siguiente, se crea el tipo `ReadSeekerAt` que cumple con ambas interfaces:

```
type ReadSeekerAt interface {
    io.ReadSeeker
    io.ReaderAt
}
```

En el caso de los tipos `io.Reader`, los bytes del lector deben almacenarse en la memoria del búfer antes de que pueda cargarse la parte. Al aumentar el valor `PartSize` o `Concurrency`, la memoria (RAM) necesaria para el elemento `Uploader` aumenta considerablemente. La memoria necesaria es de aproximadamente *PartSize* por *Concurrency*. Por ejemplo, si se especifican 100 MB para `PartSize` y un valor de diez para `Concurrency`, se requiere al menos 1 GB.

Como un tipo `io.Reader` no puede determinar su tamaño antes de leer sus bytes, `Uploader` no puede calcular cuántas partes se cargarán. En consecuencia, `Uploader` puede alcanzar el límite de carga de Amazon S3 de 10 000 partes para archivos de gran tamaño si establece el valor `PartSize` demasiado bajo. Si intenta cargar más de 10 000 partes, la carga se detendrá y devolverá un error.

En el caso de los valores `body` que implementan el tipo `ReadSeekerAt`, `Uploader` no almacena en la memoria del búfer el contenido del cuerpo antes de enviarlo a Amazon S3. `Uploader` calcula el número de partes previsto antes de cargar el archivo en Amazon S3. Si el valor actual de `PartSize` requiere más de 10 000 partes para cargar el archivo, `Uploader` aumenta el valor de tamaño de las partes de modo que se requieran menos.

Gestión de cargas con errores

Si se produce un error de carga en Amazon S3, de forma predeterminada, Uploader utiliza la operación `AbortMultipartUpload` de Amazon S3 para eliminar las partes cargadas. Esta funcionalidad garantiza que las cargas con errores no consuman el almacenamiento de Amazon S3.

Puede establecer `LeavePartsOnError` en `true` para que Uploader no elimine las partes que se hayan cargado correctamente. Esto resulta útil para reanudar las cargas completadas parcialmente. Para poder actuar en las partes ya cargadas, debe obtener el valor `UploadID` de la carga con errores. En el ejemplo siguiente se muestra cómo utilizar el tipo de interfaz de error `manager.MultiUploadFailure` para obtener el valor `UploadID`.

```
result, err := uploader.Upload(context.TODO(), &s3.PutObjectInput{
    Bucket: aws.String("amzn-s3-demo-bucket"),
    Key:    aws.String("my-object-key"),
    Body:   uploadFile,
})
output, err := u.upload(input)
if err != nil {
    var mu manager.MultiUploadFailure
    if errors.As(err, &mu) {
        // Process error and its associated uploadID
        fmt.Println("Error:", mu)
        _ = mu.UploadID() // retrieve the associated UploadID
    } else {
        // Process error generically
        fmt.Println("Error:", err.Error())
    }
    return
}
```

Reemplazo de opciones de cargador por carga

Puede reemplazar las opciones de Uploader al llamar a `Upload`. Para ello, proporcione uno o varios argumentos al método. Estos reemplazos son modificaciones seguras para el uso simultáneo y no afectan a las cargas en curso ni a las llamadas de `Upload` posteriores al administrador. Por ejemplo, si desea reemplazar la configuración de `PartSize` para una solicitud de carga específica:

```
params := &s3.PutObjectInput{
    Bucket: aws.String("amzn-s3-demo-bucket"),
    Key:    aws.String("my-key"),
    Body:   myBody,
```

```
}  
resp, err := uploader.Upload(context.TODO(), params, func(u *manager.Uploader) {  
    u.PartSize = 10 * 1024 * 1024, // 10 MiB  
})
```

Ejemplos

Carga de una carpeta en Amazon S3

En el ejemplo siguiente se utiliza el paquete `path/filepath` para recopilar de forma recursiva una lista de archivos y cargarlos en el bucket de Amazon S3 especificado. Las claves de los objetos de Amazon S3 llevan como prefijo la ruta relativa del archivo.

```
package main  
  
import (  
    "context"  
    "log"  
    "os"  
    "path/filepath"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/config"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
)  
  
var (  
    localPath string  
    bucket    string  
    prefix    string  
)  
  
func init() {  
    if len(os.Args) != 4 {  
        log.Fatalln("Usage:", os.Args[0], "<local path> <bucket> <prefix>")  
    }  
    localPath = os.Args[1]  
    bucket = os.Args[2]  
    prefix = os.Args[3]  
}  
  
func main() {
```

```

walker := make(fileWalk)
go func() {
    // Gather the files to upload by walking the path recursively
    if err := filepath.Walk(localPath, walker.Walk); err != nil {
        log.Fatalln("Walk failed:", err)
    }
    close(walker)
}()

cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    log.Fatalln("error:", err)
}

// For each file found walking, upload it to Amazon S3
uploader := manager.NewUploader(s3.NewFromConfig(cfg))
for path := range walker {
    rel, err := filepath.Rel(localPath, path)
    if err != nil {
        log.Fatalln("Unable to get relative path:", path, err)
    }
    file, err := os.Open(path)
    if err != nil {
        log.Println("Failed opening file", path, err)
        continue
    }
    defer file.Close()
    result, err := uploader.Upload(context.TODO(), &s3.PutObjectInput{
        Bucket: &bucket,
        Key:     aws.String(filepath.Join(prefix, rel)),
        Body:    file,
    })
    if err != nil {
        log.Fatalln("Failed to upload", path, err)
    }
    log.Println("Uploaded", path, result.Location)
}

}

type fileWalk chan string

func (f fileWalk) Walk(path string, info os.FileInfo, err error) error {
    if err != nil {
        return err
    }

```

```

    }
    if !info.IsDir() {
        f <- path
    }
    return nil
}

```

Administrador de descargas

El administrador [Downloader](#) de Amazon S3 determina si un archivo puede dividirse en partes más pequeñas y descargarse en paralelo. Puede personalizar el número de descargas paralelas y el tamaño de las partes descargadas.

Ejemplo: Descarga de un archivo

En el ejemplo siguiente se utiliza Downloader de Amazon S3 para descargar un archivo.

Downloader [El uso es similar al s3. GetObject](#) operación.

```

import "context"
import "github.com/aws/aws-sdk-go-v2/aws"
import "github.com/aws/aws-sdk-go-v2/config"
import "github.com/aws/aws-sdk-go-v2/service/s3"
import "github.com/aws/aws-sdk-go-v2/feature/s3/manager"

// ...

cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    log.Println("error:", err)
    return
}

client := s3.NewFromConfig(cfg)

downloader := manager.NewDownloader(client)
numBytes, err := downloader.Download(context.TODO(), downloadFile, &s3.GetObjectInput{
    Bucket: aws.String("amzn-s3-demo-bucket"),
    Key:    aws.String("my-key"),
})

```

El parámetro `downloadFile` es un tipo `io.WriterAt`. La interfaz `WriterAt` permite a Downloader escribir varias partes del archivo en paralelo.

Opciones de configuración

Al crear una instancia de `Downloader`, puede especificar opciones de configuración para personalizar la forma en que se descargan los objetos:

- `PartSize`: especifica el tamaño del búfer, en bytes, de cada una de las partes que se van a descargar. El tamaño mínimo por parte es de 5 MB.
- `Concurrency`: especifica el número de partes que se van a descargar en paralelo.

El valor `Concurrency` limita el número simultáneo de descargas de partes que pueden producirse durante una llamada a `Download` determinada. No se trata de un límite de simultaneidad de clientes global. Ajuste los valores de configuración `PartSize` y `Concurrency` para encontrar la configuración óptima. Por ejemplo, los sistemas con conexiones de gran ancho de banda pueden recibir partes más grandes y más descargas en paralelo.

Por ejemplo, la aplicación configura `Downloader` con un ajuste de `Concurrency` establecido en 5. Si, a continuación, la aplicación llama a `Download` desde dos rutinas de Go diferentes, el resultado son 10 descargas de partes simultáneas (2 rutinas de Go por 5 `Concurrency`).

Warning

Se espera que su aplicación limite las llamadas simultáneas a `Download` para evitar el agotamiento de los recursos de la aplicación.

Para obtener más información sobre `Downloader` sus demás opciones de configuración, consulte [Manager.Downloader](#) en la referencia de la AWS SDK para Go API.

Reemplazo de opciones de Downloader por descarga

Puede reemplazar las opciones de `Downloader` al llamar a `Download`. Para ello, proporcione uno o varios argumentos funcionales al método. Estos reemplazos son modificaciones seguras para el uso simultáneo y no afectan a las cargas en curso ni a las llamadas de `Download` posteriores al administrador. Por ejemplo, si desea reemplazar la configuración de `PartSize` para una solicitud de carga específica:

```
params := &s3.GetObjectInput{
    Bucket: aws.String("amzn-s3-demo-bucket"),
    Key:    aws.String("my-key"),
}
```



```
resp, err := downloader.Download(context.TODO(), targetWriter, params, func(u
    *manager.Downloader) {
    u.PartSize = 10 * 1024 * 1024, // 10 MiB
})
```

Ejemplos

Descarga de todos los objetos de un bucket

En el ejemplo siguiente se utiliza la paginación para recopilar una lista de objetos desde un bucket de Amazon S3. A continuación, se descarga cada objeto en un archivo local.

```
package main

import (
    "context"
    "fmt"
    "log"
    "os"
    "path/filepath"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
)

var (
    Bucket      = "amzn-s3-demo-bucket" // Download from this bucket
    Prefix      = "logs/"              // Using this key prefix
    LocalDirectory = "s3logs"          // Into this directory
)

func main() {
    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        log.Fatalln("error:", err)
    }

    client := s3.NewFromConfig(cfg)
    manager := manager.NewDownloader(client)

    paginator := s3.NewListObjectsV2Paginator(client, &s3.ListObjectsV2Input{
        Bucket: &Bucket,
```

```

    Prefix: &Prefix,
  })

  for paginator.HasMorePages() {
    page, err := paginator.NextPage(context.TODO())
    if err != nil {
      log.Fatalln("error:", err)
    }
    for _, obj := range page.Contents {
      if err := downloadToFile(manager, LocalDirectory, Bucket,
aws.ToString(obj.Key)); err != nil {
        log.Fatalln("error:", err)
      }
    }
  }
}

func downloadToFile(downloader *manager.Downloader, targetDirectory, bucket, key
string) error {
  // Create the directories in the path
  file := filepath.Join(targetDirectory, key)
  if err := os.MkdirAll(filepath.Dir(file), 0775); err != nil {
    return err
  }

  // Set up the local file
  fd, err := os.Create(file)
  if err != nil {
    return err
  }
  defer fd.Close()

  // Download the file using the AWS SDK for Go
  fmt.Printf("Downloading s3://%s/%s to %s...\n", bucket, key, file)
  _, err = downloader.Download(context.TODO(), fd, &s3.GetObjectInput{Bucket:
&bucket, Key: &key})

  return err
}

```

GetBucketRegion

[GetBucketRegion](#) es una función de utilidad para determinar la ubicación AWS regional de un bucket de Amazon S3. `GetBucketRegion` toma un cliente Amazon S3 y lo usa para determinar la ubicación del bucket solicitado dentro de la AWS partición asociada a la región configurada del cliente.

Por ejemplo, para encontrar la región del bucket *amzn-s3-demo-bucket*:

```
cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    log.Println("error:", err)
    return
}

bucket := "amzn-s3-demo-bucket"
region, err := manager.GetBucketRegion(ctx, s3.NewFromConfig(cfg), bucket)
if err != nil {
    var bnf manager.BucketNotFound
    if errors.As(err, &bnf) {
        log.Printf("unable to find bucket %s's Region\n", bucket)
    } else {
        log.Println("error:", err)
    }
    return
}
fmt.Printf("Bucket %s is in %s region\n", bucket, region)
```

Si `GetBucketRegion` no puede resolver la ubicación de un bucket, la función devuelve un tipo de [BucketNotFound](#) error, como se muestra en el ejemplo.

Entrada de transmisión no buscable

En el caso de operaciones de API como `PutObject` y `UploadPart`, el cliente de Amazon S3 espera que el valor del parámetro de entrada `Body` implemente la interfaz [io.Seeker](#) de forma predeterminada. El cliente utiliza la interfaz `io.Seeker` a fin de determinar la longitud del valor que se va a cargar y calcular el hash de carga útil para la [firma de solicitud](#). Si el valor del parámetro de entrada `Body` no implementa `io.Seeker`, la aplicación recibirá un error.

```
operation error S3: PutObject, failed to compute payload hash: failed to seek
body to start, request stream is not seekable
```

Para cambiar este comportamiento, modifique el [Middleware](#) del método de operación con opciones funcionales. El APIOptions asistente [With](#) devuelve una opción funcional para cero o más mutadores de middleware. [Para impedir que el cliente calcule el hash de la carga útil y utilice la firma de solicitud de carga útil sin firmar, añada la versión v4.SwapComputePayloadSHA256ForUnsignedPayloadMiddleware.](#)

```
resp, err := client.PutObject(context.TODO(), &s3.PutObjectInput{
    Bucket: &bucketName,
    Key: &objectName,
    Body: bytes.NewBuffer([]byte(`example object!`)),
    ContentLength: 15, // length of body
}, s3.WithAPIOptions(
    v4.SwapComputePayloadSHA256ForUnsignedPayloadMiddleware,
))
```

 Warning

Amazon S3 requiere que se proporcione la longitud del contenido para todos los objetos cargados en un bucket. Como el parámetro de entrada Body no implementa la interfaz `io.Seeker`, el cliente no podrá calcular el parámetro ContentLength para la solicitud. La aplicación debe proporcionar el parámetro. Si no se proporciona el parámetro ContentLength se producirá un error en la solicitud.

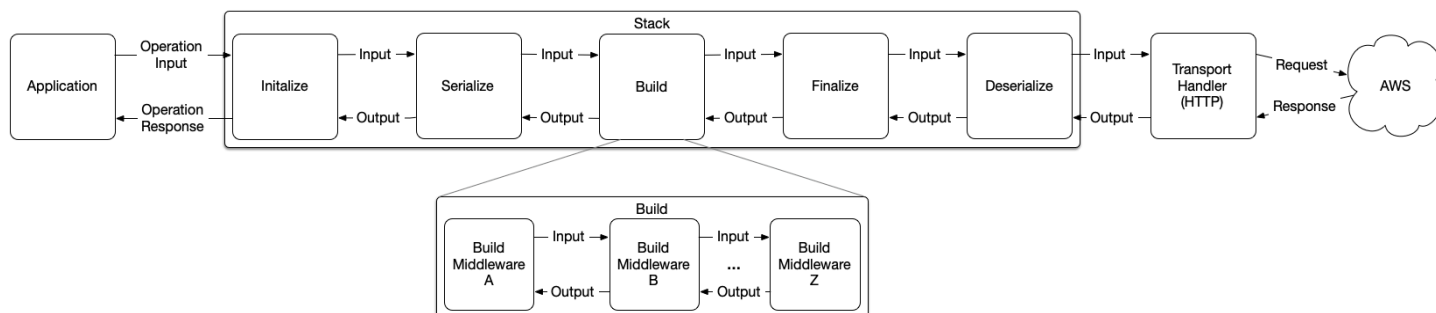
Use los SDK [Administrador de cargas de Amazon S3](#) para aquellas cargas que no se puedan buscar y que no tengan una longitud conocida.

Personalización de las solicitudes de cliente de AWS SDK para Go v2 con middleware

⚠ Warning

La modificación de la canalización de solicitudes de cliente puede generar solicitudes no válidas o con formato incorrecto, o puede provocar errores de aplicación inesperados. Esta funcionalidad está pensada para casos de uso avanzados que la interfaz del SDK no proporciona de forma predeterminada.

Puede personalizar las solicitudes de cliente de AWS SDK para Go mediante el registro de middleware (uno o más) en la [pila](#) de una operación de servicio. La pila se compone de una serie de pasos: Initialize, Serialize, Build, Finalize y Deserialize. El middleware (cero o más) que contiene cada paso opera en los tipos de entrada y salida de ese paso. El diagrama y la tabla siguientes proporcionan información general sobre cómo la solicitud y la respuesta de una operación recorren la pila:



Paso de la pila	Descripción
Initialize	Prepara la entrada y establece los parámetros predeterminados según sea necesario.
Serialize	Serializa la entrada en un formato de protocolo adecuado para la capa de transporte de destino.

Paso de la pila	Descripción
Build	Asocie metadatos adicionales a la entrada serializada, como la longitud del contenido HTTP.
Finalize	Preparación del mensaje final, incluidos los reintentos y la autenticación (firma de SiGv4).
Deserialize	Deserialice las respuestas del formato de protocolo en un error o tipo estructurado.

Cada middleware de un paso determinado debe tener un identificador único, que viene determinado por el método ID del middleware. Los identificadores de middleware garantizan que solo se registre una instancia de un middleware determinado en un paso y permiten insertar otro middleware de paso en relación con este.

Para asociar middleware de paso, use los métodos `Insert` o `Add` de un paso. Use `Add` para asociar middleware al principio de un paso al especificar `middleware.Before` como `RelativePosition`, y `middleware.After` para asociarlo al final del paso. Use `Insert` para asociar middleware a un paso con la inserción del middleware relativo a otro middleware de paso.

Warning

Debe utilizar el método `Add` para insertar middleware de paso personalizado de forma segura. El uso de `Insert` crea una dependencia entre su middleware personalizado y aquel relacionado que inserta. El middleware incluido en un paso de pila debe considerarse opaco para evitar que se produzcan cambios importantes en la aplicación.

Escritura de un middleware personalizado

Cada paso de pila tiene una interfaz que debe cumplirse para poder asociar un middleware a un paso determinado. Puede utilizar una de las funciones `StepMiddlewareFunc` proporcionadas para cumplir rápidamente con esta interfaz. En la tabla siguiente se describen los pasos, su interfaz y la función auxiliar que se puede utilizar para cumplir con la interfaz.

Paso	Interfaz	Función Helper
Initialize	<u>InitializeMiddleware</u>	<u>InitializeMiddlewareFunc</u>
Build	<u>BuildMiddleware</u>	<u>BuildMiddlewareFunc</u>
Serialize	<u>SerializeMiddleware</u>	<u>SerializeMiddlewareFunc</u>
Finalize	<u>FinalizeMiddleware</u>	<u>FinalizeMiddlewareFunc</u>
Deserialize	<u>DeserializeMiddleware</u>	<u>DeserializeMiddlewareFunc</u>

En los ejemplos siguientes se muestra cómo puede escribir un middleware personalizado para rellenar el miembro `Bucket` de las llamadas a la API `GetObject` de Amazon S3 en caso de que no se proporcione uno. Se hará referencia a este middleware en ejemplos posteriores para mostrar cómo asociar middleware de un paso a la pila.

```
import "github.com/aws/smithy-go/aws"
import "github.com/aws/smithy-go/middleware"
import "github.com/aws/aws-sdk-go-v2/service/s3"

// ...

var defaultBucket = middleware.InitializeMiddlewareFunc("DefaultBucket", func(
    ctx context.Context, in middleware.InitializeInput, next
    middleware.InitializeHandler,
) (
    out middleware.InitializeOutput, metadata middleware.Metadata, err error,
) {
    // Type switch to check if the input is s3.GetObjectInput, if so and the bucket is
    // not set, populate it with
    // our default.
    switch v := in.Parameters.(type) {
    case *s3.GetObjectInput:
        if v.Bucket == nil {
            v.Bucket = aws.String("amzn-s3-demo-bucket")
        }
    }
}

// Middleware must call the next middleware to be executed in order to continue
// execution of the stack.
```

```
// If an error occurs, you can return to prevent further execution.
return next.HandleInitialize(ctx, in)
})
```

Cómo asociar middleware a todos los clientes

Para asociar su middleware de paso personalizado a cada uno de los clientes, añada el middleware mediante el miembro `APIOptions` del tipo [aws.Config](#). En los ejemplos siguientes se asocia el middleware `defaultBucket` a todos los clientes creados con el objeto `aws.Config` de sus aplicaciones:

```
import "context"
import "github.com/aws/aws-sdk-go-v2/aws"
import "github.com/aws/aws-sdk-go-v2/config"
import "github.com/aws/aws-sdk-go-v2/service/s3"
import "github.com/aws/smithy-go/middleware"

// ...

cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    // handle error
}

cfg.APIOptions = append(cfg.APIOptions, func(stack *middleware.Stack) error {
    // Attach the custom middleware to the beginning of the Initialize step
    return stack.Initialize.Add(defaultBucket, middleware.Before)
})

client := s3.NewFromConfig(cfg)
```

Cómo asociar middleware a una operación específica

Para asociar su middleware de paso personalizado a una operación de cliente específica, modifique el miembro `APIOptions` del cliente con la lista de argumentos variádicos de una operación. En los ejemplos siguientes se asocia el middleware `defaultBucket` a la invocación de una operación `GetObject` de Amazon S3 específica:

```
import "context"
```



```
import "github.com/aws/aws-sdk-go-v2/aws"
import "github.com/aws/aws-sdk-go-v2/config"
import "github.com/aws/aws-sdk-go-v2/service/s3"
import "github.com/aws/smithy-go/middleware"

// ...

// registerDefaultBucketMiddleware registers the defaultBucket middleware with the
// provided stack.
func registerDefaultBucketMiddleware(stack *middleware.Stack) error {
    // Attach the custom middleware to the beginning of the Initialize step
    return stack.Initialize.Add(defaultBucket, middleware.Before)
}

// ...

cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    // handle error
}

client := s3.NewFromConfig(cfg)

object, err := client.GetObject(context.TODO(), &s3.GetObjectInput{
    Key: aws.String("my-key"),
}, func(options *s3.Options) {
    // Register the defaultBucketMiddleware for this operation only
    options.APIOptions = append(options.APIOptions, registerDefaultBucketMiddleware)
})
```

Transferencia de metadatos descendente en la pila

En determinadas situaciones, puede que necesite utilizar dos o más middlewares a la par mediante el intercambio de información o del estado. Puede usar [context.Context](#) para pasar estos metadatos mediante [middleware.WithStackValue](#). `middleware.WithStackValue` asocia el par clave-valor determinado al contexto proporcionado y limita de forma segura el alcance a la pila en ejecución. Estos valores para la pila se pueden recuperar de un contexto mediante [middleware.GetStackValue](#) y con la clave usada para almacenar el valor correspondiente. Las claves deben ser comparables y debe definir sus propios tipos como claves de contexto para evitar conflictos. En los ejemplos siguientes se muestra cómo dos middleware pueden utilizar `context.Context` para realizar una transferencia de información descendente en la pila.

```
import "context"
import "github.com/aws/smithy-go/middleware"

// ...

type customKey struct {}

func GetCustomKey(ctx context.Context) (v string) {
    v, _ = middleware.GetStackValue(ctx, customKey{}).(string)
    return v
}

func SetCustomKey(ctx context.Context, value string) context.Context {
    return middleware.WithStackValue(ctx, customKey{}, value)
}

// ...

var customInitialize = middleware.InitializeMiddlewareFunc("customInitialize", func(
    ctx context.Context, in middleware.InitializeInput, next
    middleware.InitializeHandler,
) (
    out middleware.InitializeOutput, metadata middleware.Metadata, err error,
) {
    ctx = SetCustomKey(ctx, "my-custom-value")

    return next.HandleInitialize(ctx, in)
})

var customBuild = middleware.BuildMiddlewareFunc("customBuild", func(
    ctx context.Context, in middleware.BuildInput, next middleware.BuildHandler,
) (
    out middleware.BuildOutput, metadata middleware.Metadata, err error,
) {
    customValue := GetCustomKey(ctx)

    // use customValue

    return next.HandleBuild(ctx, in)
})
```

Metadatos proporcionados por el SDK

AWS SDK para Go proporciona varios valores de metadatos que se pueden recuperar del contexto proporcionado. Estos valores pueden utilizarse para habilitar un middleware más dinámico que modifique su comportamiento en función de la región de destino, la operación o el servicio en ejecución. En la tabla siguiente se proporcionan algunas de las claves disponibles:

Clave	Recuperador	Descripción
ServiceID	GetServiceID	Recupera el identificador de servicio para la pila en ejecución. Esto puede compararse con la constante <code>ServiceID</code> del paquete de clientes de servicio.
OperationName	GetOperationName	Recupera el nombre de la operación para la pila en ejecución.
Logger	GetLogger	Recupera el registrador que se puede usar para registrar mensajes del middleware.

Transferencia de metadatos ascendente en la pila

Puede realizar una transferencia de metadatos ascendente en la pila mediante la adición de pares clave-valor de los metadatos con [middleware.Metadata](#). Cada paso de middleware devuelve una estructura de salida, metadatos y un error. Su middleware personalizado debe devolver los metadatos recibidos al llamar al siguiente controlador del paso. Esto garantiza que los metadatos agregados por el middleware descendente se propaguen a la aplicación que invoca la operación de servicio. La aplicación que realiza la invocación puede acceder a los metadatos resultantes mediante la forma de salida de la operación a través del miembro de la estructura `ResultMetadata`.

En los ejemplos siguientes se muestra cómo un middleware personalizado puede agregar metadatos que se devuelven como parte del resultado de la operación.

```

import "context"
import "github.com/aws/aws-sdk-go-v2/service/s3"
import "github.com/aws/smithy-go/middleware"

// ...

type customKey struct{}

func GetCustomKey(metadata middleware.Metadata) (v string) {
    v, _ = metadata.Get(customKey{}).(string)
    return v
}

func SetCustomKey(metadata *middleware.Metadata, value string) {
    metadata.Set(customKey{}, value)
}

// ...

var customInitalize = middleware.InitializeMiddlewareFunc("customInitialize", func (
    ctx context.Context, in middleware.InitializeInput, next
    middleware.InitializeHandler,
) (
    out middleware.InitializeOutput, metadata middleware.Metadata, err error,
) {
    out, metadata, err = next.HandleInitialize(ctx, in)
    if err != nil {
        return out, metadata, err
    }

    SetCustomKey(&metadata, "my-custom-value")

    return out, metadata, nil
})

// ...

client := s3.NewFromConfig(cfg, func (options *s3.Options) {
    options.APIOptions = append(options.APIOptions, func(stack *middleware.Stack) error
    {
        return stack.Initialize.Add(customInitalize, middleware.After)
    })
})

```

```
out, err := client.GetObject(context.TODO(), &s3.GetObjectInput{
    // input parameters
})
if err != nil {
    // handle error
}

customValue := GetCustomKey(out.ResponseMetadata)
```

Preguntas frecuentes

¿Cómo se configura el cliente HTTP de mi SDK? ¿Existen directrices o prácticas recomendadas?

No podemos proporcionar instrucciones a los clientes sobre cómo configurar su flujo de trabajo HTTP de una forma que resulte más eficaz para su carga de trabajo concreta. La respuesta a esto es el producto de una ecuación multivariante, con factores de entrada que incluyen, entre otras cosas, lo siguiente:

- la huella de red de la aplicación (TPS, rendimiento, etc.)
- los servicios que se utilizan
- las características de computación de la implementación
- la naturaleza geográfica de la implementación
- el comportamiento deseado de la aplicación o las necesidades de la propia aplicación (tiemposSLAs, etc.)

¿Cómo se configuran los tiempos de espera de las operaciones?

Al igual que en la pregunta anterior, depende. Entre los elementos que deben tenerse en cuenta a este respecto se incluyen los siguientes:

- Todos los factores anteriores relacionados con la configuración del cliente HTTP
- Las restricciones de tiempo o SLA de la propia aplicación (por ejemplo, si usted mismo envía el tráfico a otros consumidores)

La respuesta a esta pregunta casi NUNCA debe basarse en una observación empírica pura de un comportamiento previo (por ejemplo: “He realizado mil llamadas a esta operación en cinco segundos como máximo, de modo que tendré esto en cuenta para establecer el tiempo de espera con un factor de seguridad del doble, es decir, diez segundos”). Las condiciones del entorno pueden cambiar, los servicios pueden degradarse temporalmente y estos tipos de suposiciones pueden dejar de ser válidas en cualquier momento.

Las solicitudes que realiza el SDK agotan el tiempo de espera o tardan demasiado, ¿cómo lo soluciono?

No podemos ayudar con las llamadas a operaciones que se prolongan o cuyo tiempo de espera se ha agotado debido a un tiempo de transmisión prolongado. En el SDK, el “tiempo de transmisión” se define de cualquiera de las siguientes formas:

- Tiempo dedicado al método `HTTPClient.Do()` del cliente del SDK
- Tiempo dedicado a las funciones `Read()` en un cuerpo de respuesta HTTP que se ha reenviado al intermediario (por ejemplo, `GetObject`)

Si tiene problemas debido a la latencia o los tiempos de espera de las operaciones, lo primero que debe hacer es obtener la telemetría del ciclo de vida de la operación del SDK para determinar el desglose temporal entre el tiempo de transmisión y la sobrecarga asociada a la operación. Consulte la sección de la guía [Tiempos de las operaciones del SDK](#), que contiene un fragmento de código reutilizable que puede realizar esta tarea.

¿Cómo puedo solucionar un error **read: connection reset**?

El SDK reintentará cualquier error que coincida con el patrón `connection reset` de forma predeterminada. Esto cubrirá la gestión de errores en la mayoría de las operaciones, en las que la respuesta HTTP de la operación se consume por completo y se deserializa en su tipo de resultado modelado.

Sin embargo, este error puede seguir produciéndose en un contexto externo al bucle de reintentos: determinadas operaciones del servicio reenvían directamente el cuerpo de respuesta HTTP de la API al intermediario para que se consuma de forma directa desde la red a través de `io.ReadCloser` (por ejemplo, carga útil de objeto de `GetObject`). Este error puede producirse al realizar una operación `Read` en el cuerpo de respuesta.

El error indica que el host, el servicio o cualquier intermediario (por ejemplo, puertas de enlace de NAT, proxies, equilibradores de carga) cerró la conexión mientras se intentaba leer la respuesta.

Esto se puede producir por varios motivos:

- No ha consumido el cuerpo de la respuesta durante algún tiempo tras la recepción de la respuesta en sí (después de la llamada a la operación de servicio). Recomendamos que consuma el cuerpo de respuesta HTTP lo antes posible para estos tipos de operaciones.

- No ha cerrado un cuerpo de respuesta recibido anteriormente. Esto puede hacer que se restablezca la conexión en determinadas plataformas. DEBE cerrar cualquier instancia de **io.ReadCloser** proporcionada en la respuesta de una operación, independientemente de si consume su contenido.

Aparte de esto, pruebe a ejecutar `tcpdump` para una conexión afectada en la periferia de su red (por ejemplo, más allá de cualquier proxy bajo su control). Si ve que el AWS punto final parece estar enviando un RST TCP, debería utilizar la consola de AWS soporte para iniciar una demanda contra el servicio infractor. Prepárate para proporcionar una solicitud IDs y marcas horarias específicas sobre cuándo ocurrió el problema.

¿Por qué recibo errores de “firma no válida” al usar un proxy HTTP con el SDK?

El algoritmo de firma de los AWS servicios (generalmente sigv4) está vinculado a los encabezados de la solicitud serializada, más específicamente a la mayoría de los encabezados con el prefijo. X- Los proxies tienden a modificar la solicitud saliente añadiendo información de reenvío adicional (a menudo con un encabezado X-Forwarded-For), lo que invalida la eficacia de la firma calculada por el SDK.

Si utiliza un proxy HTTP y experimenta errores de firma, debe tratar de capturar la solicitud tal y como aparece al salir del proxy y determinar si es diferente.

Tiempos de las operaciones del SDK

Al depurar problemas de tiempo de espera o latencia en el SDK, es fundamental identificar los componentes del ciclo de vida de la operación que tardan en ejecutarse más de lo esperado. Como punto de partida, normalmente tendrá que inspeccionar el desglose de tiempo entre la llamada a operación general y la propia llamada HTTP.

El programa de ejemplo siguiente implementa una sonda de instrumentación básica en términos del middleware `smithy-go` para clientes de SQS y demuestra cómo se usa. La sonda emite la siguiente información para cada llamada a operación:

- ID de solicitud de AWS
- ID de servicio
- Nombre de operación

- Tiempo de invocación de la operación
- Hora de llamada http

Cada mensaje emitido tiene como prefijo un “ID de invocación” único (para una sola operación) que se establece al principio de la pila de controladores.

El punto de entrada para la instrumentación se expone como `WithOperationTiming`, que está parametrizado para aceptar una función de gestión de mensajes que recibirá “eventos” de instrumentación como cadenas formateadas. `PrintfMSGHandler`, que simplemente volcará mensajes en `stdout`, se proporciona para mayor comodidad.

El servicio utilizado aquí es intercambiable: TODAS las opciones del cliente de servicio aceptan `APIOptions` y `HTTPClient` como configuración. Por ejemplo, `WithOperationTiming` podría declararse en su lugar como:

```
func WithOperationTiming(msgHandler func(string)) func(*s3.Options)
func WithOperationTiming(msgHandler func(string)) func(*dynamodb.Options)
// etc.
```

Si lo cambia, asegúrese de cambiar también la firma de la función que devuelve.

```
import (
    "context"
    "fmt"
    "log"
    "net/http"
    "sync"
    "time"

    awsmiddleware "github.com/aws/aws-sdk-go-v2/aws/middleware"
    awshttp "github.com/aws/aws-sdk-go-v2/aws/transport/http"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
    "github.com/aws/smithy-go/middleware"
    smithyrand "github.com/aws/smithy-go/rand"
)

// WithOperationTiming instruments an SQS client to dump timing information for
// the following spans:
//   - overall operation time
//   - HTTPClient call time
```

```
//
// This instrumentation will also emit the request ID, service name, and
// operation name for each invocation.
//
// Accepts a message "handler" which is invoked with formatted messages to be
// handled externally, you can use the declared PrintfMSGHandler to simply dump
// these values to stdout.
func WithOperationTiming(msgHandler func(string)) func(*sqs.Options) {
    return func(o *sqs.Options) {
        o.APIOptions = append(o.APIOptions, addTimingMiddlewares(msgHandler))
        o.HTTPClient = &timedHTTPClient{
            client:    awshttp.NewBuildableClient(),
            msgHandler: msgHandler,
        }
    }
}

// PrintfMSGHandler writes messages to stdout.
func PrintfMSGHandler(msg string) {
    fmt.Printf("%s\n", msg)
}

type invokeIDKey struct{}

func setInvokeID(ctx context.Context, id string) context.Context {
    return middleware.WithStackValue(ctx, invokeIDKey{}, id)
}

func getInvokeID(ctx context.Context) string {
    id, _ := middleware.GetStackValue(ctx, invokeIDKey{}).(string)
    return id
}

// Records the current time, and returns a function to be called when the
// target span of events is completed. The return function will emit the given
// span name and time elapsed to the given message consumer.
func timeSpan(ctx context.Context, name string, consumer func(string)) func() {
    start := time.Now()
    return func() {
        elapsed := time.Now().Sub(start)
        consumer(fmt.Sprintf("[%s] %s: %s", getInvokeID(ctx), name, elapsed))
    }
}
}
```

```

type timedHTTPClient struct {
    client      *awshttp.BuildableClient
    msgHandler func(string)
}

func (c *timedHTTPClient) Do(r *http.Request) (*http.Response, error) {
    defer timeSpan(r.Context(), "http", c.msgHandler)()

    resp, err := c.client.Do(r)
    if err != nil {
        return nil, fmt.Errorf("inner client do: %v", err)
    }

    return resp, nil
}

type addInvokeIDMiddleware struct {
    msgHandler func(string)
}

func (*addInvokeIDMiddleware) ID() string { return "addInvokeID" }

func (*addInvokeIDMiddleware) HandleInitialize(ctx context.Context, in
    middleware.InitializeInput, next middleware.InitializeHandler) (
    out middleware.InitializeOutput, md middleware.Metadata, err error,
) {
    id, err := smithyrand.NewUUID(smithyrand.Reader).GetUUID()
    if err != nil {
        return out, md, fmt.Errorf("new uuid: %v", err)
    }

    return next.HandleInitialize(setInvokeID(ctx, id), in)
}

type timeOperationMiddleware struct {
    msgHandler func(string)
}

func (*timeOperationMiddleware) ID() string { return "timeOperation" }

func (m *timeOperationMiddleware) HandleInitialize(ctx context.Context, in
    middleware.InitializeInput, next middleware.InitializeHandler) (
    middleware.InitializeOutput, middleware.Metadata, error,
) {

```

```

    defer timeSpan(ctx, "operation", m.msgHandler)()
    return next.HandleInitialize(ctx, in)
}

type emitMetadataMiddleware struct {
    msgHandler func(string)
}

func (*emitMetadataMiddleware) ID() string { return "emitMetadata" }

func (m *emitMetadataMiddleware) HandleInitialize(ctx context.Context, in
    middleware.InitializeInput, next middleware.InitializeHandler) (
    middleware.InitializeOutput, middleware.Metadata, error,
) {
    out, md, err := next.HandleInitialize(ctx, in)

    invokeID := getInvokeID(ctx)
    requestID, _ := awsmiddleware.GetRequestIDMetadata(md)
    service := awsmiddleware.GetServiceID(ctx)
    operation := awsmiddleware.GetOperationName(ctx)
    m.msgHandler(fmt.Sprintf("[%s] requestID = %s", invokeID, requestID))
    m.msgHandler(fmt.Sprintf("[%s] service = %s", invokeID, service))
    m.msgHandler(fmt.Sprintf("[%s] operation = %s", invokeID, operation))

    return out, md, err
}

func addTimingMiddlewares(mh func(string)) func(*middleware.Stack) error {
    return func(s *middleware.Stack) error {
        if err := s.Initialize.Add(&timeOperationMiddleware{msgHandler: mh},
            middleware.Before); err != nil {
            return fmt.Errorf("add time operation middleware: %v", err)
        }
        if err := s.Initialize.Add(&addInvokeIDMiddleware{msgHandler: mh},
            middleware.Before); err != nil {
            return fmt.Errorf("add invoke id middleware: %v", err)
        }
        if err := s.Initialize.Insert(&emitMetadataMiddleware{msgHandler: mh},
            "RegisterServiceMetadata", middleware.After); err != nil {
            return fmt.Errorf("add emit metadata middleware: %v", err)
        }
        return nil
    }
}

```

```

func main() {
    cfg, err := config.LoadDefaultConfig(context.Background())
    if err != nil {
        log.Fatal(fmt.Errorf("load default config: %v", err))
    }

    svc := sqs.NewFromConfig(cfg, WithOperationTiming(PrintfMSGHandler))

    var wg sync.WaitGroup

    for i := 0; i < 6; i++ {
        wg.Add(1)
        go func() {
            defer wg.Done()

            _, err = svc.ListQueues(context.Background(), nil)
            if err != nil {
                fmt.Println(fmt.Errorf("list queues: %v", err))
            }
        }()
    }
    wg.Wait()
}

```

A continuación se muestra un ejemplo de salida de este programa:

```

[e9a801bb-c51d-45c8-8e9f-a202e263fde8] http: 192.24067ms
[e9a801bb-c51d-45c8-8e9f-a202e263fde8] requestID = "dbee3082-96a3-5b23-
adca-6d005696fa94"
[e9a801bb-c51d-45c8-8e9f-a202e263fde8] service    = "SQS"
[e9a801bb-c51d-45c8-8e9f-a202e263fde8] operation = "ListQueues"
[e9a801bb-c51d-45c8-8e9f-a202e263fde8] operation: 193.098393ms
[0740f0e0-953e-4328-94fc-830a5052e763] http: 195.185732ms
[0740f0e0-953e-4328-94fc-830a5052e763] requestID = "48b301fa-
fc9f-5f1f-9007-5c783caa9322"
[0740f0e0-953e-4328-94fc-830a5052e763] service    = "SQS"
[0740f0e0-953e-4328-94fc-830a5052e763] operation = "ListQueues"
[0740f0e0-953e-4328-94fc-830a5052e763] operation: 195.725491ms
[c0589832-f351-4cc7-84f1-c656eb79dbd7] http: 200.52383ms
[444030d0-6743-4de5-bd91-bc40b2b94c55] http: 200.525919ms
[c0589832-f351-4cc7-84f1-c656eb79dbd7] requestID = "4a73cc82-b47b-56e1-
b327-9100744e1b1f"

```

```

[c0589832-f351-4cc7-84f1-c656eb79dbd7] service    = "SQS"
[c0589832-f351-4cc7-84f1-c656eb79dbd7] operation = "ListQueues"
[c0589832-f351-4cc7-84f1-c656eb79dbd7] operation: 201.214365ms
[444030d0-6743-4de5-bd91-bc40b2b94c55] requestID = "ca1523ed-1879-5610-
bf5d-7e6fd84cabee"
[444030d0-6743-4de5-bd91-bc40b2b94c55] service    = "SQS"
[444030d0-6743-4de5-bd91-bc40b2b94c55] operation = "ListQueues"
[444030d0-6743-4de5-bd91-bc40b2b94c55] operation: 201.197071ms
[079e8dbd-bb93-43ab-89e5-a7bb392b86a5] http: 206.449568ms
[12b2b39d-df86-4648-a436-ff0482d13340] http: 206.526603ms
[079e8dbd-bb93-43ab-89e5-a7bb392b86a5] requestID = "64229710-b552-56ed-8f96-
ca927567ec7b"
[079e8dbd-bb93-43ab-89e5-a7bb392b86a5] service    = "SQS"
[079e8dbd-bb93-43ab-89e5-a7bb392b86a5] operation = "ListQueues"
[079e8dbd-bb93-43ab-89e5-a7bb392b86a5] operation: 207.252357ms
[12b2b39d-df86-4648-a436-ff0482d13340] requestID =
"76d9cbc0-07aa-58aa-98b7-9642c79f9851"
[12b2b39d-df86-4648-a436-ff0482d13340] service    = "SQS"
[12b2b39d-df86-4648-a436-ff0482d13340] operation = "ListQueues"
[12b2b39d-df86-4648-a436-ff0482d13340] operation: 207.360621ms

```

Pruebas unitarias con AWS SDK para Go v2

Cuando use el SDK en su aplicación, puede que quiera hacer una simulación de este para la prueba unitaria de la aplicación. Hacer una simulación del SDK permite que la prueba se centre en lo que quiere probar, no en los aspectos internos del SDK.

Para respaldar la simulación, use interfaces Go en lugar de tipos concretos de clientes de servicio, paginadores y esperadores, como `s3.Client`. Esto permite a la aplicación usar patrones como la inyección de dependencias para probar la lógica de la aplicación.

Simulación de operaciones de cliente

En este ejemplo, `S3GetObjectAPI` es una interfaz que define el conjunto de operaciones de la API de Amazon S3 que requiere la función `GetObjectFromS3`. `S3GetObjectAPI` se cumple con el método [GetObject](#) del cliente de Amazon S3.

```
import "context"
import "github.com/aws/aws-sdk-go-v2/service/s3"

// ...

type S3GetObjectAPI interface {
    GetObject(ctx context.Context, params *s3.GetObjectInput,
        optFns ...func(*s3.Options)) (*s3.GetObjectOutput, error)
}

func GetObjectFromS3(ctx context.Context, api S3GetObjectAPI, bucket, key string)
([]byte, error) {
    object, err := api.GetObject(ctx, &s3.GetObjectInput{
        Bucket: &bucket,
        Key:    &key,
    })
    if err != nil {
        return nil, err
    }
    defer object.Body.Close()

    return ioutil.ReadAll(object.Body)
}
```

Para probar la función `GetObjectFromS3`, utilice `mockGetObjectAPI` a fin de cumplir con la definición de la interfaz `S3GetObjectAPI`. A continuación, utilice el tipo `mockGetObjectAPI` para simular respuestas de salida y de error devueltas por el cliente de servicio.

```
import "testing"
import "github.com/aws/aws-sdk-go-v2/service/s3"

// ...

type mockGetObjectAPI func(ctx context.Context, params *s3.GetObjectInput,
    optFns ...func(*s3.Options)) (*s3.GetObjectOutput, error)

func (m mockGetObjectAPI) GetObject(ctx context.Context, params *s3.GetObjectInput,
    optFns ...func(*s3.Options)) (*s3.GetObjectOutput, error) {
    return m(ctx, params, optFns...)
}

func TestGetObjectFromS3(t *testing.T) {
    cases := []struct {
        client func(t *testing.T) S3GetObjectAPI
        bucket string
        key string
        expect []byte
    }{
        {
            client: func(t *testing.T) S3GetObjectAPI {
                return mockGetObjectAPI(func(ctx context.Context, params
*s3.GetObjectInput, optFns ...func(*s3.Options)) (*s3.GetObjectOutput, error) {
                    t.Helper()
                    if params.Bucket == nil {
                        t.Fatal("expect bucket to not be nil")
                    }
                    if e, a := "fooBucket", *params.Bucket; e != a {
                        t.Errorf("expect %v, got %v", e, a)
                    }
                    if params.Key == nil {
                        t.Fatal("expect key to not be nil")
                    }
                    if e, a := "barKey", *params.Key; e != a {
                        t.Errorf("expect %v, got %v", e, a)
                    }

                    return &s3.GetObjectOutput{
```



```

        Body: ioutil.NopCloser(bytes.NewReader([]byte("this is the body
foo bar baz"))),
    }, nil
})
},
bucket: "amzn-s3-demo-bucket>",
key:    "barKey",
expect: []byte("this is the body foo bar baz"),
},
}

for i, tt := range cases {
    t.Run(strconv.Itoa(i), func(t *testing.T) {
        ctx := context.TODO()
        content, err := GetObjectFromS3(ctx, tt.client(t), tt.bucket, tt.key)
        if err != nil {
            t.Fatalf("expect no error, got %v", err)
        }
        if e, a := tt.expect, content; bytes.Compare(e, a) != 0 {
            t.Errorf("expect %v, got %v", e, a)
        }
    })
}
}
}

```

Simulación de paginadores

Al igual que con los clientes de servicio, se puede hacer una simulación de los paginadores mediante la definición de una interfaz Go para el paginador. El código de su aplicación usaría esa interfaz. Esto permite utilizar la implementación del SDK durante la ejecución de la aplicación, así como una implementación simulada para la realización de pruebas.

En el siguiente ejemplo, `ListObjectsV2Pager` es una interfaz que define los comportamientos del objeto [ListObjectsV2Paginator](#) de Amazon S3 que requiere la función `CountObjects`.

```

import "context"
import "github.com/aws/aws-sdk-go-v2/service/s3"

// ...

type ListObjectsV2Pager interface {
    HasMorePages() bool
}

```

```

    NextPage(context.Context, ...func(*s3.Options)) (*s3.ListObjectsV2Output, error)
}

func CountObjects(ctx context.Context, pager ListObjectsV2Pager) (count int, err error) {
    for pager.HasMorePages() {
        var output *s3.ListObjectsV2Output
        output, err = pager.NextPage(ctx)
        if err != nil {
            return count, err
        }
        count += int(output.KeyCount)
    }
    return count, nil
}

```

Para probar `CountObjects`, cree el tipo `mockListObjectsV2Pager` que cumpla con la definición de la interfaz `ListObjectsV2Pager`. A continuación, use `mockListObjectsV2Pager` para replicar el comportamiento de paginación de las respuestas de salida y de error del paginador de operaciones de servicio.

```

import "context"
import "fmt"
import "testing"
import "github.com/aws/aws-sdk-go-v2/service/s3"

// ...

type mockListObjectsV2Pager struct {
    PageNum int
    Pages    []*s3.ListObjectsV2Output
}

func (m *mockListObjectsV2Pager) HasMorePages() bool {
    return m.PageNum < len(m.Pages)
}

func (m *mockListObjectsV2Pager) NextPage(ctx context.Context, f ...func(*s3.Options))
(output *s3.ListObjectsV2Output, err error) {
    if m.PageNum >= len(m.Pages) {
        return nil, fmt.Errorf("no more pages")
    }
    output = m.Pages[m.PageNum]
}

```

```
    m.PageNum++
    return output, nil
}

func TestCountObjects(t *testing.T) {
    pager := &mockListObjectsV2Pager{
        Pages: []*s3.ListObjectsV2Output{
            {
                KeyCount: 5,
            },
            {
                KeyCount: 10,
            },
            {
                KeyCount: 15,
            },
        },
    }
    objects, err := CountObjects(context.TODO(), pager)
    if err != nil {
        t.Fatalf("expect no error, got %v", err)
    }
    if expect, actual := 30, objects; expect != actual {
        t.Errorf("expect %v, got %v", expect, actual)
    }
}
```

Ejemplos de código del SDK para Go V2

Los ejemplos de código de este tema muestran cómo usar la AWS SDK para Go V2 con 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 API Gateway con SDK para Go V2](#)
- [Ejemplos de Aurora usando SDK para Go V2](#)
- [Ejemplos de Amazon Bedrock con SDK para Go V2](#)
- [Ejemplos de tiempo de ejecución de Amazon Bedrock con SDK para Go V2](#)
- [CloudFormation ejemplos de uso de SDK for Go V2](#)
- [CloudWatch Ejemplos de registros con SDK for Go V2](#)
- [Ejemplos del proveedor de identidades de Amazon Cognito con SDK para Go V2](#)
- [Ejemplos de Amazon DocumentDB con SDK para Go V2](#)
- [Ejemplos de DynamoDB usando SDK para Go V2](#)
- [Ejemplos de IAM usando SDK para Go V2](#)
- [Ejemplos de Kinesis usando SDK para Go V2](#)
- [Ejemplos de Lambda usando SDK para Go V2](#)
- [Ejemplos de Amazon MSK con SDK para Go V2](#)
- [Ejemplos de Centro de socios con SDK para Go V2](#)
- [Ejemplos de Amazon RDS usando SDK para Go V2](#)
- [Ejemplos de Amazon Redshift con SDK para Go V2](#)

- [Ejemplos de Amazon S3 usando SDK para Go V2](#)
- [Ejemplos de Amazon SNS usando SDK para Go V2](#)
- [Ejemplos de Amazon SQS usando SDK para Go V2](#)

Ejemplos de API Gateway con SDK para Go V2

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante la AWS SDK para Go V2 con API Gateway.

AWS Las contribuciones de la comunidad son ejemplos que fueron creados y mantenidos por varios equipos de distintos equipos AWS. Para enviar comentarios, utilice el mecanismo previsto en los repositorios vinculados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [AWS contribuciones de la comunidad](#)

AWS contribuciones de la comunidad

Cómo crear y probar una aplicación sin servidor

El siguiente ejemplo de código muestra cómo crear y probar una aplicación sin servidor mediante API Gateway con Lambda y DynamoDB.

SDK para Go V2

Muestra cómo crear y probar una aplicación sin servidor que consta de una API Gateway con Lambda y DynamoDB mediante el SDK de Go.

Para obtener el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda

Ejemplos de Aurora usando SDK para Go V2

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso de la AWS SDK para Go V2 con 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.

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 Aurora

En el siguiente ejemplo de código, se muestra cómo empezar a utilizar Aurora.

SDK para Go V2

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](#).

```
package main

import (
    "context"
```

```
"fmt"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/config"
"github.com/aws/aws-sdk-go-v2/service/rds"
)

// main uses the AWS SDK for Go V2 to create an Amazon Aurora client and list up to
20
// DB clusters in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
        fmt.Println(err)
        return
    }
    auroraClient := rds.NewFromConfig(sdkConfig)
    const maxClusters = 20
    fmt.Printf("Let's list up to %v DB clusters.\n", maxClusters)
    output, err := auroraClient.DescribeDBClusters(
        ctx, &rds.DescribeDBClustersInput{MaxRecords: aws.Int32(maxClusters)})
    if err != nil {
        fmt.Printf("Couldn't list DB clusters: %v\n", err)
        return
    }
    if len(output.DBClusters) == 0 {
        fmt.Println("No DB clusters found.")
    } else {
        for _, cluster := range output.DBClusters {
            fmt.Printf("DB cluster %v has database %v.\n", *cluster.DBClusterIdentifier,
                *cluster.DatabaseName)
        }
    }
}
```

- Para obtener más información sobre la API, consulta la [sección Describir DBClusters](#) en la referencia de la AWS SDK para Go 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 Go V2

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](#).

Ejecutar un escenario interactivo en un símbolo del sistema.

```
import (  
    "aurora/actions"  
    "context"  
    "fmt"  
    "log"  
    "slices"  
    "sort"  
    "strconv"  
    "strings"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/rds"
```



```

"github.com/aws/aws-sdk-go-v2/service/rds/types"
"github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
"github.com/google/uuid"
)

// GetStartedClusters is an interactive example that shows you how to use the AWS
// SDK for Go
// with Amazon Aurora to do the following:
//
// 1. Create a custom DB cluster parameter group and set parameter values.
// 2. Create an Aurora DB cluster that is configured to use the parameter group.
// 3. Create a DB instance in the DB cluster that contains a database.
// 4. Take a snapshot of the DB cluster.
// 5. Delete the DB instance, DB cluster, and parameter group.
type GetStartedClusters struct {
    sdkConfig  aws.Config
    dbClusters actions.DbClusters
    questioner demotools.IQuestioner
    helper     IScenarioHelper
    isTestRun  bool
}

// NewGetStartedClusters constructs a GetStartedClusters instance from a
// configuration.
// It uses the specified config to get an Amazon Relational Database Service (Amazon
// RDS)
// client and create wrappers for the actions used in the scenario.
func NewGetStartedClusters(sdkConfig aws.Config, questioner demotools.IQuestioner,
    helper IScenarioHelper) GetStartedClusters {
    auroraClient := rds.NewFromConfig(sdkConfig)
    return GetStartedClusters{
        sdkConfig:  sdkConfig,
        dbClusters: actions.DbClusters{AuroraClient: auroraClient},
        questioner: questioner,
        helper:     helper,
    }
}

// Run runs the interactive scenario.
func (scenario GetStartedClusters) Run(ctx context.Context, dbEngine string,
    parameterGroupName string,
    clusterName string, dbName string) {
    defer func() {
        if r := recover(); r != nil {

```

```

    log.Println("Something went wrong with the demo.")
}
}()

log.Println(strings.Repeat("-", 88))
log.Println("Welcome to the Amazon Aurora DB Cluster demo.")
log.Println(strings.Repeat("-", 88))

parameterGroup := scenario.CreateParameterGroup(ctx, dbEngine, parameterGroupName)
scenario.SetUserParameters(ctx, parameterGroupName)
cluster := scenario.CreateCluster(ctx, clusterName, dbEngine, dbName,
parameterGroup)
scenario.helper.Pause(5)
dbInstance := scenario.CreateInstance(ctx, cluster)
scenario.DisplayConnection(cluster)
scenario.CreateSnapshot(ctx, clusterName)
scenario.Cleanup(ctx, dbInstance, cluster, parameterGroup)

log.Println(strings.Repeat("-", 88))
log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}

// CreateParameterGroup shows how to get available engine versions for a specified
// database engine and create a DB cluster parameter group that is compatible with a
// selected engine family.
func (scenario GetStartedClusters) CreateParameterGroup(ctx context.Context,
dbEngine string,
parameterGroupName string) *types.DBClusterParameterGroup {

log.Printf("Checking for an existing DB cluster parameter group named %v.\n",
parameterGroupName)
parameterGroup, err := scenario.dbClusters.GetParameterGroup(ctx,
parameterGroupName)
if err != nil {
panic(err)
}
if parameterGroup == nil {
log.Printf("Getting available database engine versions for %v.\n", dbEngine)
engineVersions, err := scenario.dbClusters.GetEngineVersions(ctx, dbEngine, "")
if err != nil {
panic(err)
}

```

```

    familySet := map[string]struct{}{}
    for _, family := range engineVersions {
        familySet[*family.DBParameterGroupFamily] = struct{}{}
    }
    var families []string
    for family := range familySet {
        families = append(families, family)
    }
    sort.Strings(families)
    familyIndex := scenario.questioner.AskChoice("Which family do you want to use?\n",
families)
    log.Println("Creating a DB cluster parameter group.")
    _, err = scenario.dbClusters.CreateParameterGroup(
        ctx, parameterGroupName, families[familyIndex], "Example parameter group.")
    if err != nil {
        panic(err)
    }
    parameterGroup, err = scenario.dbClusters.GetParameterGroup(ctx,
parameterGroupName)
    if err != nil {
        panic(err)
    }
}
log.Printf("Parameter group %v:\n", *parameterGroup.DBParameterGroupFamily)
log.Printf("\tName: %v\n", *parameterGroup.DBClusterParameterGroupName)
log.Printf("\tARN: %v\n", *parameterGroup.DBClusterParameterGroupArn)
log.Printf("\tFamily: %v\n", *parameterGroup.DBParameterGroupFamily)
log.Printf("\tDescription: %v\n", *parameterGroup.Description)
log.Println(strings.Repeat("-", 88))
return parameterGroup
}

// SetUserParameters shows how to get the parameters contained in a custom parameter
// group and update some of the parameter values in the group.
func (scenario GetStartedClusters) SetUserParameters(ctx context.Context,
parameterGroupName string) {
    log.Println("Let's set some parameter values in your parameter group.")
    dbParameters, err := scenario.dbClusters.GetParameters(ctx, parameterGroupName, "")
    if err != nil {
        panic(err)
    }
    var updateParams []types.Parameter
    for _, dbParam := range dbParameters {

```

```

    if strings.HasPrefix(*dbParam.ParameterName, "auto_increment") &&
       *dbParam.IsModifiable && *dbParam.DataType == "integer" {
        log.Printf("The %v parameter is described as:\n\t%v",
            *dbParam.ParameterName, *dbParam.Description)
        rangeSplit := strings.Split(*dbParam.AllowedValues, "-")
        lower, _ := strconv.Atoi(rangeSplit[0])
        upper, _ := strconv.Atoi(rangeSplit[1])
        newValue := scenario.questioner.AskInt(
            fmt.Sprintf("Enter a value between %v and %v:", lower, upper),
            demotools.InIntRange{Lower: lower, Upper: upper})
        dbParam.ParameterValue = aws.String(strconv.Itoa(newValue))
        updateParams = append(updateParams, dbParam)
    }
}
err = scenario.dbClusters.UpdateParameters(ctx, parameterGroupName, updateParams)
if err != nil {
    panic(err)
}
log.Println("You can get a list of parameters you've set by specifying a source of
'user'.")
userParameters, err := scenario.dbClusters.GetParameters(ctx, parameterGroupName,
"user")
if err != nil {
    panic(err)
}
log.Println("Here are the parameters you've set:")
for _, param := range userParameters {
    log.Printf("\t%v: %v\n", *param.ParameterName, *param.ParameterValue)
}
log.Println(strings.Repeat("-", 88))
}

// CreateCluster shows how to create an Aurora DB cluster that contains a database
// of a specified type. The database is also configured to use a custom DB cluster
// parameter group.
func (scenario GetStartedClusters) CreateCluster(ctx context.Context, clusterName
string, dbEngine string,
dbName string, parameterGroup *types.DBClusterParameterGroup) *types.DBCluster {

    log.Println("Checking for an existing DB cluster.")
    cluster, err := scenario.dbClusters.GetDbCluster(ctx, clusterName)
    if err != nil {
        panic(err)
    }
}

```

```

if cluster == nil {
    adminUsername := scenario.questioner.Ask(
        "Enter an administrator user name for the database: ", demotools.NotEmpty{})
    adminPassword := scenario.questioner.Ask(
        "Enter a password for the administrator (at least 8 characters): ",
        demotools.NotEmpty{})
    engineVersions, err := scenario.dbClusters.GetEngineVersions(ctx, dbEngine,
        *parameterGroup.DBParameterGroupFamily)
    if err != nil {
        panic(err)
    }
    var engineChoices []string
    for _, engine := range engineVersions {
        engineChoices = append(engineChoices, *engine.EngineVersion)
    }
    log.Println("The available engines for your parameter group are:")
    engineIndex := scenario.questioner.AskChoice("Which engine do you want to use?\n",
        engineChoices)
    log.Printf("Creating DB cluster %v and database %v.\n", clusterName, dbName)
    log.Printf("The DB cluster is configured to use\nyour custom parameter group %v\n",
        *parameterGroup.DBClusterParameterGroupName)
    log.Printf("and selected engine %v.\n", engineChoices[engineIndex])
    log.Println("This typically takes several minutes.")
    cluster, err = scenario.dbClusters.CreateDbCluster(
        ctx, clusterName, *parameterGroup.DBClusterParameterGroupName, dbName, dbEngine,
        engineChoices[engineIndex], adminUsername, adminPassword)
    if err != nil {
        panic(err)
    }
    for *cluster.Status != "available" {
        scenario.helper.Pause(30)
        cluster, err = scenario.dbClusters.GetDbCluster(ctx, clusterName)
        if err != nil {
            panic(err)
        }
        log.Println("Cluster created and available.")
    }
}
log.Println("Cluster data:")
log.Printf("\tDBClusterIdentifier: %v\n", *cluster.DBClusterIdentifier)
log.Printf("\tARN: %v\n", *cluster.DBClusterArn)
log.Printf("\tStatus: %v\n", *cluster.Status)
log.Printf("\tEngine: %v\n", *cluster.Engine)

```

```

log.Printf("\tEngine version: %v\n", *cluster.EngineVersion)
log.Printf("\tDBClusterParameterGroup: %v\n", *cluster.DBClusterParameterGroup)
log.Printf("\tEngineMode: %v\n", *cluster.EngineMode)
log.Println(strings.Repeat("-", 88))
return cluster
}

// CreateInstance shows how to create a DB instance in an existing Aurora DB
// cluster.
// A new DB cluster contains no DB instances, so you must add one. The first DB
// instance
// that is added to a DB cluster defaults to a read-write DB instance.
func (scenario GetStartedClusters) CreateInstance(ctx context.Context, cluster
    *types.DBCluster) *types.DBInstance {
    log.Println("Checking for an existing database instance.")
    dbInstance, err := scenario.dbClusters.GetInstance(ctx,
        *cluster.DBClusterIdentifier)
    if err != nil {
        panic(err)
    }
    if dbInstance == nil {
        log.Println("Let's create a database instance in your DB cluster.")
        log.Println("First, choose a DB instance type:")
        instOpts, err := scenario.dbClusters.GetOrderableInstances(
            ctx, *cluster.Engine, *cluster.EngineVersion)
        if err != nil {
            panic(err)
        }
        var instChoices []string
        for _, opt := range instOpts {
            instChoices = append(instChoices, *opt.DBInstanceClass)
        }
        slices.Sort(instChoices)
        instChoices = slices.Compact(instChoices)
        instIndex := scenario.questioner.AskChoice(
            "Which DB instance class do you want to use?\n", instChoices)
        log.Println("Creating a database instance. This typically takes several minutes.")
        dbInstance, err = scenario.dbClusters.CreateInstanceInCluster(
            ctx, *cluster.DBClusterIdentifier, *cluster.DBClusterIdentifier, *cluster.Engine,
            instChoices[instIndex])
        if err != nil {
            panic(err)
        }
        for *dbInstance.DBInstanceStatus != "available" {

```

```

    scenario.helper.Pause(30)
    dbInstance, err = scenario.dbClusters.GetInstance(ctx,
*cluster.DBClusterIdentifier)
    if err != nil {
        panic(err)
    }
}

log.Println("Instance data:")
log.Printf("\tDBInstanceIdentifier: %v\n", *dbInstance.DBInstanceIdentifier)
log.Printf("\tARN: %v\n", *dbInstance.DBInstanceArn)
log.Printf("\tStatus: %v\n", *dbInstance.DBInstanceStatus)
log.Printf("\tEngine: %v\n", *dbInstance.Engine)
log.Printf("\tEngine version: %v\n", *dbInstance.EngineVersion)
log.Println(strings.Repeat("-", 88))
return dbInstance
}

// DisplayConnection displays connection information about an Aurora DB cluster and
tips
// on how to connect to it.
func (scenario GetStartedClusters) DisplayConnection(cluster *types.DBCluster) {
    log.Println(
        "You can now connect to your database using your favorite MySQL client.\n" +
        "One way to connect is by using the 'mysql' shell on an Amazon EC2 instance\n" +
        "that is running in the same VPC as your database cluster. Pass the endpoint,\n"
    +
        "port, and administrator user name to 'mysql' and enter your password\n" +
        "when prompted:")
    log.Printf("\n\tmysql -h %v -P %v -u %v -p\n",
        *cluster.Endpoint, *cluster.Port, *cluster.MasterUsername)
    log.Println("For more information, see the User Guide for Aurora:\n" +
        "\thttps://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/
CHAP_GettingStartedAurora.CreatingConnecting.Aurora.html#CHAP_GettingStartedAurora.Aurora.Co
log.Println(strings.Repeat("-", 88))
}

// CreateSnapshot shows how to create a DB cluster snapshot and wait until it's
available.
func (scenario GetStartedClusters) CreateSnapshot(ctx context.Context, clusterName
string) {
    if scenario.questioner.AskBool(
        "Do you want to create a snapshot of your DB cluster (y/n)? ", "y") {
        snapshotId := fmt.Sprintf("%v-%v", clusterName, scenario.helper.UniqueId())

```

```

    log.Printf("Creating a snapshot named %v. This typically takes a few minutes.\n",
snapshotId)
    snapshot, err := scenario.dbClusters.CreateClusterSnapshot(ctx, clusterName,
snapshotId)
    if err != nil {
        panic(err)
    }
    for *snapshot.Status != "available" {
        scenario.helper.Pause(30)
        snapshot, err = scenario.dbClusters.GetClusterSnapshot(ctx, snapshotId)
        if err != nil {
            panic(err)
        }
    }
    log.Println("Snapshot data:")
    log.Printf("\tDBClusterSnapshotIdentifier: %v\n",
*snapshot.DBClusterSnapshotIdentifier)
    log.Printf("\tARN: %v\n", *snapshot.DBClusterSnapshotArn)
    log.Printf("\tStatus: %v\n", *snapshot.Status)
    log.Printf("\tEngine: %v\n", *snapshot.Engine)
    log.Printf("\tEngine version: %v\n", *snapshot.EngineVersion)
    log.Printf("\tDBClusterIdentifier: %v\n", *snapshot.DBClusterIdentifier)
    log.Printf("\tSnapshotCreateTime: %v\n", *snapshot.SnapshotCreateTime)
    log.Println(strings.Repeat("-", 88))
}
}

// Cleanup shows how to clean up a DB instance, DB cluster, and DB cluster parameter
group.
// Before the DB cluster parameter group can be deleted, all associated DB instances
and
// DB clusters must first be deleted.
func (scenario GetStartedClusters) Cleanup(ctx context.Context, dbInstance
*types.DBInstance, cluster *types.DBCluster,
parameterGroup *types.DBClusterParameterGroup) {

    if scenario.questioner.AskBool(
        "\nDo you want to delete the database instance, DB cluster, and parameter group
(y/n)? ", "y") {
        log.Printf("Deleting database instance %v.\n", *dbInstance.DBInstanceIdentifier)
        err := scenario.dbClusters.DeleteInstance(ctx, *dbInstance.DBInstanceIdentifier)
        if err != nil {
            panic(err)
        }
    }
}

```



```

log.Printf("Deleting database cluster %v.\n", *cluster.DBClusterIdentifier)
err = scenario.dbClusters.DeleteDbCluster(ctx, *cluster.DBClusterIdentifier)
if err != nil {
    panic(err)
}
log.Println(
    "Waiting for the DB instance and DB cluster to delete. This typically takes
several minutes.")
for dbInstance != nil || cluster != nil {
    scenario.helper.Pause(30)
    if dbInstance != nil {
        dbInstance, err = scenario.dbClusters.GetInstance(ctx,
*dbInstance.DBInstanceIdentifier)
        if err != nil {
            panic(err)
        }
    }
    if cluster != nil {
        cluster, err = scenario.dbClusters.GetDbCluster(ctx,
*cluster.DBClusterIdentifier)
        if err != nil {
            panic(err)
        }
    }
}
log.Printf("Deleting parameter group %v.",
*parameterGroup.DBClusterParameterGroupName)
err = scenario.dbClusters.DeleteParameterGroup(ctx,
*parameterGroup.DBClusterParameterGroupName)
if err != nil {
    panic(err)
}
}
}

// IScenarioHelper abstracts the function from a scenario so that it
// can be mocked for unit testing.
type IScenarioHelper interface {
    Pause(secs int)
    UniqueId() string
}
type ScenarioHelper struct{}

// Pause waits for the specified number of seconds.

```

```
func (helper ScenarioHelper) Pause(secs int) {
    time.Sleep(time.Duration(secs) * time.Second)
}

// UniqueId returns a new UUID.
func (helper ScenarioHelper) UniqueId() string {
    return uuid.New().String()
}
```

Defina las funciones a las que llama el escenario para administrar las acciones de Aurora.

```
import (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/rds"
    "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
    AuroraClient *rds.Client
}

// GetParameterGroup gets a DB cluster parameter group by name.
func (clusters *DbClusters) GetParameterGroup(ctx context.Context,
    parameterGroupName string) (
    *types.DBClusterParameterGroup, error) {
    output, err := clusters.AuroraClient.DescribeDBClusterParameterGroups(
        ctx, &rds.DescribeDBClusterParameterGroupsInput{
            DBClusterParameterGroupName: aws.String(parameterGroupName),
        })
    if err != nil {
        var notFoundError *types.DBParameterGroupNotFoundFault
        if errors.As(err, &notFoundError) {
            log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
            err = nil
        } else {
```

```

    log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
}
return nil, err
} else {
    return &output.DBClusterParameterGroups[0], err
}
}

// CreateParameterGroup creates a DB cluster parameter group that is based on the
// specified
// parameter group family.
func (clusters *DbClusters) CreateParameterGroup(
    ctx context.Context, parameterGroupName string, parameterGroupFamily string,
    description string) (
    *types.DBClusterParameterGroup, error) {

    output, err := clusters.AuroraClient.CreateDBClusterParameterGroup(ctx,
        &rds.CreateDBClusterParameterGroupInput{
            DBClusterParameterGroupName: aws.String(parameterGroupName),
            DBParameterGroupFamily:      aws.String(parameterGroupFamily),
            Description:                 aws.String(description),
        })
    if err != nil {
        log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
        return nil, err
    } else {
        return output.DBClusterParameterGroup, err
    }
}

// DeleteParameterGroup deletes the named DB cluster parameter group.
func (clusters *DbClusters) DeleteParameterGroup(ctx context.Context,
    parameterGroupName string) error {
    _, err := clusters.AuroraClient.DeleteDBClusterParameterGroup(ctx,
        &rds.DeleteDBClusterParameterGroupInput{
            DBClusterParameterGroupName: aws.String(parameterGroupName),
        })
    if err != nil {
        log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
        return err
    } else {

```

```

    return nil
}
}

// GetParameters gets the parameters that are contained in a DB cluster parameter
// group.
func (clusters *DbClusters) GetParameters(ctx context.Context, parameterGroupName
string, source string) (
[]types.Parameter, error) {

var output *rds.DescribeDBClusterParametersOutput
var params []types.Parameter
var err error
parameterPaginator :=
rds.NewDescribeDBClusterParametersPaginator(clusters.AuroraClient,
&rds.DescribeDBClusterParametersInput{
    DBClusterParameterGroupName: aws.String(parameterGroupName),
    Source:                        aws.String(source),
})
for parameterPaginator.HasMorePages() {
    output, err = parameterPaginator.NextPage(ctx)
    if err != nil {
        log.Printf("Couldn't get parameters for %v: %v\n", parameterGroupName, err)
        break
    } else {
        params = append(params, output.Parameters...)
    }
}
return params, err
}

// UpdateParameters updates parameters in a named DB cluster parameter group.
func (clusters *DbClusters) UpdateParameters(ctx context.Context, parameterGroupName
string, params []types.Parameter) error {
_, err := clusters.AuroraClient.ModifyDBClusterParameterGroup(ctx,
&rds.ModifyDBClusterParameterGroupInput{
    DBClusterParameterGroupName: aws.String(parameterGroupName),
    Parameters:                  params,
})
if err != nil {

```

```

    log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
    return err
} else {
    return nil
}
}

// GetDbCluster gets data about an Aurora DB cluster.
func (clusters *DbClusters) GetDbCluster(ctx context.Context, clusterName string)
(*types.DBCluster, error) {
    output, err := clusters.AuroraClient.DescribeDBClusters(ctx,
        &rds.DescribeDBClustersInput{
            DBClusterIdentifier: aws.String(clusterName),
        })
    if err != nil {
        var notFoundError *types.DBClusterNotFoundFault
        if errors.As(err, &notFoundError) {
            log.Printf("DB cluster %v does not exist.\n", clusterName)
            err = nil
        } else {
            log.Printf("Couldn't get DB cluster %v: %v\n", clusterName, err)
        }
        return nil, err
    } else {
        return &output.DBClusters[0], err
    }
}

// CreateDbCluster creates a DB cluster that is configured to use the specified
// parameter group.
// The newly created DB cluster contains a database that uses the specified engine
// and
// engine version.
func (clusters *DbClusters) CreateDbCluster(ctx context.Context, clusterName string,
    parameterGroupName string,
    dbName string, dbEngine string, dbEngineVersion string, adminName string,
    adminPassword string) (
    *types.DBCluster, error) {

```

```

output, err := clusters.AuroraClient.CreateDBCluster(ctx,
&rds.CreateDBClusterInput{
    DBClusterIdentifier:    aws.String(clusterName),
    Engine:                 aws.String(dbEngine),
    DBClusterParameterGroupName: aws.String(parameterGroupName),
    DatabaseName:          aws.String(dbName),
    EngineVersion:         aws.String(dbEngineVersion),
    MasterUserPassword:     aws.String(adminPassword),
    MasterUsername:        aws.String(adminName),
})
if err != nil {
    log.Printf("Couldn't create DB cluster %v: %v\n", clusterName, err)
    return nil, err
} else {
    return output.DBCluster, err
}
}

// DeleteDbCluster deletes a DB cluster without keeping a final snapshot.
func (clusters *DbClusters) DeleteDbCluster(ctx context.Context, clusterName string)
error {
    _, err := clusters.AuroraClient.DeleteDBCluster(ctx, &rds.DeleteDBClusterInput{
        DBClusterIdentifier: aws.String(clusterName),
        SkipFinalSnapshot:   aws.Bool(true),
    })
    if err != nil {
        log.Printf("Couldn't delete DB cluster %v: %v\n", clusterName, err)
        return err
    } else {
        return nil
    }
}

// CreateClusterSnapshot creates a snapshot of a DB cluster.
func (clusters *DbClusters) CreateClusterSnapshot(ctx context.Context, clusterName
string, snapshotName string) (
    *types.DBClusterSnapshot, error) {
    output, err := clusters.AuroraClient.CreateDBClusterSnapshot(ctx,
&rds.CreateDBClusterSnapshotInput{
        DBClusterIdentifier:    aws.String(clusterName),

```

```

    DBClusterSnapshotIdentifier: aws.String(snapshotName),
  })
  if err != nil {
    log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
    return nil, err
  } else {
    return output.DBClusterSnapshot, nil
  }
}

// GetClusterSnapshot gets a DB cluster snapshot.
func (clusters *DbClusters) GetClusterSnapshot(ctx context.Context, snapshotName
string) (*types.DBClusterSnapshot, error) {
  output, err := clusters.AuroraClient.DescribeDBClusterSnapshots(ctx,
    &rds.DescribeDBClusterSnapshotsInput{
      DBClusterSnapshotIdentifier: aws.String(snapshotName),
    })
  if err != nil {
    log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
    return nil, err
  } else {
    return &output.DBClusterSnapshots[0], nil
  }
}

// CreateInstanceInCluster creates a database instance in an existing DB cluster.
// The first database that is
// created defaults to a read-write DB instance.
func (clusters *DbClusters) CreateInstanceInCluster(ctx context.Context, clusterName
string, instanceName string,
dbEngine string, dbInstanceClass string) (*types.DBInstance, error) {
  output, err := clusters.AuroraClient.CreateDBInstance(ctx,
    &rds.CreateDBInstanceInput{
      DBInstanceIdentifier: aws.String(instanceName),
      DBClusterIdentifier:   aws.String(clusterName),
      Engine:               aws.String(dbEngine),
      DBInstanceClass:      aws.String(dbInstanceClass),
    })
  if err != nil {
    log.Printf("Couldn't create instance %v: %v\n", instanceName, err)

```

```

    return nil, err
} else {
    return output.DBInstance, nil
}
}

// GetInstance gets data about a DB instance.
func (clusters *DbClusters) GetInstance(ctx context.Context, instanceName string) (
    *types.DBInstance, error) {
    output, err := clusters.AuroraClient.DescribeDBInstances(ctx,
        &rds.DescribeDBInstancesInput{
            DBInstanceIdentifier: aws.String(instanceName),
        })
    if err != nil {
        var notFoundError *types.DBInstanceNotFoundFault
        if errors.As(err, &notFoundError) {
            log.Printf("DB instance %v does not exist.\n", instanceName)
            err = nil
        } else {
            log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
        }
        return nil, err
    } else {
        return &output.DBInstances[0], nil
    }
}

// DeleteInstance deletes a DB instance.
func (clusters *DbClusters) DeleteInstance(ctx context.Context, instanceName string)
    error {
    _, err := clusters.AuroraClient.DeleteDBInstance(ctx, &rds.DeleteDBInstanceInput{
        DBInstanceIdentifier:  aws.String(instanceName),
        SkipFinalSnapshot:     aws.Bool(true),
        DeleteAutomatedBackups: aws.Bool(true),
    })
    if err != nil {
        log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
        return err
    } else {
        return nil
    }
}

```



```

}
}

// GetEngineVersions gets database engine versions that are available for the
// specified engine
// and parameter group family.
func (clusters *DbClusters) GetEngineVersions(ctx context.Context, engine string,
parameterGroupFamily string) (
[]types.DBEngineVersion, error) {
output, err := clusters.AuroraClient.DescribeDBEngineVersions(ctx,
&rds.DescribeDBEngineVersionsInput{
Engine:          aws.String(engine),
DBParameterGroupFamily: aws.String(parameterGroupFamily),
})
if err != nil {
log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
return nil, err
} else {
return output.DBEngineVersions, nil
}
}

// GetOrderableInstances uses a paginator to get DB instance options that can be
// used to create DB instances that are
// compatible with a set of specifications.
func (clusters *DbClusters) GetOrderableInstances(ctx context.Context, engine
string, engineVersion string) (
[]types.OrderableDBInstanceOption, error) {

var output *rds.DescribeOrderableDBInstanceOptionsOutput
var instances []types.OrderableDBInstanceOption
var err error
orderablePaginator :=
rds.NewDescribeOrderableDBInstanceOptionsPaginator(clusters.AuroraClient,
&rds.DescribeOrderableDBInstanceOptionsInput{
Engine:          aws.String(engine),
EngineVersion: aws.String(engineVersion),
})
for orderablePaginator.HasMorePages() {
output, err = orderablePaginator.NextPage(ctx)

```

```
if err != nil {
    log.Printf("Couldn't get orderable DB instances: %v\n", err)
    break
} else {
    instances = append(instances, output.OrderableDBInstanceOptions...)
}
}
return instances, err
}
```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK para Go .
 - [CrearDBCluster](#)
 - [CrearDBClusterParameterGroup](#)
 - [Crear DBCluster instantánea](#)
 - [CrearDBInstance](#)
 - [DeleteDBCluster](#)
 - [DeleteDBClusterParameterGroup](#)
 - [DeleteDBInstance](#)
 - [DescribirDBClusterParameterGroups](#)
 - [Describa DBCluster los parámetros](#)
 - [Describa las DBCluster instantáneas](#)
 - [DescribirDBClusters](#)
 - [Describa las versiones DBEngine](#)
 - [DescribirDBInstances](#)
 - [DescribeOrderableDBInstanceOpciones](#)
 - [ModifyDBClusterParameterGroup](#)

Acciones

CreateDBCluster

En el siguiente ejemplo de código, se muestra cómo utilizar CreateDBCluster.

SDK para Go V2

 Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
import (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/rds"
    "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
    AuroraClient *rds.Client
}

// CreateDbCluster creates a DB cluster that is configured to use the specified
// parameter group.
// The newly created DB cluster contains a database that uses the specified engine
// and
// engine version.
func (clusters *DbClusters) CreateDbCluster(ctx context.Context, clusterName string,
    parameterGroupName string,
    dbName string, dbEngine string, dbEngineVersion string, adminName string,
    adminPassword string) (
    *types.DBCluster, error) {

    output, err := clusters.AuroraClient.CreateDBCluster(ctx,
    &rds.CreateDBClusterInput{
        DBClusterIdentifier:    aws.String(clusterName),
        Engine:                 aws.String(dbEngine),
        DBClusterParameterGroupName: aws.String(parameterGroupName),
        DatabaseName:          aws.String(dbName),
```

```

    EngineVersion:      aws.String(dbEngineVersion),
    MasterUserPassword:  aws.String(adminPassword),
    MasterUsername:      aws.String(adminName),
  })
  if err != nil {
    log.Printf("Couldn't create DB cluster %v: %v\n", clusterName, err)
    return nil, err
  } else {
    return output.DBCluster, err
  }
}

```

- Para obtener más información sobre la API, consulta la [sección Crear DBCluster](#) en la referencia de la AWS SDK para Go API.

CreateDBClusterParameterGroup

En el siguiente ejemplo de código, se muestra cómo utilizar CreateDBClusterParameterGroup.

SDK para Go V2

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 (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/rds"
    "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {

```

```

AuroraClient *rds.Client
}

// CreateParameterGroup creates a DB cluster parameter group that is based on the
// specified
// parameter group family.
func (clusters *DbClusters) CreateParameterGroup(
    ctx context.Context, parameterGroupName string, parameterGroupFamily string,
    description string) (
    *types.DBClusterParameterGroup, error) {

    output, err := clusters.AuroraClient.CreateDBClusterParameterGroup(ctx,
        &rds.CreateDBClusterParameterGroupInput{
            DBClusterParameterGroupName: aws.String(parameterGroupName),
            DBParameterGroupFamily:      aws.String(parameterGroupFamily),
            Description:                 aws.String(description),
        })
    if err != nil {
        log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
        return nil, err
    } else {
        return output.DBClusterParameterGroup, err
    }
}

```

- Para obtener más información sobre la API, consulta la [sección Crear DBCluster ParameterGroup](#) en la referencia de la AWS SDK para Go API.

CreateDBClusterSnapshot

En el siguiente ejemplo de código, se muestra cómo utilizar CreateDBClusterSnapshot.

SDK para Go V2

 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 (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/rds"
    "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
    AuroraClient *rds.Client
}

// CreateClusterSnapshot creates a snapshot of a DB cluster.
func (clusters *DbClusters) CreateClusterSnapshot(ctx context.Context, clusterName
    string, snapshotName string) (
    *types.DBClusterSnapshot, error) {
    output, err := clusters.AuroraClient.CreateDBClusterSnapshot(ctx,
    &rds.CreateDBClusterSnapshotInput{
        DBClusterIdentifier:      aws.String(clusterName),
        DBClusterSnapshotIdentifier: aws.String(snapshotName),
    })
    if err != nil {
        log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
        return nil, err
    } else {
        return output.DBClusterSnapshot, nil
    }
}
```

- Para obtener más información sobre la API, consulta [Crear DBCluster una instantánea](#) en la referencia AWS SDK para Go de la API.

CreateDBInstance

En el siguiente ejemplo de código, se muestra cómo utilizar CreateDBInstance.

SDK para Go V2

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 (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/rds"
    "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
    AuroraClient *rds.Client
}

// CreateInstanceInCluster creates a database instance in an existing DB cluster.
// The first database that is
// created defaults to a read-write DB instance.
func (clusters *DbClusters) CreateInstanceInCluster(ctx context.Context, clusterName
    string, instanceName string,
    dbEngine string, dbInstanceClass string) (*types.DBInstance, error) {
```

```
output, err := clusters.AuroraClient.CreateDBInstance(ctx,
&rds.CreateDBInstanceInput{
    DBInstanceIdentifier: aws.String(instanceName),
    DBClusterIdentifier:  aws.String(clusterName),
    Engine:               aws.String(dbEngine),
    DBInstanceClass:      aws.String(dbInstanceClass),
})
if err != nil {
    log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
    return nil, err
} else {
    return output.DBInstance, nil
}
}
```

- Para obtener más información sobre la API, consulta la [sección Crear DBInstance](#) en la referencia de la AWS SDK para Go API.

DeleteDBCluster

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteDBCluster.

SDK para Go V2

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 (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/rds"
    "github.com/aws/aws-sdk-go-v2/service/rds/types"
)
```



```
)

type DbClusters struct {
    AuroraClient *rds.Client
}

// DeleteDbCluster deletes a DB cluster without keeping a final snapshot.
func (clusters *DbClusters) DeleteDbCluster(ctx context.Context, clusterName string)
error {
    _, err := clusters.AuroraClient.DeleteDBCluster(ctx, &rds.DeleteDBClusterInput{
        DBClusterIdentifier: aws.String(clusterName),
        SkipFinalSnapshot:   aws.Bool(true),
    })
    if err != nil {
        log.Printf("Couldn't delete DB cluster %v: %v\n", clusterName, err)
        return err
    } else {
        return nil
    }
}
```

- Para obtener más información sobre la API, consulta [Eliminar DBCluster](#) en la referencia AWS SDK para Go de la API.

DeleteDBClusterParameterGroup

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteDBClusterParameterGroup.

SDK para Go V2

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 (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/rds"
    "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
    AuroraClient *rds.Client
}

// DeleteParameterGroup deletes the named DB cluster parameter group.
func (clusters *DbClusters) DeleteParameterGroup(ctx context.Context,
    parameterGroupName string) error {
    _, err := clusters.AuroraClient.DeleteDBClusterParameterGroup(ctx,
        &rds.DeleteDBClusterParameterGroupInput{
            DBClusterParameterGroupName: aws.String(parameterGroupName),
        })
    if err != nil {
        log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
        return err
    } else {
        return nil
    }
}
```

- Para obtener más información sobre la API, consulta [Eliminar DBCluster ParameterGroup](#) en la referencia AWS SDK para Go de la API.

DeleteDBInstance

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteDBInstance.

SDK para Go V2

 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 (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/rds"
    "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
    AuroraClient *rds.Client
}

// DeleteInstance deletes a DB instance.
func (clusters *DbClusters) DeleteInstance(ctx context.Context, instanceName string)
    error {
    _, err := clusters.AuroraClient.DeleteDBInstance(ctx, &rds.DeleteDBInstanceInput{
        DBInstanceIdentifier: aws.String(instanceName),
        SkipFinalSnapshot:    aws.Bool(true),
        DeleteAutomatedBackups: aws.Bool(true),
    })
    if err != nil {
        log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
        return err
    } else {
        return nil
    }
}
```

- Para obtener más información sobre la API, consulta [Eliminar DBInstance](#) en la referencia AWS SDK para Go de la API.

DescribeDBClusterParameterGroups

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeDBClusterParameterGroups`.

SDK para Go V2

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 (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/rds"
    "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
    AuroraClient *rds.Client
}

// GetParameterGroup gets a DB cluster parameter group by name.
func (clusters *DbClusters) GetParameterGroup(ctx context.Context,
    parameterGroupName string) (
    *types.DBClusterParameterGroup, error) {
    output, err := clusters.AuroraClient.DescribeDBClusterParameterGroups(
        ctx, &rds.DescribeDBClusterParameterGroupsInput{
            DBClusterParameterGroupName: aws.String(parameterGroupName),
        })
}
```

```
if err != nil {
    var notFoundError *types.DBParameterGroupNotFoundFault
    if errors.As(err, &notFoundError) {
        log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
        err = nil
    } else {
        log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
    }
    return nil, err
} else {
    return &output.DBClusterParameterGroups[0], err
}
}
```

- Para obtener más información sobre la API, consulta la [sección Describir DBClusterParameterGroups](#) en la referencia de la AWS SDK para Go API.

DescribeDBClusterParameters

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeDBClusterParameters`.

SDK para Go V2

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 (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/rds"
    "github.com/aws/aws-sdk-go-v2/service/rds/types"
)
```

```
type DbClusters struct {
    AuroraClient *rds.Client
}

// GetParameters gets the parameters that are contained in a DB cluster parameter
// group.
func (clusters *DbClusters) GetParameters(ctx context.Context, parameterGroupName
    string, source string) (
    []types.Parameter, error) {

    var output *rds.DescribeDBClusterParametersOutput
    var params []types.Parameter
    var err error
    parameterPaginator :=
    rds.NewDescribeDBClusterParametersPaginator(clusters.AuroraClient,
        &rds.DescribeDBClusterParametersInput{
            DBClusterParameterGroupName: aws.String(parameterGroupName),
            Source:                      aws.String(source),
        })
    for parameterPaginator.HasMorePages() {
        output, err = parameterPaginator.NextPage(ctx)
        if err != nil {
            log.Printf("Couldn't get parameters for %v: %v\n", parameterGroupName, err)
            break
        } else {
            params = append(params, output.Parameters...)
        }
    }
    return params, err
}
```

- Para obtener información detallada sobre la API, consulta [Describir DBCluster los parámetros](#) en la referencia de la AWS SDK para Go API.

DescribeDBClusterSnapshots

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeDBClusterSnapshots`.

SDK para Go V2

 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 (  
    "context"  
    "errors"  
    "log"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/rds"  
    "github.com/aws/aws-sdk-go-v2/service/rds/types"  
)  
  
type DbClusters struct {  
    AuroraClient *rds.Client  
}  
  
// GetClusterSnapshot gets a DB cluster snapshot.  
func (clusters *DbClusters) GetClusterSnapshot(ctx context.Context, snapshotName  
    string) (*types.DBClusterSnapshot, error) {  
    output, err := clusters.AuroraClient.DescribeDBClusterSnapshots(ctx,  
        &rds.DescribeDBClusterSnapshotsInput{  
            DBClusterSnapshotIdentifier: aws.String(snapshotName),  
        })  
    if err != nil {  
        log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)  
        return nil, err  
    } else {  
        return &output.DBClusterSnapshots[0], nil  
    }  
}
```

- Para obtener más información sobre la API, consulte [Describir DBCluster las instantáneas](#) en la referencia de AWS SDK para Go la API.

DescribeDBClusters

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeDBClusters.

SDK para Go V2

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 (  
    "context"  
    "errors"  
    "log"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/rds"  
    "github.com/aws/aws-sdk-go-v2/service/rds/types"  
)  
  
type DbClusters struct {  
    AuroraClient *rds.Client  
}  
  
// GetDbCluster gets data about an Aurora DB cluster.  
func (clusters *DbClusters) GetDbCluster(ctx context.Context, clusterName string)  
(*types.DBCluster, error) {  
    output, err := clusters.AuroraClient.DescribeDBClusters(ctx,  
        &rds.DescribeDBClustersInput{  
            DBClusterIdentifier: aws.String(clusterName),  
        })  
    if err != nil {  
        var notFoundError *types.DBClusterNotFoundFault  
        if errors.As(err, &notFoundError) {
```



```
    log.Printf("DB cluster %v does not exist.\n", clusterName)
    err = nil
} else {
    log.Printf("Couldn't get DB cluster %v: %v\n", clusterName, err)
}
return nil, err
} else {
    return &output.DBClusters[0], err
}
}
```

- Para obtener más información sobre la API, consulta la [sección Describir DBClusters](#) en la referencia de la AWS SDK para Go API.

DescribeDBEngineVersions

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeDBEngineVersions.

SDK para Go V2

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 (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/rds"
    "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
    AuroraClient *rds.Client
```

```
}

// GetEngineVersions gets database engine versions that are available for the
// specified engine
// and parameter group family.
func (clusters *DbClusters) GetEngineVersions(ctx context.Context, engine string,
parameterGroupFamily string) (
[]types.DBEngineVersion, error) {
output, err := clusters.AuroraClient.DescribeDBEngineVersions(ctx,
&rds.DescribeDBEngineVersionsInput{
Engine:          aws.String(engine),
DBParameterGroupFamily: aws.String(parameterGroupFamily),
})
if err != nil {
log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
return nil, err
} else {
return output.DBEngineVersions, nil
}
}
```

- Para obtener más información sobre la API, consulta la [sección Describir DBEngine las versiones](#) en la referencia de la AWS SDK para Go API.

DescribeDBInstances

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeDBInstances.

SDK para Go V2

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 (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/rds"
    "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
    AuroraClient *rds.Client
}

// GetInstance gets data about a DB instance.
func (clusters *DbClusters) GetInstance(ctx context.Context, instanceName string) (
    *types.DBInstance, error) {
    output, err := clusters.AuroraClient.DescribeDBInstances(ctx,
        &rds.DescribeDBInstancesInput{
            DBInstanceIdentifier: aws.String(instanceName),
        })
    if err != nil {
        var notFoundError *types.DBInstanceNotFoundFault
        if errors.As(err, &notFoundError) {
            log.Printf("DB instance %v does not exist.\n", instanceName)
            err = nil
        } else {
            log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
        }
        return nil, err
    } else {
        return &output.DBInstances[0], nil
    }
}
```

- Para obtener más información sobre la API, consulta la [sección Describir DBInstances](#) en la referencia de la AWS SDK para Go API.

DescribeOrderableDBInstanceOptions

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeOrderableDBInstanceOptions`.

SDK para Go V2

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 (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/rds"
    "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
    AuroraClient *rds.Client
}

// GetOrderableInstances uses a paginator to get DB instance options that can be
// used to create DB instances that are
// compatible with a set of specifications.
func (clusters *DbClusters) GetOrderableInstances(ctx context.Context, engine
    string, engineVersion string) (
    []types.OrderableDBInstanceOption, error) {

    var output *rds.DescribeOrderableDBInstanceOptionsOutput
    var instances []types.OrderableDBInstanceOption
    var err error
    orderablePaginator :=
    rds.NewDescribeOrderableDBInstanceOptionsPaginator(clusters.AuroraClient,
```

```
&rds.DescribeOrderableDBInstanceOptionsInput{
    Engine:      aws.String(engine),
    EngineVersion: aws.String(engineVersion),
})
for orderablePaginator.HasMorePages() {
    output, err = orderablePaginator.NextPage(ctx)
    if err != nil {
        log.Printf("Couldn't get orderable DB instances: %v\n", err)
        break
    } else {
        instances = append(instances, output.OrderableDBInstanceOptions...)
    }
}
return instances, err
}
```

- Para obtener más información sobre la API, consulta [DescribeOrderableDBInstanceOptions](#) en la referencia AWS SDK para Go de la API.

ModifyDBClusterParameterGroup

En el siguiente ejemplo de código, se muestra cómo utilizar ModifyDBClusterParameterGroup.

SDK para Go V2

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 (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/rds"
```

```

    "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbClusters struct {
    AuroraClient *rds.Client
}

// UpdateParameters updates parameters in a named DB cluster parameter group.
func (clusters *DbClusters) UpdateParameters(ctx context.Context, parameterGroupName
    string, params []types.Parameter) error {
    _, err := clusters.AuroraClient.ModifyDBClusterParameterGroup(ctx,
        &rds.ModifyDBClusterParameterGroupInput{
            DBClusterParameterGroupName: aws.String(parameterGroupName),
            Parameters:                  params,
        })
    if err != nil {
        log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
        return err
    } else {
        return nil
    }
}

```

- Para obtener más información sobre la API, consulta [Modificar DBCluster ParameterGroup](#) en la referencia AWS SDK para Go de la API.

Ejemplos de Amazon Bedrock con SDK para Go V2

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso de la AWS SDK para Go V2 con Amazon Bedrock.

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

Introducción a Amazon Bedrock

En el siguiente ejemplo de código se muestra cómo empezar a utilizar Amazon Bedrock.

SDK para Go V2

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](#).

```
package main

import (
    "context"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/bedrock"
)

const region = "us-east-1"

// main uses the AWS SDK for Go (v2) to create an Amazon Bedrock client and
// list the available foundation models in your account and the chosen region.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx, config.WithRegion(region))
    if err != nil {
        fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
    }
}
```

```
    fmt.Println(err)
    return
}
bedrockClient := bedrock.NewFromConfig(sdkConfig)
result, err := bedrockClient.ListFoundationModels(ctx,
    &bedrock.ListFoundationModelsInput{})
if err != nil {
    fmt.Printf("Couldn't list foundation models. Here's why: %v\n", err)
    return
}
if len(result.ModelSummaries) == 0 {
    fmt.Println("There are no foundation models.")
}
for _, modelSummary := range result.ModelSummaries {
    fmt.Println(*modelSummary.ModelId)
}
}
```

- Para obtener más información sobre la API, consulta [ListFoundationModels](#) la Referencia AWS SDK para Go de la API.

Acciones

ListFoundationModels

En el siguiente ejemplo de código, se muestra cómo utilizar ListFoundationModels.

SDK para Go V2

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](#).

Enumerar los modelos fundacionales Bedrock disponibles

```
import (
```



```

"context"
"log"

"github.com/aws/aws-sdk-go-v2/service/bedrock"
"github.com/aws/aws-sdk-go-v2/service/bedrock/types"
)

// FoundationModelWrapper encapsulates Amazon Bedrock actions used in the examples.
// It contains a Bedrock service client that is used to perform foundation model
// actions.
type FoundationModelWrapper struct {
    BedrockClient *bedrock.Client
}

// ListPolicies lists Bedrock foundation models that you can use.
func (wrapper FoundationModelWrapper) ListFoundationModels(ctx context.Context)
    ([]types.FoundationModelSummary, error) {

    var models []types.FoundationModelSummary

    result, err := wrapper.BedrockClient.ListFoundationModels(ctx,
        &bedrock.ListFoundationModelsInput{})

    if err != nil {
        log.Printf("Couldn't list foundation models. Here's why: %v\n", err)
    } else {
        models = result.ModelSummaries
    }
    return models, err
}

```

- Para obtener más información sobre la API, consulta [ListFoundationModels](#) la Referencia AWS SDK para Go de la API.

Ejemplos de tiempo de ejecución de Amazon Bedrock con SDK para Go V2

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso de la AWS SDK para Go V2 con Amazon Bedrock Runtime.

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](#)
- [Escenarios](#)
- [Amazon Titan Image Generator](#)
- [Anthropic Claude](#)

Introducción

Introducción a Amazon Bedrock

En el siguiente ejemplo de código se muestra cómo empezar a utilizar Amazon Bedrock.

SDK para Go V2

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](#).

```
package main

import (
    "context"
    "encoding/json"
```

```
"flag"
"fmt"
"log"
"os"
"strings"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/config"
"github.com/aws/aws-sdk-go-v2/service/bedrockruntime"
)

// Each model provider defines their own individual request and response formats.
// For the format, ranges, and default values for the different models, refer to:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters.html

type ClaudeRequest struct {
    Prompt          string `json:"prompt"`
    MaxTokensToSample int    `json:"max_tokens_to_sample"`
    // Omitting optional request parameters
}

type ClaudeResponse struct {
    Completion string `json:"completion"`
}

// main uses the AWS SDK for Go (v2) to create an Amazon Bedrock Runtime client
// and invokes Anthropic Claude 2 inside your account and the chosen region.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {

    region := flag.String("region", "us-east-1", "The AWS region")
    flag.Parse()

    fmt.Printf("Using AWS region: %s\n", *region)

    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx, config.WithRegion(*region))
    if err != nil {
        fmt.Println("Couldn't load default configuration. Have you set up your AWS account?")
        fmt.Println(err)
        return
    }
}
```

```
client := bedrockruntime.NewFromConfig(sdkConfig)

modelId := "anthropic.claude-v2"

prompt := "Hello, how are you today?"

// Anthropic Claude requires you to enclose the prompt as follows:
prefix := "Human: "
postfix := "\n\nAssistant:"
wrappedPrompt := prefix + prompt + postfix

request := ClaudeRequest{
    Prompt:          wrappedPrompt,
    MaxTokensToSample: 200,
}

body, err := json.Marshal(request)
if err != nil {
    log.Panicln("Couldn't marshal the request: ", err)
}

result, err := client.InvokeModel(ctx, &bedrockruntime.InvokeModelInput{
    ModelId:      aws.String(modelId),
    ContentType: aws.String("application/json"),
    Body:         body,
})

if err != nil {
    errMsg := err.Error()
    if strings.Contains(errMsg, "no such host") {
        fmt.Printf("Error: The Bedrock service is not available in the selected
region. Please double-check the service availability for your region at https://
aws.amazon.com/about-aws/global-infrastructure/regional-product-services/.\n")
    } else if strings.Contains(errMsg, "Could not resolve the foundation model") {
        fmt.Printf("Error: Could not resolve the foundation model from model identifier:
\"%v\". Please verify that the requested model exists and is accessible within the
specified region.\n", modelId)
    } else {
        fmt.Printf("Error: Couldn't invoke Anthropic Claude. Here's why: %v\n", err)
    }
    os.Exit(1)
}
```

```
var response ClaudeResponse

err = json.Unmarshal(result.Body, &response)

if err != nil {
    log.Fatal("failed to unmarshal", err)
}
fmt.Println("Prompt:\n", prompt)
fmt.Println("Response from Anthropic Claude:\n", response.Completion)
}
```

- Para obtener más información sobre la API, consulta [InvokeModel](#) la Referencia AWS SDK para Go de la API.

Escenarios

Invocar varios modelos fundacionales en Amazon Bedrock

El siguiente ejemplo de código muestra cómo preparar y enviar un mensaje a una variedad de modelos de idiomas extensos (LLMs) en Amazon Bedrock

SDK para Go V2

Note

Hay más información sobre. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Invoque varios modelos fundacionales en Amazon Bedrock.

```
import (
    "context"
    "encoding/base64"
    "fmt"
    "log"
    "math/rand"
    "os"
```

```

"path/filepath"
"strings"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/service/bedrockruntime"
"github.com/awsdocs/aws-doc-sdk-examples/gov2/bedrock-runtime/actions"
"github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// InvokeModelsScenario demonstrates how to use the Amazon Bedrock Runtime client
// to invoke various foundation models for text and image generation
//
// 1. Generate text with Anthropic Claude 2
// 2. Generate text with Meta Llama 2 Chat
// 3. Generate text and asynchronously process the response stream with Anthropic
//    Claude 2
// 4. Generate an image with the Amazon Titan image generation model
type InvokeModelsScenario struct {
    sdkConfig          aws.Config
    invokeModelWrapper actions.InvokeModelWrapper
    responseStreamWrapper actions.InvokeModelWithResponseStreamWrapper
    questioner         demotools.IQuestioner
}

// NewInvokeModelsScenario constructs an InvokeModelsScenario instance from a
// configuration.
// It uses the specified config to get a Bedrock Runtime client and create wrappers
// for the
// actions used in the scenario.
func NewInvokeModelsScenario(sdkConfig aws.Config, questioner demotools.IQuestioner)
    InvokeModelsScenario {
    client := bedrockruntime.NewFromConfig(sdkConfig)
    return InvokeModelsScenario{
        sdkConfig:          sdkConfig,
        invokeModelWrapper: actions.InvokeModelWrapper{BedrockRuntimeClient: client},
        responseStreamWrapper:
            actions.InvokeModelWithResponseStreamWrapper{BedrockRuntimeClient: client},
        questioner:         questioner,
    }
}

// Runs the interactive scenario.
func (scenario InvokeModelsScenario) Run(ctx context.Context) {
    defer func() {

```

```

    if r := recover(); r != nil {
        log.Printf("Something went wrong with the demo: %v\n", r)
    }
}()

log.Println(strings.Repeat("=", 77))
log.Println("Welcome to the Amazon Bedrock Runtime model invocation demo.")
log.Println(strings.Repeat("=", 77))

log.Printf("First, let's invoke a few large-language models using the synchronous
client:\n\n")

text2textPrompt := "In one paragraph, who are you?"

log.Println(strings.Repeat("-", 77))
log.Printf("Invoking Claude with prompt: %v\n", text2textPrompt)
scenario.InvokeClaude(ctx, text2textPrompt)

log.Println(strings.Repeat("=", 77))
log.Printf("Now, let's invoke Claude with the asynchronous client and process the
response stream:\n\n")

log.Println(strings.Repeat("-", 77))
log.Printf("Invoking Claude with prompt: %v\n", text2textPrompt)
scenario.InvokeWithResponseStream(ctx, text2textPrompt)

log.Println(strings.Repeat("=", 77))
log.Printf("Now, let's create an image with the Amazon Titan image generation
model:\n\n")

text2ImagePrompt := "stylized picture of a cute old steampunk robot"
seed := rand.Int63n(2147483648)

log.Println(strings.Repeat("-", 77))
log.Printf("Invoking Amazon Titan with prompt: %v\n", text2ImagePrompt)
scenario.InvokeTitanImage(ctx, text2ImagePrompt, seed)

log.Println(strings.Repeat("=", 77))
log.Println("Thanks for watching!")
log.Println(strings.Repeat("=", 77))
}

func (scenario InvokeModelsScenario) InvokeClaude(ctx context.Context, prompt
string) {

```

```

completion, err := scenario.invokeModelWrapper.InvokeClaude(ctx, prompt)
if err != nil {
    panic(err)
}
log.Printf("\nClaude      : %v\n", strings.TrimSpace(completion))
}

func (scenario InvokeModelsScenario) InvokeWithResponseStream(ctx context.Context,
    prompt string) {
    log.Println("\nClaude with response stream:")
    _, err := scenario.responseStreamWrapper.InvokeModelWithResponseStream(ctx, prompt)
    if err != nil {
        panic(err)
    }
    log.Println()
}

func (scenario InvokeModelsScenario) InvokeTitanImage(ctx context.Context, prompt
    string, seed int64) {
    base64ImageData, err := scenario.invokeModelWrapper.InvokeTitanImage(ctx, prompt,
        seed)
    if err != nil {
        panic(err)
    }
    imagePath := saveImage(base64ImageData, "amazon.titan-image-generator-v2")
    fmt.Printf("The generated image has been saved to %s\n", imagePath)
}

```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Go .
 - [InvokeModel](#)
 - [InvokeModelWithResponseStream](#)

Amazon Titan Image Generator

InvokeModel

El siguiente ejemplo de código muestra cómo invocar Amazon Titan Image en Amazon Bedrock para generar una imagen.

SDK para Go V2

 Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Cree una imagen con Amazon Titan Image Generator.

```
import (
    "context"
    "encoding/json"
    "log"
    "strings"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/bedrockruntime"
)

// InvokeModelWrapper encapsulates Amazon Bedrock actions used in the examples.
// It contains a Bedrock Runtime client that is used to invoke foundation models.
type InvokeModelWrapper struct {
    BedrockRuntimeClient *bedrockruntime.Client
}

type TitanImageRequest struct {
    TaskType          string          `json:"taskType"`
    TextToImageParams TextToImageParams `json:"textToImageParams"`
    ImageGenerationConfig ImageGenerationConfig `json:"imageGenerationConfig"`
}

type TextToImageParams struct {
    Text string `json:"text"`
}

type ImageGenerationConfig struct {
    NumberOfImages int    `json:"numberOfImages"`
    Quality        string `json:"quality"`
    CfgScale       float64 `json:"cfgScale"`
    Height         int    `json:"height"`
    Width          int    `json:"width"`
}
```

```

    Seed          int64    `json:"seed"`
}

type TitanImageResponse struct {
    Images []string `json:"images"`
}

// Invokes the Titan Image model to create an image using the input provided
// in the request body.
func (wrapper InvokeModelWrapper) InvokeTitanImage(ctx context.Context, prompt
    string, seed int64) (string, error) {
    modelId := "amazon.titan-image-generator-v2:0"

    body, err := json.Marshal(TitanImageRequest{
        TaskType: "TEXT_IMAGE",
        TextToImageParams: TextToImageParams{
            Text: prompt,
        },
        ImageGenerationConfig: ImageGenerationConfig{
            NumberOfImages: 1,
            Quality:       "standard",
            CfgScale:       8.0,
            Height:        512,
            Width:         512,
            Seed:           seed,
        },
    })

    if err != nil {
        log.Fatal("failed to marshal", err)
    }

    output, err := wrapper.BedrockRuntimeClient.InvokeModel(ctx,
        &bedrockruntime.InvokeModelInput{
            ModelId:      aws.String(modelId),
            ContentType: aws.String("application/json"),
            Body:        body,
        })

    if err != nil {
        ProcessError(err, modelId)
    }

    var response TitanImageResponse

```

```
if err := json.Unmarshal(output.Body, &response); err != nil {
    log.Fatal("failed to unmarshal", err)
}

base64ImageData := response.Images[0]

return base64ImageData, nil
}
```

- Para obtener más información sobre la API, consulta [InvokeModel](#) la Referencia AWS SDK para Go de la API.

Anthropic Claude

Converse

El siguiente ejemplo de código muestra cómo enviar un mensaje de texto a Amazon Claude con la API de Converse de Bedrock.

SDK para Go V2

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](#).

Envíe un mensaje de texto a Anthropic Claude mediante la API de Converse de Bedrock.

```
import (
    "context"
    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/bedrockruntime"
    "github.com/aws/aws-sdk-go-v2/service/bedrockruntime/types"
)

// ConverseWrapper encapsulates Amazon Bedrock actions used in the examples.
```

```
// It contains a Bedrock Runtime client that is used to invoke Bedrock.
type ConverseWrapper struct {
    BedrockRuntimeClient *bedrockruntime.Client
}

func (wrapper ConverseWrapper) ConverseClaude(ctx context.Context, prompt string)
(string, error) {
    var content = types.ContentBlockMemberText{
        Value: prompt,
    }
    var message = types.Message{
        Content: []types.ContentBlock{&content},
        Role:    "user",
    }
    modelId := "anthropic.claude-3-haiku-20240307-v1:0"
    var converseInput = bedrockruntime.ConverseInput{
        ModelId:  aws.String(modelId),
        Messages: []types.Message{message},
    }
    response, err := wrapper.BedrockRuntimeClient.Converse(ctx, &converseInput)
    if err != nil {
        ProcessError(err, modelId)
    }

    responseText, _ := response.Output.(*types.ConverseOutputMemberMessage)
    responseContentBlock := responseText.Value.Content[0]
    text, _ := responseContentBlock.(*types.ContentBlockMemberText)
    return text.Value, nil
}
```

- Para obtener información sobre la API, consulte [Converse](#) en la Referencia de la API de AWS SDK para Go .

InvokeModel

El siguiente ejemplo de código muestra cómo enviar un mensaje de texto a Anthropic Claude con la API de Invoke Model.

SDK para Go V2

 Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Invoke el modelo fundacional Anthropic Claude 2 para generar texto.

```
import (
    "context"
    "encoding/json"
    "log"
    "strings"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/bedrockruntime"
)

// InvokeModelWrapper encapsulates Amazon Bedrock actions used in the examples.
// It contains a Bedrock Runtime client that is used to invoke foundation models.
type InvokeModelWrapper struct {
    BedrockRuntimeClient *bedrockruntime.Client
}

// Each model provider has their own individual request and response formats.
// For the format, ranges, and default values for Anthropic Claude, refer to:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters-claude.html

type ClaudeRequest struct {
    Prompt          string    `json:"prompt"`
    MaxTokensToSample int       `json:"max_tokens_to_sample"`
    Temperature     float64   `json:"temperature,omitempty"`
    StopSequences   []string  `json:"stop_sequences,omitempty"`
}

type ClaudeResponse struct {
    Completion string `json:"completion"`
}
```

```
// Invokes Anthropic Claude on Amazon Bedrock to run an inference using the input
// provided in the request body.
func (wrapper InvokeModelWrapper) InvokeClaude(ctx context.Context, prompt string)
(string, error) {
    modelId := "anthropic.claude-v2"

    // Anthropic Claude requires enclosing the prompt as follows:
    enclosedPrompt := "Human: " + prompt + "\n\nAssistant:"

    body, err := json.Marshal(ClaudeRequest{
        Prompt:          enclosedPrompt,
        MaxTokensToSample: 200,
        Temperature:     0.5,
        StopSequences:    []string{"\n\nHuman:"},
    })

    if err != nil {
        log.Fatal("failed to marshal", err)
    }

    output, err := wrapper.BedrockRuntimeClient.InvokeModel(ctx,
        &bedrockruntime.InvokeModelInput{
            ModelId:      aws.String(modelId),
            ContentType: aws.String("application/json"),
            Body:        body,
        })

    if err != nil {
        ProcessError(err, modelId)
    }

    var response ClaudeResponse
    if err := json.Unmarshal(output.Body, &response); err != nil {
        log.Fatal("failed to unmarshal", err)
    }

    return response.Completion, nil
}
```

- Para obtener más información sobre la API, consulta [InvokeModel](#) la Referencia AWS SDK para Go de la API.

InvokeModelWithResponseStream

El siguiente ejemplos de código muestran cómo enviar un mensaje de texto a los modelos de Anthropic Claude mediante la API de Invoke Model e imprimir el flujo de respuesta.

SDK para Go V2

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](#).

Utilice la API de Invoke Model para enviar un mensaje de texto y procesar el flujo de respuesta en tiempo real.

```
import (
    "bytes"
    "context"
    "encoding/json"
    "fmt"
    "log"
    "strings"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/bedrockruntime"
    "github.com/aws/aws-sdk-go-v2/service/bedrockruntime/types"
)

// InvokeModelWithResponseStreamWrapper encapsulates Amazon Bedrock actions used in
// the examples.
// It contains a Bedrock Runtime client that is used to invoke foundation models.
type InvokeModelWithResponseStreamWrapper struct {
    BedrockRuntimeClient *bedrockruntime.Client
}
```

```

// Each model provider defines their own individual request and response formats.
// For the format, ranges, and default values for the different models, refer to:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters.html

type Request struct {
    Prompt      string `json:"prompt"`
    MaxTokensToSample int    `json:"max_tokens_to_sample"`
    Temperature float64 `json:"temperature,omitempty"`
}

type Response struct {
    Completion string `json:"completion"`
}

// Invokes Anthropic Claude on Amazon Bedrock to run an inference and asynchronously
// process the response stream.

func (wrapper InvokeModelWithResponseStreamWrapper)
    InvokeModelWithResponseStream(ctx context.Context, prompt string) (string, error) {

    modelId := "anthropic.claude-v2"

    // Anthropic Claude requires you to enclose the prompt as follows:
    prefix := "Human: "
    postfix := "\n\nAssistant:"
    prompt = prefix + prompt + postfix

    request := ClaudeRequest{
        Prompt:      prompt,
        MaxTokensToSample: 200,
        Temperature: 0.5,
        StopSequences: []string{"\n\nHuman:"},
    }

    body, err := json.Marshal(request)
    if err != nil {
        log.Panicln("Couldn't marshal the request: ", err)
    }

    output, err := wrapper.BedrockRuntimeClient.InvokeModelWithResponseStream(ctx,
        &bedrockruntime.InvokeModelWithResponseStreamInput{
            Body:      body,
            ModelId:   aws.String(modelId),

```



```

    ContentType: aws.String("application/json"),
  })

  if err != nil {
    errMsg := err.Error()
    if strings.Contains(errMsg, "no such host") {
      log.Printf("The Bedrock service is not available in the selected region. Please
double-check the service availability for your region at https://aws.amazon.com/
about-aws/global-infrastructure/regional-product-services/.\\n")
    } else if strings.Contains(errMsg, "Could not resolve the foundation model") {
      log.Printf("Could not resolve the foundation model from model identifier: \"%v
\\n. Please verify that the requested model exists and is accessible within the
specified region.\\n", modelId)
    } else {
      log.Printf("Couldn't invoke Anthropic Claude. Here's why: %v\\n", err)
    }
  }

  resp, err := processStreamingOutput(ctx, output, func(ctx context.Context, part
[]byte) error {
    fmt.Print(string(part))
    return nil
  })

  if err != nil {
    log.Fatal("streaming output processing error: ", err)
  }

  return resp.Completion, nil
}

type StreamingOutputHandler func(ctx context.Context, part []byte) error

func processStreamingOutput(ctx context.Context, output
*bedrockruntime.InvokeModelWithResponseStreamOutput, handler
StreamingOutputHandler) (Response, error) {

  var combinedResult string
  resp := Response{}

  for event := range output.GetStream().Events() {
    switch v := event.(type) {
      case *types.ResponseStreamMemberChunk:

```

```

    //fmt.Println("payload", string(v.Value.Bytes))

    var resp Response
    err := json.NewDecoder(bytes.NewReader(v.Value.Bytes)).Decode(&resp)
    if err != nil {
        return resp, err
    }

    err = handler(ctx, []byte(resp.Completion))
    if err != nil {
        return resp, err
    }

    combinedResult += resp.Completion

    case *types.UnknownUnionMember:
        fmt.Println("unknown tag:", v.Tag)

    default:
        fmt.Println("union is nil or unknown type")
    }
}

resp.Completion = combinedResult

return resp, nil
}

```

- Para obtener más información sobre la API, consulta [InvokeModelWithResponseStream](#) la Referencia AWS SDK para Go de la API.

CloudFormation ejemplos de uso de SDK for Go V2

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso de la AWS SDK para Go V2 con CloudFormation.

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

DescribeStacks

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeStacks.

SDK para Go V2

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 (
    "context"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
    CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
// structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName string)
    StackOutputs {
```

```
output, err := actor.CfnClient.DescribeStacks(ctx,
&cloudformation.DescribeStacksInput{
    StackName: aws.String(stackName),
})
if err != nil || len(output.Stacks) == 0 {
    log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
stackName, err)
}
stackOutputs := StackOutputs{}
for _, out := range output.Stacks[0].Outputs {
    stackOutputs[*out.OutputKey] = *out.OutputValue
}
return stackOutputs
}
```

- Para obtener más información sobre la API, consulta [DescribeStacks](#) la Referencia AWS SDK para Go de la API.

CloudWatch Ejemplos de registros con SDK for Go V2

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso de la AWS SDK para Go V2 con 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

StartLiveTail

En el siguiente ejemplo de código, se muestra cómo utilizar StartLiveTail.

SDK para Go V2

Incluir los archivos requeridos.

```
import (  
    "context"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/config"  
    "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"  
    "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"  
)
```

Gestione los eventos de la sesión de Live Tail.

```
func handleEventStreamAsync(stream *cloudwatchlogs.StartLiveTailEventStream) {  
    eventsChan := stream.Events()  
    for {  
        event := <-eventsChan  
        switch e := event.(type) {  
        case *types.StartLiveTailResponseStreamMemberSessionStart:  
            log.Println("Received SessionStart event")  
        case *types.StartLiveTailResponseStreamMemberSessionUpdate:  
            for _, logEvent := range e.Value.SessionResults {  
                log.Println(*logEvent.Message)  
            }  
        default:  
            // Handle on-stream exceptions  
            if err := stream.Err(); err != nil {  
                log.Fatalf("Error occurred during streaming: %v", err)  
            } else if event == nil {  
                log.Println("Stream is Closed")  
                return  
            } else {  
                log.Fatalf("Unknown event type: %T", e)  
            }  
        }  
    }  
}
```

Inicie la sesión de Live Tail.

```
cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    panic("configuration error, " + err.Error())
}
client := cloudwatchlogs.NewFromConfig(cfg)

request := &cloudwatchlogs.StartLiveTailInput{
    LogGroupIdentifiers: logGroupIdentifiers,
    LogStreamNames:      logStreamNames,
    LogEventFilterPattern: logEventFilterPattern,
}

response, err := client.StartLiveTail(context.TODO(), request)
// Handle pre-stream Exceptions
if err != nil {
    log.Fatalf("Failed to start streaming: %v", err)
}

// Start a Goroutine to handle events over stream
stream := response.GetStream()
go handleEventStreamAsync(stream)
```

Detenga la sesión de Live Tail una vez transcurrido un periodo de tiempo.

```
// Close the stream (which ends the session) after a timeout
time.Sleep(10 * time.Second)
stream.Close()
log.Println("Event stream closed")
```

- Para obtener más información sobre la API, consulte [StartLiveTail](#) la Referencia AWS SDK para Go de la API.

Ejemplos del proveedor de identidades de Amazon Cognito con SDK para Go V2

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar situaciones comunes mediante el uso de la AWS SDK para Go V2 con 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 Go V2

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](#).

```
package main
```

```
import (  
    "context"  
    "fmt"  
    "log"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/config"  
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"  
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"  
)  
  
// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification Service  
// (Amazon SNS) client and list the topics in your account.  
// This example uses the default settings specified in your shared credentials  
// and config files.  
func main() {  
    ctx := context.Background()  
    sdkConfig, err := config.LoadDefaultConfig(ctx)  
    if err != nil {  
        fmt.Println("Couldn't load default configuration. Have you set up your AWS  
account?")  
        fmt.Println(err)  
        return  
    }  
    cognitoClient := cognitoidentityprovider.NewFromConfig(sdkConfig)  
    fmt.Println("Let's list the user pools for your account.")  
    var pools []types.UserPoolDescriptionType  
    paginator := cognitoidentityprovider.NewListUserPoolsPaginator(  
        cognitoClient, &cognitoidentityprovider.ListUserPoolsInput{MaxResults:  
aws.Int32(10)})  
    for paginator.HasMorePages() {  
        output, err := paginator.NextPage(ctx)  
        if err != nil {  
            log.Printf("Couldn't get user pools. Here's why: %v\n", err)  
        } else {  
            pools = append(pools, output.UserPools...)  
        }  
    }  
    if len(pools) == 0 {  
        fmt.Println("You don't have any user pools!")  
    } else {  
        for _, pool := range pools {  
            fmt.Printf("\t%v: %v\n", *pool.Name, *pool.Id)  
        }  
    }  
}
```



```
}  
}
```

- Para obtener más información sobre la API, consulta [ListUserPools](#) la Referencia AWS SDK para Go de la API.

Acciones

AdminCreateUser

En el siguiente ejemplo de código, se muestra cómo utilizar AdminCreateUser.

SDK para Go V2

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 (  
    "context"  
    "errors"  
    "log"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"  
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"  
)  
  
type CognitoActions struct {  
    CognitoClient *cognitoidentityprovider.Client  
}  
  
// AdminCreateUser uses administrator credentials to add a user to a user pool. This  
// method leaves the user
```

```
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId string,
    userName string, userEmail string) error {
    _, err := actor.CognitoClient.AdminCreateUser(ctx,
        &cognitoidentityprovider.AdminCreateUserInput{
            UserPoolId:      aws.String(userPoolId),
            Username:        aws.String(userName),
            MessageAction:   types.MessageActionTypeSuppress,
            UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
aws.String(userEmail)}}},
        })
    if err != nil {
        var userExists *types.UsernameExistsException
        if errors.As(err, &userExists) {
            log.Printf("User %v already exists in the user pool.", userName)
            err = nil
        } else {
            log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
        }
    }
    return err
}
```

- Para obtener más información sobre la API, consulta [AdminCreateUser](#) la Referencia AWS SDK para Go de la API.

AdminSetUserPassword

En el siguiente ejemplo de código, se muestra cómo utilizar AdminSetUserPassword.

SDK para Go V2

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 (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
    CognitoClient *cognitoidentityprovider.Client
}

// AdminSetUserPassword uses administrator credentials to set a password for a user
// without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
    string, userName string, password string) error {
    _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
    &cognitoidentityprovider.AdminSetUserPasswordInput{
        Password:    aws.String(password),
        UserPoolId:  aws.String(userPoolId),
        Username:    aws.String(userName),
        Permanent:   true,
    })
    if err != nil {
        var invalidPassword *types.InvalidPasswordException
        if errors.As(err, &invalidPassword) {
            log.Println(*invalidPassword.Message)
        } else {
            log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName, err)
        }
    }
    return err
}

```

- Para obtener más información sobre la API, consulta [AdminSetUserPassword](#) la Referencia AWS SDK para Go de la API.

ConfirmForgotPassword

En el siguiente ejemplo de código, se muestra cómo utilizar ConfirmForgotPassword.

SDK para Go V2

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 (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
    CognitoClient *cognitoidentityprovider.Client
}

// ConfirmForgotPassword confirms a user with a confirmation code and a new
// password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
    string, code string, userName string, password string) error {
    _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
        &cognitoidentityprovider.ConfirmForgotPasswordInput{
            ClientId:      aws.String(clientId),
            ConfirmationCode: aws.String(code),
            Password:      aws.String(password),
            Username:      aws.String(userName),
        })
    if err != nil {
        var invalidPassword *types.InvalidPasswordException
        if errors.As(err, &invalidPassword) {
```

```
    log.Println(*invalidPassword.Message)
} else {
    log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
}
}
return err
}
```

- Para obtener más información sobre la API, consulta [ConfirmForgotPassword](#) la Referencia AWS SDK para Go de la API.

DeleteUser

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteUser.

SDK para Go V2

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 (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
    CognitoClient *cognitoidentityprovider.Client
}
```

```
// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken string)
error {
    _, err := actor.CognitoClient.DeleteUser(ctx,
        &cognitoidentityprovider.DeleteUserInput{
            AccessToken: aws.String(userAccessToken),
        })
    if err != nil {
        log.Printf("Couldn't delete user. Here's why: %v\n", err)
    }
    return err
}
```

- Para obtener más información sobre la API, consulta [DeleteUser](#) la Referencia AWS SDK para Go de la API.

ForgotPassword

En el siguiente ejemplo de código, se muestra cómo utilizar ForgotPassword.

SDK para Go V2

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 (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)
```

```
type CognitoActions struct {
    CognitoClient *cognitoidentityprovider.Client
}

// ForgotPassword starts a password recovery flow for a user. This flow typically
// sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
    userName string) (*types.CodeDeliveryDetailsType, error) {
    output, err := actor.CognitoClient.ForgotPassword(ctx,
        &cognitoidentityprovider.ForgotPasswordInput{
            ClientId: aws.String(clientId),
            Username: aws.String(userName),
        })
    if err != nil {
        log.Printf("Couldn't start password reset for user '%v'. Here's why: %v\n",
            userName, err)
    }
    return output.CodeDeliveryDetails, err
}
```

- Para obtener más información sobre la API, consulta [ForgotPassword](#) la Referencia AWS SDK para Go de la API.

InitiateAuth

En el siguiente ejemplo de código, se muestra cómo utilizar InitiateAuth.

SDK para Go V2

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 (  
    "context"  
    "errors"  
    "log"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"  
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"  
)  
  
type CognitoActions struct {  
    CognitoClient *cognitoidentityprovider.Client  
}  
  
// SignIn signs in a user to Amazon Cognito using a username and password  
authentication flow.  
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName  
string, password string) (*types.AuthenticationResultType, error) {  
    var authResult *types.AuthenticationResultType  
    output, err := actor.CognitoClient.InitiateAuth(ctx,  
    &cognitoidentityprovider.InitiateAuthInput{  
        AuthFlow:      "USER_PASSWORD_AUTH",  
        ClientId:      aws.String(clientId),  
        AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},  
    })  
    if err != nil {  
        var resetRequired *types.PasswordResetRequiredException  
        if errors.As(err, &resetRequired) {  
            log.Println(*resetRequired.Message)  
        } else {  
            log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)  
        }  
    } else {  
        authResult = output.AuthenticationResult  
    }  
    return authResult, err  
}
```


- Para obtener más información sobre la API, consulta [InitiateAuth](#) la Referencia AWS SDK para Go de la API.

ListUserPools

En el siguiente ejemplo de código, se muestra cómo utilizar ListUserPools.

SDK para Go V2

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](#).

```
package main

import (
    "context"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
        fmt.Println(err)
        return
    }
}
```

```
cognitoClient := cognitoidentityprovider.NewFromConfig(sdkConfig)
fmt.Println("Let's list the user pools for your account.")
var pools []types.UserPoolDescriptionType
paginator := cognitoidentityprovider.NewListUserPoolsPaginator(
    cognitoClient, &cognitoidentityprovider.ListUserPoolsInput{MaxResults:
aws.Int32(10)})
for paginator.HasMorePages() {
    output, err := paginator.NextPage(ctx)
    if err != nil {
        log.Printf("Couldn't get user pools. Here's why: %v\n", err)
    } else {
        pools = append(pools, output.UserPools...)
    }
}
if len(pools) == 0 {
    fmt.Println("You don't have any user pools!")
} else {
    for _, pool := range pools {
        fmt.Printf("\t\tv: %v\n", *pool.Name, *pool.Id)
    }
}
}
```

- Para obtener más información sobre la API, consulta [ListUserPools](#) la Referencia AWS SDK para Go de la API.

SignUp

En el siguiente ejemplo de código, se muestra cómo utilizar SignUp.

SDK para Go V2

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 (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
    CognitoClient *cognitoidentityprovider.Client
}

// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
string, password string, userEmail string) (bool, error) {
    confirmed := false
    output, err := actor.CognitoClient.SignUp(ctx,
    &cognitoidentityprovider.SignUpInput{
        ClientId: aws.String(clientId),
        Password: aws.String(password),
        Username: aws.String(userName),
        UserAttributes: []types.AttributeType{
            {Name: aws.String("email"), Value: aws.String(userEmail)},
        },
    })
    if err != nil {
        var invalidPassword *types.InvalidPasswordException
        if errors.As(err, &invalidPassword) {
            log.Println(*invalidPassword.Message)
        } else {
            log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
        }
    } else {
        confirmed = output.UserConfirmed
    }
    return confirmed, err
}

```

- Para obtener más información sobre la API, consulta [SignUp](#) la Referencia AWS SDK para Go de la API.

UpdateUserPool

En el siguiente ejemplo de código, se muestra cómo utilizar UpdateUserPool.

SDK para Go V2

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 (  
    "context"  
    "errors"  
    "log"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"  
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"  
)  
  
type CognitoActions struct {  
    CognitoClient *cognitoidentityprovider.Client  
}  
  
// Trigger and TriggerInfo define typed data for updating an Amazon Cognito trigger.  
type Trigger int  
  
const (  
    PreSignUp Trigger = iota  
    UserMigration  
    PostAuthentication  
)  
  
type TriggerInfo struct {
```

```

    Trigger    Trigger
    HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger is
// specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId string,
    triggers ...TriggerInfo) error {
    output, err := actor.CognitoClient.DescribeUserPool(ctx,
        &cognitoidentityprovider.DescribeUserPoolInput{
            UserPoolId: aws.String(userPoolId),
        })
    if err != nil {
        log.Printf("Couldn't get info about user pool %v. Here's why: %v\n", userPoolId,
            err)
        return err
    }
    lambdaConfig := output.UserPool.LambdaConfig
    for _, trigger := range triggers {
        switch trigger.Trigger {
        case PreSignUp:
            lambdaConfig.PreSignUp = trigger.HandlerArn
        case UserMigration:
            lambdaConfig.UserMigration = trigger.HandlerArn
        case PostAuthentication:
            lambdaConfig.PostAuthentication = trigger.HandlerArn
        }
    }
    _, err = actor.CognitoClient.UpdateUserPool(ctx,
        &cognitoidentityprovider.UpdateUserPoolInput{
            UserPoolId:  aws.String(userPoolId),
            LambdaConfig: lambdaConfig,
        })
    if err != nil {
        log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
    }
    return err
}

```

- Para obtener más información sobre la API, consulta [UpdateUserPool](#) la Referencia AWS SDK para Go de la API.

Escenarios

Confirmación de manera automática a los usuarios conocidos con una función de Lambda

En el siguiente ejemplo de código, se muestra cómo confirmar de manera automática los usuarios conocidos de Amazon Cognito con una función de Lambda.

- Configure un grupo de usuarios para que llame a una función de Lambda para el desencadenador PreSignUp.
- Inscripción de un usuario mediante Amazon Cognito
- La función de Lambda escanea una tabla de DynamoDB y confirma de manera automática los usuarios conocidos.
- Inicie sesión con el nuevo usuario y, luego, elimine los recursos.

SDK para Go V2

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](#).

Ejecutar un escenario interactivo en un símbolo del sistema.

```
import (  
    "context"  
    "errors"  
    "log"  
    "strings"  
    "user_pools_and_lambda_triggers/actions"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"  
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"  
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
```

```

)

// AutoConfirm separates the steps of this scenario into individual functions so
// that
// they are simpler to read and understand.
type AutoConfirm struct {
    helper      IScenarioHelper
    questioner  demotools.IQuestioner
    resources   Resources
    cognitoActor *actions.CognitoActions
}

// NewAutoConfirm constructs a new auto confirm runner.
func NewAutoConfirm(sdkConfig aws.Config, questioner demotools.IQuestioner, helper
IScenarioHelper) AutoConfirm {
    scenario := AutoConfirm{
        helper:      helper,
        questioner:  questioner,
        resources:   Resources{},
        cognitoActor: &actions.CognitoActions{CognitoClient:
cognitoidentityprovider.NewFromConfig(sdkConfig)},
    }
    scenario.resources.init(scenario.cognitoActor, questioner)
    return scenario
}

// AddPreSignUpTrigger adds a Lambda handler as an invocation target for the
// PreSignUp trigger.
func (runner *AutoConfirm) AddPreSignUpTrigger(ctx context.Context, userPoolId
string, functionArn string) {
    log.Printf("Let's add a Lambda function to handle the PreSignUp trigger from
Cognito.\n" +
        "This trigger happens when a user signs up, and lets your function take action
before the main Cognito\n" +
        "sign up processing occurs.\n")
    err := runner.cognitoActor.UpdateTriggers(
        ctx, userPoolId,
        actions.TriggerInfo{Trigger: actions.PreSignUp, HandlerArn:
aws.String(functionArn)})
    if err != nil {
        panic(err)
    }
    log.Printf("Lambda function %v added to user pool %v to handle the PreSignUp
trigger.\n",

```

```

    functionArn, userPoolId)
}

// SignUpUser signs up a user from the known user table with a password you specify.
func (runner *AutoConfirm) SignUpUser(ctx context.Context, clientId string,
    usersTable string) (string, string) {
    log.Println("Let's sign up a user to your Cognito user pool. When the user's email
    matches an email in the\n" +
        "DynamoDB known users table, it is automatically verified and the user is
    confirmed.")

    knownUsers, err := runner.helper.GetKnownUsers(ctx, usersTable)
    if err != nil {
        panic(err)
    }
    userChoice := runner.questioner.AskChoice("Which user do you want to use?\n",
        knownUsers.UserNameList())
    user := knownUsers.Users[userChoice]

    var signedUp bool
    var userConfirmed bool
    password := runner.questioner.AskPassword("Enter a password that has at least eight
    characters, uppercase, lowercase, numbers and symbols.\n"+
        "(the password will not display as you type):", 8)
    for !signedUp {
        log.Printf("Signing up user '%v' with email '%v' to Cognito.\n", user.UserName,
            user.UserEmail)
        userConfirmed, err = runner.cognitoActor.SignUp(ctx, clientId, user.UserName,
            password, user.UserEmail)
        if err != nil {
            var invalidPassword *types.InvalidPasswordException
            if errors.As(err, &invalidPassword) {
                password = runner.questioner.AskPassword("Enter another password:", 8)
            } else {
                panic(err)
            }
        } else {
            signedUp = true
        }
    }
    log.Printf("User %v signed up, confirmed = %v.\n", user.UserName, userConfirmed)

    log.Println(strings.Repeat("-", 88))
}

```



```

    return user.UserName, password
}

// SignInUser signs in a user.
func (runner *AutoConfirm) SignInUser(ctx context.Context, clientId string, userName
    string, password string) string {
    runner.questioner.Ask("Press Enter when you're ready to continue.")
    log.Printf("Let's sign in as %v...\n", userName)
    authResult, err := runner.cognitoActor.SignIn(ctx, clientId, userName, password)
    if err != nil {
        panic(err)
    }
    log.Printf("Successfully signed in. Your access token starts with: %v...\n",
        (*authResult.AccessToken)[:10])
    log.Println(strings.Repeat("-", 88))
    return *authResult.AccessToken
}

// Run runs the scenario.
func (runner *AutoConfirm) Run(ctx context.Context, stackName string) {
    defer func() {
        if r := recover(); r != nil {
            log.Println("Something went wrong with the demo.")
            runner.resources.Cleanup(ctx)
        }
    }()

    log.Println(strings.Repeat("-", 88))
    log.Printf("Welcome\n")

    log.Println(strings.Repeat("-", 88))

    stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
    if err != nil {
        panic(err)
    }
    runner.resources.userPoolId = stackOutputs["UserPoolId"]
    runner.helper.PopulateUserTable(ctx, stackOutputs["TableName"])

    runner.AddPreSignUpTrigger(ctx, stackOutputs["UserPoolId"],
        stackOutputs["AutoConfirmFunctionArn"])
    runner.resources.triggers = append(runner.resources.triggers, actions.PreSignUp)
    userName, password := runner.SignUpUser(ctx, stackOutputs["UserPoolClientId"],
        stackOutputs["TableName"])

```

```

runner.helper.ListRecentLogEvents(ctx, stackOutputs["AutoConfirmFunction"])
runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
    runner.SignInUser(ctx, stackOutputs["UserPoolClientId"], userName, password))

runner.resources.Cleanup(ctx)

log.Println(strings.Repeat("-", 88))
log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}

```

Controle el desencadenador PreSignUp con una función de Lambda.

```

import (
    "context"
    "log"
    "os"

    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
    dynamodbtypes "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
// format.
type UserInfo struct {
    UserName  string `dynamodbav:"UserName"`
    UserEmail string `dynamodbav:"UserEmail"`
}

// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
    userEmail, err := attributevalue.Marshal(user.UserEmail)
    if err != nil {

```

```

    panic(err)
}
return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
    dynamoClient *dynamodb.Client
}

// HandleRequest handles the PreSignUp event by looking up a user in an Amazon
// DynamoDB table and
// specifying whether they should be confirmed and verified.
func (h *handler) HandleRequest(ctx context.Context, event
events.CognitoEventUserPoolsPreSignup) (events.CognitoEventUserPoolsPreSignup,
error) {
    log.Printf("Received presignup from %v for user '%v'", event.TriggerSource,
event.UserName)
    if event.TriggerSource != "PreSignUp_SignUp" {
        // Other trigger sources, such as PreSignUp_AdminInitiateAuth, ignore the response
        from this handler.
        return event, nil
    }
    tableName := os.Getenv(TABLE_NAME)
    user := UserInfo{
        UserEmail: event.Request.UserAttributes["email"],
    }
    log.Printf("Looking up email %v in table %v.\n", user.UserEmail, tableName)
    output, err := h.dynamoClient.GetItem(ctx, &dynamodb.GetItemInput{
        Key:      user.GetKey(),
        TableName: aws.String(tableName),
    })
    if err != nil {
        log.Printf("Error looking up email %v.\n", user.UserEmail)
        return event, err
    }
    if output.Item == nil {
        log.Printf("Email %v not found. Email verification is required.\n",
user.UserEmail)
        return event, err
    }

    err = attributevalue.UnmarshalMap(output.Item, &user)
    if err != nil {
        log.Printf("Couldn't unmarshal DynamoDB item. Here's why: %v\n", err)
    }
}

```

```

    return event, err
}

if user.UserName != event.UserName {
    log.Printf("UserEmail %v found, but stored UserName '%v' does not match supplied
    UserName '%v'. Verification is required.\n",
        user.UserEmail, user.UserName, event.UserName)
} else {
    log.Printf("UserEmail %v found with matching UserName %v. User is confirmed.\n",
        user.UserEmail, user.UserName)
    event.Response.AutoConfirmUser = true
    event.Response.AutoVerifyEmail = true
}

return event, err
}

func main() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        log.Panicln(err)
    }
    h := handler{
        dynamoClient: dynamodb.NewFromConfig(sdkConfig),
    }
    lambda.Start(h.HandleRequest)
}

```

Cree una estructura que lleve a cabo las tareas habituales.

```

import (
    "context"
    "log"
    "strings"
    "time"
    "user_pools_and_lambda_triggers/actions"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cloudformation"

```

```

"github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
"github.com/aws/aws-sdk-go-v2/service/dynamodb"
"github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this example.
type IScenarioHelper interface {
    Pause(secs int)
    GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
        error)
    PopulateUserTable(ctx context.Context, tableName string)
    GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
    AddKnownUser(ctx context.Context, tableName string, user actions.User)
    ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
// example.
type ScenarioHelper struct {
    questioner demotools.IQuestioner
    dynamoActor *actions.DynamoActions
    cfnActor     *actions.CloudFormationActions
    cwlActor     *actions.CloudWatchLogsActions
    isTestRun    bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
    ScenarioHelper {
    scenario := ScenarioHelper{
        questioner: questioner,
        dynamoActor: &actions.DynamoActions{DynamoClient:
            dynamodb.NewFromConfig(sdkConfig)},
        cfnActor:     &actions.CloudFormationActions{CfnClient:
            cloudformation.NewFromConfig(sdkConfig)},
        cwlActor:     &actions.CloudWatchLogsActions{CwlClient:
            cloudwatchlogs.NewFromConfig(sdkConfig)},
    }
    return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
    if !helper.isTestRun {

```

```

    time.Sleep(time.Duration(secs) * time.Second)
}
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
// structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName string)
(actions.StackOutputs, error) {
    return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
string) {
    log.Printf("First, let's add some users to the DynamoDB %v table we'll use for this
example.\n", tableName)
    err := helper.dynamoActor.PopulateTable(ctx, tableName)
    if err != nil {
        panic(err)
    }
}

// GetKnownUsers gets the users from the known users table in a structured format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
(actions.UserList, error) {
    knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
    if err != nil {
        log.Printf("Couldn't get known users from table %v. Here's why: %v\n", tableName,
err)
    }
    return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
user actions.User) {
    log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users table...\n",
user.UserName, user.UserEmail)
    err := helper.dynamoActor.AddUser(ctx, tableName, user)
    if err != nil {
        panic(err)
    }
}

```

```
// ListRecentLogEvents gets the most recent log stream and events for the specified
// Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context, functionName
string) {
    log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
    helper.Pause(10)
    log.Println("Okay, let's check the logs to find what's happened recently with your
    Lambda function.")
    logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
    if err != nil {
        panic(err)
    }
    log.Printf("Getting some recent events from log stream %v\n",
        *logStream.LogStreamName)
    events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
        *logStream.LogStreamName, 10)
    if err != nil {
        panic(err)
    }
    for _, event := range events {
        log.Printf("\t\t%v", *event.Message)
    }
    log.Println(strings.Repeat("-", 88))
}
```

Cree una estructura que ajuste las acciones de Amazon Cognito.

```
import (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
    CognitoClient *cognitoidentityprovider.Client
```

```

}

// Trigger and TriggerInfo define typed data for updating an Amazon Cognito trigger.
type Trigger int

const (
    PreSignUp Trigger = iota
    UserMigration
    PostAuthentication
)

type TriggerInfo struct {
    Trigger    Trigger
    HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger is
// specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId string,
    triggers ...TriggerInfo) error {
    output, err := actor.CognitoClient.DescribeUserPool(ctx,
        &cognitoidentityprovider.DescribeUserPoolInput{
            UserPoolId: aws.String(userPoolId),
        })
    if err != nil {
        log.Printf("Couldn't get info about user pool %v. Here's why: %v\n", userPoolId,
            err)
        return err
    }
    lambdaConfig := output.UserPool.LambdaConfig
    for _, trigger := range triggers {
        switch trigger.Trigger {
        case PreSignUp:
            lambdaConfig.PreSignUp = trigger.HandlerArn
        case UserMigration:
            lambdaConfig.UserMigration = trigger.HandlerArn
        case PostAuthentication:
            lambdaConfig.PostAuthentication = trigger.HandlerArn
        }
    }
}

```



```

_, err = actor.CognitoClient.UpdateUserPool(ctx,
&cognitoidentityprovider.UpdateUserPoolInput{
    UserPoolId:    aws.String(userPoolId),
    LambdaConfig: lambdaConfig,
})
if err != nil {
    log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
}
return err
}

// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
string, password string, userEmail string) (bool, error) {
    confirmed := false
    output, err := actor.CognitoClient.SignUp(ctx,
&cognitoidentityprovider.SignUpInput{
    ClientId: aws.String(clientId),
    Password: aws.String(password),
    Username: aws.String(userName),
    UserAttributes: []types.AttributeType{
        {Name: aws.String("email"), Value: aws.String(userEmail)},
    },
})
if err != nil {
    var invalidPassword *types.InvalidPasswordException
    if errors.As(err, &invalidPassword) {
        log.Println(*invalidPassword.Message)
    } else {
        log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
    }
} else {
    confirmed = output.UserConfirmed
}
return confirmed, err
}

// SignIn signs in a user to Amazon Cognito using a username and password
authentication flow.

```

```

func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
string, password string) (*types.AuthenticationResultType, error) {
    var authResult *types.AuthenticationResultType
    output, err := actor.CognitoClient.InitiateAuth(ctx,
    &cognitoidentityprovider.InitiateAuthInput{
        AuthFlow:      "USER_PASSWORD_AUTH",
        ClientId:      aws.String(clientId),
        AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
    })
    if err != nil {
        var resetRequired *types.PasswordResetRequiredException
        if errors.As(err, &resetRequired) {
            log.Println(*resetRequired.Message)
        } else {
            log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
        }
    } else {
        authResult = output.AuthenticationResult
    }
    return authResult, err
}

// ForgotPassword starts a password recovery flow for a user. This flow typically
// sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
userName string) (*types.CodeDeliveryDetailsType, error) {
    output, err := actor.CognitoClient.ForgotPassword(ctx,
    &cognitoidentityprovider.ForgotPasswordInput{
        ClientId: aws.String(clientId),
        Username: aws.String(userName),
    })
    if err != nil {
        log.Printf("Couldn't start password reset for user '%v'. Here's why: %v\n",
        userName, err)
    }
    return output.CodeDeliveryDetails, err
}

```

```

// ConfirmForgotPassword confirms a user with a confirmation code and a new
password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
string, code string, userName string, password string) error {
_, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
&cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:      aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:      aws.String(password),
  Username:      aws.String(userName),
})
if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
    log.Println(*invalidPassword.Message)
  } else {
    log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
}
return err
}

// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken string)
error {
_, err := actor.CognitoClient.DeleteUser(ctx,
&cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
})
if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
}
return err
}

// AdminCreateUser uses administrator credentials to add a user to a user pool. This
method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId string,
userName string, userEmail string) error {

```

```

_, err := actor.CognitoClient.AdminCreateUser(ctx,
&cognitoidentityprovider.AdminCreateUserInput{
    UserPoolId:    aws.String(userPoolId),
    Username:      aws.String(userName),
    MessageAction: types.MessageActionTypeSuppress,
    UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
aws.String(userEmail)}}},
)})
if err != nil {
    var userExists *types.UsernameExistsException
    if errors.As(err, &userExists) {
        log.Printf("User %v already exists in the user pool.", userName)
        err = nil
    } else {
        log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
    }
}
return err
}

// AdminSetUserPassword uses administrator credentials to set a password for a user
// without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
string, userName string, password string) error {
_, err := actor.CognitoClient.AdminSetUserPassword(ctx,
&cognitoidentityprovider.AdminSetUserPasswordInput{
    Password:    aws.String(password),
    UserPoolId:  aws.String(userPoolId),
    Username:    aws.String(userName),
    Permanent:   true,
})
if err != nil {
    var invalidPassword *types.InvalidPasswordException
    if errors.As(err, &invalidPassword) {
        log.Println(*invalidPassword.Message)
    } else {
        log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName, err)
    }
}
return err
}

```

Cree una estructura que ajuste las acciones de DynamoDB.

```
import (  
    "context"  
    "fmt"  
    "log"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"  
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"  
    "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"  
)  
  
// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)  
// actions  
// used in the examples.  
type DynamoActions struct {  
    DynamoClient *dynamodb.Client  
}  
  
// User defines structured user data.  
type User struct {  
    Username string  
    UserEmail string  
    LastLogin *LoginInfo `dynamodbav:",omitempty"`  
}  
  
// LoginInfo defines structured custom login data.  
type LoginInfo struct {  
    UserPoolId string  
    ClientId string  
    Time string  
}  
  
// UserList defines a list of users.  
type UserList struct {  
    Users []User  
}
```

```

// UserNameList returns the usernames contained in a UserList as a list of strings.
func (users *UserList) UserNameList() []string {
    names := make([]string, len(users.Users))
    for i := 0; i < len(users.Users); i++ {
        names[i] = users.Users[i].UserName
    }
    return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
    error {
    var err error
    var item map[string]types.AttributeValue
    var writeReqs []types.WriteRequest
    for i := 1; i < 4; i++ {
        item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_%v",
i), userEmail: fmt.Sprintf("test_email_%v@example.com", i)})
        if err != nil {
            log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n", err)
            return err
        }
        writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
&types.PutRequest{Item: item}})
    }
    _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
})
    if err != nil {
        log.Printf("Couldn't populate table %v with users. Here's why: %v\n", tableName,
err)
    }
    return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
    error) {
    var userList UserList
    output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
TableName: aws.String(tableName),
})
    if err != nil {
        log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName, err)
    }
}

```

```

} else {
    err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
    if err != nil {
        log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
    }
}
return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user User)
error {
    userItem, err := attributevalue.MarshalMap(user)
    if err != nil {
        log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
    }
    _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
        Item:      userItem,
        TableName: aws.String(tableName),
    })
    if err != nil {
        log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
    }
    return err
}

```

Crea una estructura que agrupe las acciones de CloudWatch Logs.

```

import (
    "context"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
    "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

type CloudWatchLogsActions struct {
    CwlClient *cloudwatchlogs.Client
}

```

```

}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
    functionName string) (types.LogStream, error) {
    var logStream types.LogStream
    logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
    output, err := actor.CwlClient.DescribeLogStreams(ctx,
        &cloudwatchlogs.DescribeLogStreamsInput{
            Descending:    aws.Bool(true),
            Limit:         aws.Int32(1),
            LogGroupName: aws.String(logGroupName),
            OrderBy:      types.OrderByLastEventTime,
        })
    if err != nil {
        log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
            logGroupName, err)
    } else {
        logStream = output.LogStreams[0]
    }
    return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
// stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
    string, logStreamName string, eventCount int32) (
    []types.OutputLogEvent, error) {
    var events []types.OutputLogEvent
    logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
    output, err := actor.CwlClient.GetLogEvents(ctx, &cloudwatchlogs.GetLogEventsInput{
        LogStreamName: aws.String(logStreamName),
        Limit:         aws.Int32(eventCount),
        LogGroupName:  aws.String(logGroupName),
    })
    if err != nil {
        log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
            logStreamName, err)
    } else {
        events = output.Events
    }
    return events, err
}

```


Crea una estructura que agrupe las acciones. CloudFormation

```
import (
    "context"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
    CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
// structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName string)
    StackOutputs {
    output, err := actor.CfnClient.DescribeStacks(ctx,
        &cloudformation.DescribeStacksInput{
            StackName: aws.String(stackName),
        })
    if err != nil || len(output.Stacks) == 0 {
        log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
            stackName, err)
    }
    stackOutputs := StackOutputs{}
    for _, out := range output.Stacks[0].Outputs {
        stackOutputs[*out.OutputKey] = *out.OutputValue
    }
    return stackOutputs
}
```

Eliminación de recursos.

```

import (
    "context"
    "log"
    "user_pools_and_lambda_triggers/actions"

    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
    userPoolId      string
    userAccessTokens []string
    triggers        []actions.Trigger

    cognitoActor *actions.CognitoActions
    questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
demotools.IQuestioner) {
    resources.userAccessTokens = []string{}
    resources.triggers = []actions.Trigger{}
    resources.cognitoActor = cognitoActor
    resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
    defer func() {
        if r := recover(); r != nil {
            log.Printf("Something went wrong during cleanup.\n%v\n", r)
            log.Println("Use the AWS Management Console to remove any remaining resources \n"
+
            "that were created for this scenario.")
        }
    }()

    wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
resources that were created "+
    "during this demo (y/n)?", "y")
    if wantDelete {

```

```

for _, accessToken := range resources.userAccessTokens {
    err := resources.cognitoActor.DeleteUser(ctx, accessToken)
    if err != nil {
        log.Println("Couldn't delete user during cleanup.")
        panic(err)
    }
    log.Println("Deleted user.")
}
triggerList := make([]actions.TriggerInfo, len(resources.triggers))
for i := 0; i < len(resources.triggers); i++ {
    triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i], HandlerArn:
nil}
}
err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
triggerList...)
if err != nil {
    log.Println("Couldn't update Cognito triggers during cleanup.")
    panic(err)
}
log.Println("Removed Cognito triggers from user pool.")
} else {
    log.Println("Be sure to remove resources when you're done with them to avoid
unexpected charges!")
}
}

```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Go .
 - [DeleteUser](#)
 - [InitiateAuth](#)
 - [SignUp](#)
 - [UpdateUserPool](#)

Migración en forma automática los usuarios conocidos con una función de Lambda

En el siguiente ejemplo de código, se muestra cómo migrar de manera automática los usuarios conocidos de Amazon Cognito con una función de Lambda.

- Configure un grupo de usuarios para que llame a una función de Lambda para el desencadenador `MigrateUser`.
- Inicie sesión en Amazon Cognito con un nombre de usuario y un correo electrónico que no estén en el grupo de usuarios.
- La función de Lambda escanea una tabla de DynamoDB y migra de manera automática los usuarios conocidos al grupo de usuarios.
- Realice el flujo en caso de olvido de contraseña para restablecer la contraseña respecto del usuario migrado.
- Inicie sesión como un nuevo usuario y, a continuación, elimine los recursos.

SDK para Go V2

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](#).

Ejecutar un escenario interactivo en un símbolo del sistema.

```
import (  
    "context"  
    "errors"  
    "fmt"  
    "log"  
    "strings"  
    "user_pools_and_lambda_triggers/actions"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"  
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"  
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"  
)  
  
// MigrateUser separates the steps of this scenario into individual functions so  
// that  
// they are simpler to read and understand.  
type MigrateUser struct {
```

```

helper      IScenarioHelper
questioner  demotools.IQuestioner
resources    Resources
cognitoActor *actions.CognitoActions
}

// NewMigrateUser constructs a new migrate user runner.
func NewMigrateUser(sdkConfig aws.Config, questioner demotools.IQuestioner, helper
IScenarioHelper) MigrateUser {
scenario := MigrateUser{
    helper:      helper,
    questioner:  questioner,
    resources:    Resources{},
    cognitoActor: &actions.CognitoActions{CognitoClient:
cognitoidentityprovider.NewFromConfig(sdkConfig)},
}
scenario.resources.init(scenario.cognitoActor, questioner)
return scenario
}

// AddMigrateUserTrigger adds a Lambda handler as an invocation target for the
MigrateUser trigger.
func (runner *MigrateUser) AddMigrateUserTrigger(ctx context.Context, userPoolId
string, functionArn string) {
log.Printf("Let's add a Lambda function to handle the MigrateUser trigger from
Cognito.\n" +
    "This trigger happens when an unknown user signs in, and lets your function take
action before Cognito\n" +
    "rejects the user.\n\n")
err := runner.cognitoActor.UpdateTriggers(
    ctx, userPoolId,
    actions.TriggerInfo{Trigger: actions.UserMigration, HandlerArn:
aws.String(functionArn)})
if err != nil {
    panic(err)
}
log.Printf("Lambda function %v added to user pool %v to handle the MigrateUser
trigger.\n",
    functionArn, userPoolId)

log.Println(strings.Repeat("-", 88))
}

```

```
// SignInUser adds a new user to the known users table and signs that user in to
// Amazon Cognito.
func (runner *MigrateUser) SignInUser(ctx context.Context, usersTable string,
    clientId string) (bool, actions.User) {
    log.Println("Let's sign in a user to your Cognito user pool. When the username and
    email matches an entry in the\n" +
        "DynamoDB known users table, the email is automatically verified and the user is
    migrated to the Cognito user pool.")

    user := actions.User{}
    user.UserName = runner.questioner.Ask("\nEnter a username:")
    user.UserEmail = runner.questioner.Ask("\nEnter an email that you own. This email
    will be used to confirm user migration\n" +
        "during this example:")

    runner.helper.AddKnownUser(ctx, usersTable, user)

    var err error
    var resetRequired *types.PasswordResetRequiredException
    var authResult *types.AuthenticationResultType
    signedIn := false
    for !signedIn && resetRequired == nil {
        log.Printf("Signing in to Cognito as user '%v'. The expected result is a
        PasswordResetRequiredException.\n\n", user.UserName)
        authResult, err = runner.cognitoActor.SignIn(ctx, clientId, user.UserName, "_")
        if err != nil {
            if errors.As(err, &resetRequired) {
                log.Printf("\nUser '%v' is not in the Cognito user pool but was found in the
                DynamoDB known users table.\n"+
                    "User migration is started and a password reset is required.", user.UserName)
            } else {
                panic(err)
            }
        } else {
            log.Printf("User '%v' successfully signed in. This is unexpected and probably
            means you have not\n"+
                "cleaned up a previous run of this scenario, so the user exist in the Cognito
            user pool.\n"+
                "You can continue this example and select to clean up resources, or manually
            remove\n"+
                "the user from your user pool and try again.", user.UserName)
            runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
                *authResult.AccessToken)
            signedIn = true
        }
    }
}
```

```

    }
}

log.Println(strings.Repeat("-", 88))
return resetRequired != nil, user
}

// ResetPassword starts a password recovery flow.
func (runner *MigrateUser) ResetPassword(ctx context.Context, clientId string, user
actions.User) {
    wantCode := runner.questioner.AskBool(fmt.Sprintf("In order to migrate the user to
Cognito, you must be able to receive a confirmation\n"+
    "code by email at %v. Do you want to send a code (y/n)?", user.UserEmail), "y")
    if !wantCode {
        log.Println("To complete this example and successfully migrate a user to Cognito,
you must enter an email\n" +
        "you own that can receive a confirmation code.")
        return
    }
    codeDelivery, err := runner.cognitoActor.ForgotPassword(ctx, clientId,
user.UserName)
    if err != nil {
        panic(err)
    }
    log.Printf("\nA confirmation code has been sent to %v.", *codeDelivery.Destination)
    code := runner.questioner.Ask("Check your email and enter it here:")

    confirmed := false
    password := runner.questioner.AskPassword("\nEnter a password that has at least
eight characters, uppercase, lowercase, numbers and symbols.\n"+
    "(the password will not display as you type):", 8)
    for !confirmed {
        log.Printf("\nConfirming password reset for user '%v'.\n", user.UserName)
        err = runner.cognitoActor.ConfirmForgotPassword(ctx, clientId, code,
user.UserName, password)
        if err != nil {
            var invalidPassword *types.InvalidPasswordException
            if errors.As(err, &invalidPassword) {
                password = runner.questioner.AskPassword("\nEnter another password:", 8)
            } else {
                panic(err)
            }
        } else {
            confirmed = true

```

```

    }
}
log.Printf("User '%v' successfully confirmed and migrated.\n", user.UserName)
log.Println("Signing in with your username and password...")
authResult, err := runner.cognitoActor.SignIn(ctx, clientId, user.UserName,
password)
if err != nil {
    panic(err)
}
log.Printf("Successfully signed in. Your access token starts with: %v...\n",
(*authResult.AccessToken)[:10])
runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
*authResult.AccessToken)

log.Println(strings.Repeat("-", 88))
}

// Run runs the scenario.
func (runner *MigrateUser) Run(ctx context.Context, stackName string) {
defer func() {
    if r := recover(); r != nil {
        log.Println("Something went wrong with the demo.")
        runner.resources.Cleanup(ctx)
    }
}()

log.Println(strings.Repeat("-", 88))
log.Printf("Welcome\n")

log.Println(strings.Repeat("-", 88))

stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
if err != nil {
    panic(err)
}
runner.resources.userPoolId = stackOutputs["UserPoolId"]

runner.AddMigrateUserTrigger(ctx, stackOutputs["UserPoolId"],
stackOutputs["MigrateUserFunctionArn"])
runner.resources.triggers = append(runner.resources.triggers,
actions.UserMigration)
resetNeeded, user := runner.SignInUser(ctx, stackOutputs["TableName"],
stackOutputs["UserPoolClientId"])
if resetNeeded {

```



```

    runner.helper.ListRecentLogEvents(ctx, stackOutputs["MigrateUserFunction"])
    runner.ResetPassword(ctx, stackOutputs["UserPoolClientId"], user)
}

runner.resources.Cleanup(ctx)

log.Println(strings.Repeat("-", 88))
log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}

```

Controle el desencadenador `MigrateUser` con una función de Lambda.

```

import (
    "context"
    "log"
    "os"

    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
)

const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
// format.
type UserInfo struct {
    UserName  string `dynamodbav:"UserName"`
    UserEmail string `dynamodbav:"UserEmail"`
}

type handler struct {
    dynamoClient *dynamodb.Client
}

```

```
// HandleRequest handles the MigrateUser event by looking up a user in an Amazon
// DynamoDB table and
// specifying whether they should be migrated to the user pool.
func (h *handler) HandleRequest(ctx context.Context, event
events.CognitoEventUserPoolsMigrateUser) (events.CognitoEventUserPoolsMigrateUser,
error) {
    log.Printf("Received migrate trigger from %v for user '%v'", event.TriggerSource,
event.UserName)
    if event.TriggerSource != "UserMigration_Authentication" {
        return event, nil
    }
    tableName := os.Getenv(TABLE_NAME)
    user := UserInfo{
        UserName: event.UserName,
    }
    log.Printf("Looking up user '%v' in table %v.\n", user.UserName, tableName)
    filterEx := expression.Name("UserName").Equal(expression.Value(user.UserName))
    expr, err := expression.NewBuilder().WithFilter(filterEx).Build()
    if err != nil {
        log.Printf("Error building expression to query for user '%v'.\n", user.UserName)
        return event, err
    }
    output, err := h.dynamoClient.Scan(ctx, &dynamodb.ScanInput{
        TableName:          aws.String(tableName),
        FilterExpression:    expr.Filter(),
        ExpressionAttributeNames: expr.Names(),
        ExpressionAttributeValues: expr.Values(),
    })
    if err != nil {
        log.Printf("Error looking up user '%v'.\n", user.UserName)
        return event, err
    }
    if len(output.Items) == 0 {
        log.Printf("User '%v' not found, not migrating user.\n", user.UserName)
        return event, err
    }

    var users []UserInfo
    err = attributevalue.UnmarshalListOfMaps(output.Items, &users)
    if err != nil {
        log.Printf("Couldn't unmarshal DynamoDB items. Here's why: %v\n", err)
        return event, err
    }
}
```

```

user = users[0]
log.Printf("UserName '%v' found with email %v. User is migrated and must reset
password.\n", user.UserName, user.UserEmail)
event.CognitoEventUserPoolsMigrateUserResponse.UserAttributes = map[string]string{
    "email":          user.UserEmail,
    "email_verified": "true", // email_verified is required for the forgot password
flow.
}
event.CognitoEventUserPoolsMigrateUserResponse.FinalUserStatus = "RESET_REQUIRED"
event.CognitoEventUserPoolsMigrateUserResponse.MessageAction = "SUPPRESS"

return event, err
}

func main() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        log.Panicln(err)
    }
    h := handler{
        dynamoClient: dynamodb.NewFromConfig(sdkConfig),
    }
    lambda.Start(h.HandleRequest)
}

```

Cree una estructura que lleve a cabo las tareas habituales.

```

import (
    "context"
    "log"
    "strings"
    "time"
    "user_pools_and_lambda_triggers/actions"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cloudformation"
    "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"

```

```

)

// IScenarioHelper defines common functions used by the workflows in this example.
type IScenarioHelper interface {
    Pause(secs int)
    GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
        error)
    PopulateUserTable(ctx context.Context, tableName string)
    GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
    AddKnownUser(ctx context.Context, tableName string, user actions.User)
    ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
// example.
type ScenarioHelper struct {
    questioner demotools.IQuestioner
    dynamoActor *actions.DynamoActions
    cfnActor     *actions.CloudFormationActions
    cwlActor     *actions.CloudWatchLogsActions
    isTestRun    bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
    ScenarioHelper {
    scenario := ScenarioHelper{
        questioner: questioner,
        dynamoActor: &actions.DynamoActions{DynamoClient:
            dynamodb.NewFromConfig(sdkConfig)},
        cfnActor:     &actions.CloudFormationActions{CfnClient:
            cloudformation.NewFromConfig(sdkConfig)},
        cwlActor:     &actions.CloudWatchLogsActions{CwlClient:
            cloudwatchlogs.NewFromConfig(sdkConfig)},
    }
    return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
    if !helper.isTestRun {
        time.Sleep(time.Duration(secs) * time.Second)
    }
}

```

```

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
// structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName string)
(actions.StackOutputs, error) {
    return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
string) {
    log.Printf("First, let's add some users to the DynamoDB %v table we'll use for this
example.\n", tableName)
    err := helper.dynamoActor.PopulateTable(ctx, tableName)
    if err != nil {
        panic(err)
    }
}

// GetKnownUsers gets the users from the known users table in a structured format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
(actions.UserList, error) {
    knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
    if err != nil {
        log.Printf("Couldn't get known users from table %v. Here's why: %v\n", tableName,
err)
    }
    return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
user actions.User) {
    log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users table...
\n",
        user.UserName, user.UserEmail)
    err := helper.dynamoActor.AddUser(ctx, tableName, user)
    if err != nil {
        panic(err)
    }
}

// ListRecentLogEvents gets the most recent log stream and events for the specified
// Lambda function and displays them.

```

```

func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context, functionName
string) {
    log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
    helper.Pause(10)
    log.Println("Okay, let's check the logs to find what's happened recently with your
Lambda function.")
    logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
    if err != nil {
        panic(err)
    }
    log.Printf("Getting some recent events from log stream %v\n",
*logStream.LogStreamName)
    events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
*logStream.LogStreamName, 10)
    if err != nil {
        panic(err)
    }
    for _, event := range events {
        log.Printf("\t\t%v", *event.Message)
    }
    log.Println(strings.Repeat("-", 88))
}

```

Cree una estructura que ajuste las acciones de Amazon Cognito.

```

import (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
    CognitoClient *cognitoidentityprovider.Client
}

```

```

// Trigger and TriggerInfo define typed data for updating an Amazon Cognito trigger.
type Trigger int

const (
    PreSignUp Trigger = iota
    UserMigration
    PostAuthentication
)

type TriggerInfo struct {
    Trigger    Trigger
    HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger is
// specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId string,
    triggers ...TriggerInfo) error {
    output, err := actor.CognitoClient.DescribeUserPool(ctx,
        &cognitoidentityprovider.DescribeUserPoolInput{
            UserPoolId: aws.String(userPoolId),
        })
    if err != nil {
        log.Printf("Couldn't get info about user pool %v. Here's why: %v\n", userPoolId,
            err)
        return err
    }
    lambdaConfig := output.UserPool.LambdaConfig
    for _, trigger := range triggers {
        switch trigger.Trigger {
        case PreSignUp:
            lambdaConfig.PreSignUp = trigger.HandlerArn
        case UserMigration:
            lambdaConfig.UserMigration = trigger.HandlerArn
        case PostAuthentication:
            lambdaConfig.PostAuthentication = trigger.HandlerArn
        }
    }
    _, err = actor.CognitoClient.UpdateUserPool(ctx,
        &cognitoidentityprovider.UpdateUserPoolInput{
            UserPoolId:    aws.String(userPoolId),
            LambdaConfig: lambdaConfig,
        })
}

```

```

    })
    if err != nil {
        log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
    }
    return err
}

// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
string, password string, userEmail string) (bool, error) {
    confirmed := false
    output, err := actor.CognitoClient.SignUp(ctx,
    &cognitoidentityprovider.SignUpInput{
        ClientId: aws.String(clientId),
        Password: aws.String(password),
        Username: aws.String(userName),
        UserAttributes: []types.AttributeType{
            {Name: aws.String("email"), Value: aws.String(userEmail)},
        },
    })
    if err != nil {
        var invalidPassword *types.InvalidPasswordException
        if errors.As(err, &invalidPassword) {
            log.Println(*invalidPassword.Message)
        } else {
            log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
        }
    } else {
        confirmed = output.UserConfirmed
    }
    return confirmed, err
}

// SignIn signs in a user to Amazon Cognito using a username and password
authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
string, password string) (*types.AuthenticationResultType, error) {
    var authResult *types.AuthenticationResultType
    output, err := actor.CognitoClient.InitiateAuth(ctx,
    &cognitoidentityprovider.InitiateAuthInput{

```



```

    AuthFlow:      "USER_PASSWORD_AUTH",
    ClientId:      aws.String(clientId),
    AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
  })
  if err != nil {
    var resetRequired *types.PasswordResetRequiredException
    if errors.As(err, &resetRequired) {
      log.Println(*resetRequired.Message)
    } else {
      log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
    }
  } else {
    authResult = output.AuthenticationResult
  }
  return authResult, err
}

```

// ForgotPassword starts a password recovery flow for a user. This flow typically sends a confirmation code

// to the user's configured notification destination, such as email.

```

func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
  userName string) (*types.CodeDeliveryDetailsType, error) {
  output, err := actor.CognitoClient.ForgotPassword(ctx,
    &cognitoidentityprovider.ForgotPasswordInput{
      ClientId: aws.String(clientId),
      Username: aws.String(userName),
    })
  if err != nil {
    log.Printf("Couldn't start password reset for user '%v'. Here's why: %v\n",
      userName, err)
  }
  return output.CodeDeliveryDetails, err
}

```

// ConfirmForgotPassword confirms a user with a confirmation code and a new password.

```

func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
  string, code string, userName string, password string) error {
  _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
    &cognitoidentityprovider.ConfirmForgotPasswordInput{

```

```

    ClientId:      aws.String(clientId),
    ConfirmationCode: aws.String(code),
    Password:      aws.String(password),
    Username:      aws.String(userName),
  })
  if err != nil {
    var invalidPassword *types.InvalidPasswordException
    if errors.As(err, &invalidPassword) {
      log.Println(*invalidPassword.Message)
    } else {
      log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
    }
  }
  return err
}

// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken string)
  error {
  _, err := actor.CognitoClient.DeleteUser(ctx,
    &cognitoidentityprovider.DeleteUserInput{
      AccessToken: aws.String(userAccessToken),
    })
  if err != nil {
    log.Printf("Couldn't delete user. Here's why: %v\n", err)
  }
  return err
}

// AdminCreateUser uses administrator credentials to add a user to a user pool. This
  method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId string,
  userName string, userEmail string) error {
  _, err := actor.CognitoClient.AdminCreateUser(ctx,
    &cognitoidentityprovider.AdminCreateUserInput{
      UserPoolId:      aws.String(userPoolId),
      Username:        aws.String(userName),
      MessageAction:   types.MessageActionTypeSuppress,
    })
  if err != nil {
    log.Printf("Couldn't create user. Here's why: %v\n", err)
  }
  return err
}

```

```

    UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
aws.String(userEmail)}}},
})
if err != nil {
    var userExists *types.UsernameExistsException
    if errors.As(err, &userExists) {
        log.Printf("User %v already exists in the user pool.", userName)
        err = nil
    } else {
        log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
    }
}
return err
}

// AdminSetUserPassword uses administrator credentials to set a password for a user
// without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
string, userName string, password string) error {
    _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
    &cognitoidentityprovider.AdminSetUserPasswordInput{
        Password:    aws.String(password),
        UserPoolId:  aws.String(userPoolId),
        Username:    aws.String(userName),
        Permanent:   true,
    })
    if err != nil {
        var invalidPassword *types.InvalidPasswordException
        if errors.As(err, &invalidPassword) {
            log.Println(*invalidPassword.Message)
        } else {
            log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName, err)
        }
    }
    return err
}

```

Cree una estructura que ajuste las acciones de DynamoDB.

```
import (  
    "context"  
    "fmt"  
    "log"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"  
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"  
    "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"  
)  
  
// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)  
// actions  
// used in the examples.  
type DynamoActions struct {  
    DynamoClient *dynamodb.Client  
}  
  
// User defines structured user data.  
type User struct {  
    Username  string  
    UserEmail string  
    LastLogin *LoginInfo `dynamodbav:",omitempty"`  
}  
  
// LoginInfo defines structured custom login data.  
type LoginInfo struct {  
    UserPoolId string  
    ClientId   string  
    Time       string  
}  
  
// UserList defines a list of users.  
type UserList struct {  
    Users []User  
}  
  
// UserNameList returns the usernames contained in a UserList as a list of strings.  
func (users *UserList) UserNameList() []string {  
    names := make([]string, len(users.Users))  
    for i := 0; i < len(users.Users); i++ {  
        names[i] = users.Users[i].Username  
    }  
}
```

```

}
return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
error {
    var err error
    var item map[string]types.AttributeValue
    var writeReqs []types.WriteRequest
    for i := 1; i < 4; i++ {
        item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_%v",
i), userEmail: fmt.Sprintf("test_email_%v@example.com", i)})
        if err != nil {
            log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n", err)
            return err
        }
        writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
&types.PutRequest{Item: item}})
    }
    _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
})
    if err != nil {
        log.Printf("Couldn't populate table %v with users. Here's why: %v\n", tableName,
err)
    }
    return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
error) {
    var userList UserList
    output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
TableName: aws.String(tableName),
})
    if err != nil {
        log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName, err)
    } else {
        err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
        if err != nil {
            log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
        }
    }
}

```

```

}
return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user User)
error {
    userItem, err := attributevalue.MarshalMap(user)
    if err != nil {
        log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
    }
    _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
        Item:      userItem,
        TableName: aws.String(tableName),
    })
    if err != nil {
        log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
    }
    return err
}

```

Crea una estructura que agrupe las acciones de CloudWatch Logs.

```

import (
    "context"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
    "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

type CloudWatchLogsActions struct {
    CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
    functionName string) (types.LogStream, error) {

```

```

var logStream types.LogStream
logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
output, err := actor.CwlClient.DescribeLogStreams(ctx,
&cloudwatchlogs.DescribeLogStreamsInput{
    Descending:    aws.Bool(true),
    Limit:         aws.Int32(1),
    LogGroupName:  aws.String(logGroupName),
    OrderBy:       types.OrderByLastEventTime,
})
if err != nil {
    log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
logGroupName, err)
} else {
    logStream = output.LogStreams[0]
}
return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
string, logStreamName string, eventCount int32) (
[]types.OutputLogEvent, error) {
var events []types.OutputLogEvent
logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
output, err := actor.CwlClient.GetLogEvents(ctx, &cloudwatchlogs.GetLogEventsInput{
    LogStreamName: aws.String(logStreamName),
    Limit:         aws.Int32(eventCount),
    LogGroupName:  aws.String(logGroupName),
})
if err != nil {
    log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
logStreamName, err)
} else {
    events = output.Events
}
return events, err
}

```

Crea una estructura que agrupe las acciones. CloudFormation

```

import (
    "context"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
    CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
// structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName string)
    StackOutputs {
    output, err := actor.CfnClient.DescribeStacks(ctx,
        &cloudformation.DescribeStacksInput{
            StackName: aws.String(stackName),
        })
    if err != nil || len(output.Stacks) == 0 {
        log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
            stackName, err)
    }
    stackOutputs := StackOutputs{}
    for _, out := range output.Stacks[0].Outputs {
        stackOutputs[*out.OutputKey] = *out.OutputValue
    }
    return stackOutputs
}

```

Eliminación de recursos.

```

import (
    "context"
    "log"

```



```

"user_pools_and_lambda_triggers/actions"

"github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
    userPoolId      string
    userAccessTokens []string
    triggers        []actions.Trigger

    cognitoActor *actions.CognitoActions
    questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
demotools.IQuestioner) {
    resources.userAccessTokens = []string{}
    resources.triggers = []actions.Trigger{}
    resources.cognitoActor = cognitoActor
    resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
    defer func() {
        if r := recover(); r != nil {
            log.Printf("Something went wrong during cleanup.\n%v\n", r)
            log.Println("Use the AWS Management Console to remove any remaining resources \n"
+
            "that were created for this scenario.")
        }
    }()

    wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
resources that were created "+
    "during this demo (y/n)?", "y")
    if wantDelete {
        for _, accessToken := range resources.userAccessTokens {
            err := resources.cognitoActor.DeleteUser(ctx, accessToken)
            if err != nil {
                log.Println("Couldn't delete user during cleanup.")
                panic(err)
            }
        }
    }
}

```

```

    }
    log.Println("Deleted user.")
}
triggerList := make([]actions.TriggerInfo, len(resources.triggers))
for i := 0; i < len(resources.triggers); i++ {
    triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i], HandlerArn:
nil}
}
err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
triggerList...)
if err != nil {
    log.Println("Couldn't update Cognito triggers during cleanup.")
    panic(err)
}
log.Println("Removed Cognito triggers from user pool.")
} else {
    log.Println("Be sure to remove resources when you're done with them to avoid
unexpected charges!")
}
}
}

```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Go .
 - [ConfirmForgotPassword](#)
 - [DeleteUser](#)
 - [ForgotPassword](#)
 - [InitiateAuth](#)
 - [SignUp](#)
 - [UpdateUserPool](#)

Escriba datos de actividad personalizados con una función de Lambda tras la autenticación de usuario de Amazon Cognito

En el siguiente ejemplo de código, se muestra cómo escribir datos de actividad personalizados con una función de Lambda tras la autenticación de usuarios de Amazon Cognito.

- Utilice las funciones de administrador para añadir un usuario a un grupo de usuarios.

- Configure un grupo de usuarios para que llame a una función de Lambda para el desencadenador `PostAuthentication`.
- Inicie sesión con el nuevo usuario en Amazon Cognito.
- La función Lambda escribe información personalizada en los CloudWatch registros y en una tabla de DynamoDB.
- Obtenga y exhiba los datos personalizados de la tabla de DynamoDB y, a continuación, elimine los recursos.

SDK para Go V2

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](#).

Ejecutar un escenario interactivo en un símbolo del sistema.

```
import (  
    "context"  
    "errors"  
    "log"  
    "strings"  
    "user_pools_and_lambda_triggers/actions"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"  
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"  
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"  
)  
  
// ActivityLog separates the steps of this scenario into individual functions so  
// that  
// they are simpler to read and understand.  
type ActivityLog struct {  
    helper      IScenarioHelper  
    questioner  demotools.IQuestioner  
    resources   Resources  
    cognitoActor *actions.CognitoActions
```

```

}

// NewActivityLog constructs a new activity log runner.
func NewActivityLog(sdkConfig aws.Config, questioner demotools.IQuestioner, helper
    IScenarioHelper) ActivityLog {
    scenario := ActivityLog{
        helper:      helper,
        questioner:   questioner,
        resources:    Resources{},
        cognitoActor: &actions.CognitoActions{CognitoClient:
cognitoidentityprovider.NewFromConfig(sdkConfig)},
    }
    scenario.resources.init(scenario.cognitoActor, questioner)
    return scenario
}

// AddUserToPool selects a user from the known users table and uses administrator
    credentials to add the user to the user pool.
func (runner *ActivityLog) AddUserToPool(ctx context.Context, userPoolId string,
    tableName string) (string, string) {
    log.Println("To facilitate this example, let's add a user to the user pool using
    administrator privileges.")
    users, err := runner.helper.GetKnownUsers(ctx, tableName)
    if err != nil {
        panic(err)
    }
    user := users.Users[0]
    log.Printf("Adding known user %v to the user pool.\n", user.UserName)
    err = runner.cognitoActor.AdminCreateUser(ctx, userPoolId, user.UserName,
    user.UserEmail)
    if err != nil {
        panic(err)
    }
    pwSet := false
    password := runner.questioner.AskPassword("\nEnter a password that has at least
    eight characters, uppercase, lowercase, numbers and symbols.\n"+
        "(the password will not display as you type):", 8)
    for !pwSet {
        log.Printf("\nSetting password for user '%v'.\n", user.UserName)
        err = runner.cognitoActor.AdminSetUserPassword(ctx, userPoolId, user.UserName,
        password)
        if err != nil {
            var invalidPassword *types.InvalidPasswordException
            if errors.As(err, &invalidPassword) {

```

```

        password = runner.questioner.AskPassword("\nEnter another password:", 8)
    } else {
        panic(err)
    }
} else {
    pwSet = true
}
}

log.Println(strings.Repeat("-", 88))

return user.UserName, password
}

// AddActivityLogTrigger adds a Lambda handler as an invocation target for the
// PostAuthentication trigger.
func (runner *ActivityLog) AddActivityLogTrigger(ctx context.Context, userPoolId
string, activityLogArn string) {
    log.Println("Let's add a Lambda function to handle the PostAuthentication trigger
from Cognito.\n" +
        "This trigger happens after a user is authenticated, and lets your function take
action, such as logging\n" +
        "the outcome.")
    err := runner.cognitoActor.UpdateTriggers(
        ctx, userPoolId,
        actions.TriggerInfo{Trigger: actions.PostAuthentication, HandlerArn:
aws.String(activityLogArn)})
    if err != nil {
        panic(err)
    }
    runner.resources.triggers = append(runner.resources.triggers,
actions.PostAuthentication)
    log.Printf("Lambda function %v added to user pool %v to handle PostAuthentication
Cognito trigger.\n",
        activityLogArn, userPoolId)

    log.Println(strings.Repeat("-", 88))
}

// SignInUser signs in as the specified user.
func (runner *ActivityLog) SignInUser(ctx context.Context, clientId string, userName
string, password string) {
    log.Printf("Now we'll sign in user %v and check the results in the logs and the
DynamoDB table.", userName)

```

```

runner.questioner.Ask("Press Enter when you're ready.")
authResult, err := runner.cognitoActor.SignIn(ctx, clientId, userName, password)
if err != nil {
    panic(err)
}
log.Println("Sign in successful.",
    "The PostAuthentication Lambda handler writes custom information to CloudWatch Logs.")

runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
    *authResult.AccessToken)
}

// GetKnownUserLastLogin gets the login info for a user from the Amazon DynamoDB
// table and displays it.
func (runner *ActivityLog) GetKnownUserLastLogin(ctx context.Context, tableName
    string, userName string) {
    log.Println("The PostAuthentication handler also writes login data to the DynamoDB
        table.")
    runner.questioner.Ask("Press Enter when you're ready to continue.")
    users, err := runner.helper.GetKnownUsers(ctx, tableName)
    if err != nil {
        panic(err)
    }
    for _, user := range users.Users {
        if user.UserName == userName {
            log.Println("The last login info for the user in the known users table is:")
            log.Printf("\t%+v", *user.LastLogin)
        }
    }
    log.Println(strings.Repeat("-", 88))
}

// Run runs the scenario.
func (runner *ActivityLog) Run(ctx context.Context, stackName string) {
    defer func() {
        if r := recover(); r != nil {
            log.Println("Something went wrong with the demo.")
            runner.resources.Cleanup(ctx)
        }
    }()

    log.Println(strings.Repeat("-", 88))
    log.Printf("Welcome\n")

```

```

log.Println(strings.Repeat("-", 88))

stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
if err != nil {
    panic(err)
}
runner.resources.userPoolId = stackOutputs["UserPoolId"]
runner.helper.PopulateUserTable(ctx, stackOutputs["TableName"])
userName, password := runner.AddUserToPool(ctx, stackOutputs["UserPoolId"],
stackOutputs["TableName"])

runner.AddActivityLogTrigger(ctx, stackOutputs["UserPoolId"],
stackOutputs["ActivityLogFunctionArn"])
runner.SignInUser(ctx, stackOutputs["UserPoolClientId"], userName, password)
runner.helper.ListRecentLogEvents(ctx, stackOutputs["ActivityLogFunction"])
runner.GetKnownUserLastLogin(ctx, stackOutputs["TableName"], userName)

runner.resources.Cleanup(ctx)

log.Println(strings.Repeat("-", 88))
log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}

```

Controle el desencadenador PostAuthentication con una función de Lambda.

```

import (
    "context"
    "fmt"
    "log"
    "os"
    "time"

    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"

```

```

    dynamodbtypes "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

const TABLE_NAME = "TABLE_NAME"

// LoginInfo defines structured login data that can be marshalled to a DynamoDB
// format.
type LoginInfo struct {
    UserPoolId string `dynamodbav:"UserPoolId"`
    ClientId    string `dynamodbav:"ClientId"`
    Time       string `dynamodbav:"Time"`
}

// UserInfo defines structured user data that can be marshalled to a DynamoDB
// format.
type UserInfo struct {
    UserName    string    `dynamodbav:"UserName"`
    UserEmail   string    `dynamodbav:"UserEmail"`
    LastLogin   LoginInfo `dynamodbav:"LastLogin"`
}

// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
    userEmail, err := attributevalue.Marshal(user.UserEmail)
    if err != nil {
        panic(err)
    }
    return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
    dynamoClient *dynamodb.Client
}

// HandleRequest handles the PostAuthentication event by writing custom data to the
// logs and
// to an Amazon DynamoDB table.
func (h *handler) HandleRequest(ctx context.Context,
    event events.CognitoEventUserPoolsPostAuthentication)
    (events.CognitoEventUserPoolsPostAuthentication, error) {
    log.Printf("Received post authentication trigger from %v for user '%v'",
        event.TriggerSource, event.UserName)
    tableName := os.Getenv(TABLE_NAME)
    user := UserInfo{

```



```

    UserName: event.UserName,
    userEmail: event.Request.UserAttributes["email"],
    LastLogin: LoginInfo{
        UserPoolId: event.UserPoolID,
        ClientId:   event.CallerContext.ClientID,
        Time:       time.Now().Format(time.UnixDate),
    },
}
// Write to CloudWatch Logs.
fmt.Printf("%#v", user)

// Also write to an external system. This examples uses DynamoDB to demonstrate.
userMap, err := attributevalue.MarshalMap(user)
if err != nil {
    log.Printf("Couldn't marshal to DynamoDB map. Here's why: %v\n", err)
} else if len(userMap) == 0 {
    log.Printf("User info marshaled to an empty map.")
} else {
    _, err := h.dynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
        Item:      userMap,
        TableName: aws.String(tableName),
    })
    if err != nil {
        log.Printf("Couldn't write to DynamoDB. Here's why: %v\n", err)
    } else {
        log.Printf("Wrote user info to DynamoDB table %v.\n", tableName)
    }
}

return event, nil
}

func main() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        log.Panicln(err)
    }
    h := handler{
        dynamoClient: dynamodb.NewFromConfig(sdkConfig),
    }
    lambda.Start(h.HandleRequest)
}

```

Cree una estructura que lleve a cabo las tareas habituales.

```
import (
    "context"
    "log"
    "strings"
    "time"
    "user_pools_and_lambda_triggers/actions"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cloudformation"
    "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this example.
type IScenarioHelper interface {
    Pause(secs int)
    GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
        error)
    PopulateUserTable(ctx context.Context, tableName string)
    GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
    AddKnownUser(ctx context.Context, tableName string, user actions.User)
    ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
// example.
type ScenarioHelper struct {
    questioner demotools.IQuestioner
    dynamoActor *actions.DynamoActions
    cfnActor     *actions.CloudFormationActions
    cwlActor     *actions.CloudWatchLogsActions
    isTestRun    bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
    ScenarioHelper {
```

```

scenario := ScenarioHelper{
    questioner: questioner,
    dynamoActor: &actions.DynamoActions{DynamoClient:
dynamodb.NewFromConfig(sdkConfig)},
    cfnActor: &actions.CloudFormationActions{CfnClient:
cloudformation.NewFromConfig(sdkConfig)},
    cwlActor: &actions.CloudWatchLogsActions{CwlClient:
cloudwatchlogs.NewFromConfig(sdkConfig)},
}
return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
    if !helper.isTestRun {
        time.Sleep(time.Duration(secs) * time.Second)
    }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName string)
(actions.StackOutputs, error) {
    return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
string) {
    log.Printf("First, let's add some users to the DynamoDB %v table we'll use for this
example.\n", tableName)
    err := helper.dynamoActor.PopulateTable(ctx, tableName)
    if err != nil {
        panic(err)
    }
}

// GetKnownUsers gets the users from the known users table in a structured format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
(actions.UserList, error) {
    knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
    if err != nil {
        log.Printf("Couldn't get known users from table %v. Here's why: %v\n", tableName,
err)
    }
}

```

```

}
return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
    user actions.User) {
    log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users table...
\n",
        user.UserName, user.UserEmail)
    err := helper.dynamoActor.AddUser(ctx, tableName, user)
    if err != nil {
        panic(err)
    }
}

// ListRecentLogEvents gets the most recent log stream and events for the specified
    Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context, functionName
    string) {
    log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
    helper.Pause(10)
    log.Println("Okay, let's check the logs to find what's happened recently with your
    Lambda function.")
    logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
    if err != nil {
        panic(err)
    }
    log.Printf("Getting some recent events from log stream %v\n",
        *logStream.LogStreamName)
    events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
        *logStream.LogStreamName, 10)
    if err != nil {
        panic(err)
    }
    for _, event := range events {
        log.Printf("\t%v", *event.Message)
    }
    log.Println(strings.Repeat("-", 88))
}

```

Cree una estructura que ajuste las acciones de Amazon Cognito.

```
import (  
    "context"  
    "errors"  
    "log"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"  
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"  
)  
  
type CognitoActions struct {  
    CognitoClient *cognitoidentityprovider.Client  
}  
  
// Trigger and TriggerInfo define typed data for updating an Amazon Cognito trigger.  
type Trigger int  
  
const (  
    PreSignUp Trigger = iota  
    UserMigration  
    PostAuthentication  
)  
  
type TriggerInfo struct {  
    Trigger    Trigger  
    HandlerArn *string  
}  
  
// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger is  
// specified with a `nil` value,  
// it is removed from the user pool.  
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId string,  
    triggers ...TriggerInfo) error {  
    output, err := actor.CognitoClient.DescribeUserPool(ctx,  
        &cognitoidentityprovider.DescribeUserPoolInput{  
            UserPoolId: aws.String(userPoolId),  
        })  
    if err != nil {
```

```

    log.Printf("Couldn't get info about user pool %v. Here's why: %v\n", userPoolId,
err)
    return err
}
lambdaConfig := output.UserPool.LambdaConfig
for _, trigger := range triggers {
    switch trigger.Trigger {
    case PreSignUp:
        lambdaConfig.PreSignUp = trigger.HandlerArn
    case UserMigration:
        lambdaConfig.UserMigration = trigger.HandlerArn
    case PostAuthentication:
        lambdaConfig.PostAuthentication = trigger.HandlerArn
    }
}
_, err = actor.CognitoClient.UpdateUserPool(ctx,
&cognitoidentityprovider.UpdateUserPoolInput{
    UserPoolId:    aws.String(userPoolId),
    LambdaConfig: lambdaConfig,
})
if err != nil {
    log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
}
return err
}

// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
string, password string, userEmail string) (bool, error) {
    confirmed := false
    output, err := actor.CognitoClient.SignUp(ctx,
&cognitoidentityprovider.SignUpInput{
        ClientId: aws.String(clientId),
        Password: aws.String(password),
        Username: aws.String(userName),
        UserAttributes: []types.AttributeType{
            {Name: aws.String("email"), Value: aws.String(userEmail)},
        },
    })
    if err != nil {
        var invalidPassword *types.InvalidPasswordException
        if errors.As(err, &invalidPassword) {

```

```

    log.Println(*invalidPassword.Message)
} else {
    log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
}
} else {
    confirmed = output.UserConfirmed
}
return confirmed, err
}

// SignIn signs in a user to Amazon Cognito using a username and password
authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
string, password string) (*types.AuthenticationResultType, error) {
    var authResult *types.AuthenticationResultType
    output, err := actor.CognitoClient.InitiateAuth(ctx,
    &cognitoidentityprovider.InitiateAuthInput{
        AuthFlow:      "USER_PASSWORD_AUTH",
        ClientId:      aws.String(clientId),
        AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
    })
    if err != nil {
        var resetRequired *types.PasswordResetRequiredException
        if errors.As(err, &resetRequired) {
            log.Println(*resetRequired.Message)
        } else {
            log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
        }
    } else {
        authResult = output.AuthenticationResult
    }
    return authResult, err
}

// ForgotPassword starts a password recovery flow for a user. This flow typically
sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
userName string) (*types.CodeDeliveryDetailsType, error) {

```

```

output, err := actor.CognitoClient.ForgotPassword(ctx,
&cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
})
if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here's why: %v\n",
userName, err)
}
return output.CodeDeliveryDetails, err
}

// ConfirmForgotPassword confirms a user with a confirmation code and a new
password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
string, code string, userName string, password string) error {
_, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
&cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:      aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:      aws.String(password),
  Username:      aws.String(userName),
})
if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
    log.Println(*invalidPassword.Message)
  } else {
    log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
}
return err
}

// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken string)
error {
_, err := actor.CognitoClient.DeleteUser(ctx,
&cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),

```



```

    })
    if err != nil {
        log.Printf("Couldn't delete user. Here's why: %v\n", err)
    }
    return err
}

// AdminCreateUser uses administrator credentials to add a user to a user pool. This
// method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId string,
    userName string, userEmail string) error {
    _, err := actor.CognitoClient.AdminCreateUser(ctx,
        &cognitoidentityprovider.AdminCreateUserInput{
            UserPoolId:      aws.String(userPoolId),
            Username:        aws.String(userName),
            MessageAction:   types.MessageActionTypeSuppress,
            UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
aws.String(userEmail)}}},
    })
    if err != nil {
        var userExists *types.UsernameExistsException
        if errors.As(err, &userExists) {
            log.Printf("User %v already exists in the user pool.", userName)
            err = nil
        } else {
            log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
        }
    }
    return err
}

// AdminSetUserPassword uses administrator credentials to set a password for a user
// without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
    string, userName string, password string) error {
    _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
        &cognitoidentityprovider.AdminSetUserPasswordInput{
            Password:      aws.String(password),

```

```

    UserPoolId: aws.String(userPoolId),
    Username:   aws.String(userName),
    Permanent:  true,
  })
  if err != nil {
    var invalidPassword *types.InvalidPasswordException
    if errors.As(err, &invalidPassword) {
      log.Println(*invalidPassword.Message)
    } else {
      log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName, err)
    }
  }
  return err
}

```

Cree una estructura que ajuste las acciones de DynamoDB.

```

import (
    "context"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
// actions
// used in the examples.
type DynamoActions struct {
    DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
    UserName  string
    UserEmail string
    LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

```

```

}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
    UserPoolId string
    ClientId    string
    Time        string
}

// UserList defines a list of users.
type UserList struct {
    Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of strings.
func (users *UserList) UserNameList() []string {
    names := make([]string, len(users.Users))
    for i := 0; i < len(users.Users); i++ {
        names[i] = users.Users[i].UserName
    }
    return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
error {
    var err error
    var item map[string]types.AttributeValue
    var writeReqs []types.WriteRequest
    for i := 1; i < 4; i++ {
        item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_%v",
i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
        if err != nil {
            log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n", err)
            return err
        }
        writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
&types.PutRequest{Item: item}})
    }
    _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
    RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
})
    if err != nil {

```

```

    log.Printf("Couldn't populate table %v with users. Here's why: %v\n", tableName,
err)
}
return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
error) {
    var userList UserList
    output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
        TableName: aws.String(tableName),
    })
    if err != nil {
        log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName, err)
    } else {
        err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
        if err != nil {
            log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
        }
    }
    return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user User)
error {
    userItem, err := attributevalue.MarshalMap(user)
    if err != nil {
        log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
    }
    _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
        Item:      userItem,
        TableName: aws.String(tableName),
    })
    if err != nil {
        log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
    }
    return err
}

```

Crea una estructura que agrupe las acciones de CloudWatch Logs.

```
import (
    "context"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
    "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

type CloudWatchLogsActions struct {
    CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
    functionName string) (types.LogStream, error) {
    var logStream types.LogStream
    logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
    output, err := actor.CwlClient.DescribeLogStreams(ctx,
        &cloudwatchlogs.DescribeLogStreamsInput{
            Descending:    aws.Bool(true),
            Limit:         aws.Int32(1),
            LogGroupName: aws.String(logGroupName),
            OrderBy:      types.OrderByLastEventTime,
        })
    if err != nil {
        log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
            logGroupName, err)
    } else {
        logStream = output.LogStreams[0]
    }
    return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
// stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
    string, logStreamName string, eventCount int32) (
    []types.OutputLogEvent, error) {
    var events []types.OutputLogEvent
```

```

logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
output, err := actor.CwlClient.GetLogEvents(ctx, &cloudwatchlogs.GetLogEventsInput{
    LogStreamName: aws.String(logStreamName),
    Limit:         aws.Int32(eventCount),
    LogGroupName:  aws.String(logGroupName),
})
if err != nil {
    log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
logStreamName, err)
} else {
    events = output.Events
}
return events, err
}

```

Crea una estructura que agrupe las acciones. CloudFormation

```

import (
    "context"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
    CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
// structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName string)
StackOutputs {
    output, err := actor.CfnClient.DescribeStacks(ctx,
&cloudformation.DescribeStacksInput{
    StackName: aws.String(stackName),
})
}

```

```

if err != nil || len(output.Stacks) == 0 {
    log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
        stackName, err)
}
stackOutputs := StackOutputs{}
for _, out := range output.Stacks[0].Outputs {
    stackOutputs[*out.OutputKey] = *out.OutputValue
}
return stackOutputs
}

```

Eliminación de recursos.

```

import (
    "context"
    "log"
    "user_pools_and_lambda_triggers/actions"

    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
    userPoolId      string
    userAccessTokens []string
    triggers        []actions.Trigger

    cognitoActor *actions.CognitoActions
    questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
    demotools.IQuestioner) {
    resources.userAccessTokens = []string{}
    resources.triggers = []actions.Trigger{}
    resources.cognitoActor = cognitoActor
    resources.questioner = questioner
}

```

```
// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
    defer func() {
        if r := recover(); r != nil {
            log.Printf("Something went wrong during cleanup.\n%v\n", r)
            log.Println("Use the AWS Management Console to remove any remaining resources \n"
+
            "that were created for this scenario.")
        }
    }()

    wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
resources that were created "+
    "during this demo (y/n)?", "y")
    if wantDelete {
        for _, accessToken := range resources.userAccessTokens {
            err := resources.cognitoActor.DeleteUser(ctx, accessToken)
            if err != nil {
                log.Println("Couldn't delete user during cleanup.")
                panic(err)
            }
            log.Println("Deleted user.")
        }
        triggerList := make([]actions.TriggerInfo, len(resources.triggers))
        for i := 0; i < len(resources.triggers); i++ {
            triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i], HandlerArn:
nil}
        }
        err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
triggerList...)
        if err != nil {
            log.Println("Couldn't update Cognito triggers during cleanup.")
            panic(err)
        }
        log.Println("Removed Cognito triggers from user pool.")
    } else {
        log.Println("Be sure to remove resources when you're done with them to avoid
unexpected charges!")
    }
}
```


- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Go .
 - [AdminCreateUser](#)
 - [AdminSetUserPassword](#)
 - [DeleteUser](#)
 - [InitiateAuth](#)
 - [UpdateUserPool](#)

Ejemplos de Amazon DocumentDB con SDK para Go V2

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso de la AWS SDK para Go V2 con Amazon DocumentDB.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Ejemplos de tecnología sin servidor](#)

Ejemplos de tecnología sin servidor

Invocación de una función de Lambda desde un desencadenador de Amazon DocumentDB

El siguiente ejemplo de código muestra cómo implementar una función de Lambda que recibe un evento desencadenado al recibir registros de una secuencia de cambios de DocumentDB. La función recupera la carga útil de DocumentDB y registra el contenido del registro.

SDK para Go V2

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Consumir un evento de Amazon DocumentDB con Lambda utilizando Go.

```
package main

import (
    "context"
    "encoding/json"
    "fmt"

    "github.com/aws/aws-lambda-go/lambda"
)

type Event struct {
    Events []Record `json:"events"`
}

type Record struct {
    Event struct {
        OperationType string `json:"operationType"`
        NS            struct {
            DB    string `json:"db"`
            Coll string `json:"coll"`
        } `json:"ns"`
        FullDocument interface{} `json:"fullDocument"`
    } `json:"event"`
}

func main() {
    lambda.Start(handler)
}

func handler(ctx context.Context, event Event) (string, error) {
    fmt.Println("Loading function")
    for _, record := range event.Events {
        logDocumentDBEvent(record)
    }

    return "OK", nil
}

func logDocumentDBEvent(record Record) {
    fmt.Printf("Operation type: %s\n", record.Event.OperationType)
    fmt.Printf("db: %s\n", record.Event.NS.DB)
    fmt.Printf("collection: %s\n", record.Event.NS.Coll)
```

```
docBytes, _ := json.MarshalIndent(record.Event.FullDocument, "", " ")
fmt.Printf("Full document: %s\n", string(docBytes))
}
```

Ejemplos de DynamoDB usando SDK para Go V2

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante la AWS SDK para Go V2 con DynamoDB.

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.

AWS Las contribuciones de la comunidad son ejemplos que fueron creados y mantenidos por varios equipos de distintos equipos. AWS Para enviar comentarios, utilice el mecanismo previsto en los repositorios vinculados.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Conceptos básicos](#)
- [Acciones](#)
- [Escenarios](#)
- [Ejemplos de tecnología sin servidor](#)
- [AWS contribuciones de la comunidad](#)

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 Go V2

Note

Hay más información sobre GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Ejecutar un escenario interactivo para crear la tabla y realizar acciones en ella.

```
import (  
    "context"  
    "fmt"  
    "log"  
    "strings"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"  
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"  
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/dynamodb/actions"  
)  
  
// RunMovieScenario is an interactive example that shows you how to use the AWS SDK  
// for Go  
// to create and use an Amazon DynamoDB table that stores data about movies.  
//
```

```
// 1. Create a table that can hold movie data.
// 2. Put, get, and update a single movie in the table.
// 3. Write movie data to the table from a sample JSON file.
// 4. Query for movies that were released in a given year.
// 5. Scan for movies that were released in a range of years.
// 6. Delete a movie from the table.
// 7. Delete the table.
//
// This example creates a DynamoDB service client from the specified sdkConfig so
// that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
// example.
// This package can be found in the ../..demotools folder of this repo.
//
// The specified movie sampler is used to get sample data from a URL that is loaded
// into the named table.
func RunMovieScenario(
    ctx context.Context, sdkConfig aws.Config, questioner demotools.IQuestioner,
    tableName string,
    movieSampler actions.IMovieSampler) {
    defer func() {
        if r := recover(); r != nil {
            fmt.Printf("Something went wrong with the demo.")
        }
    }()

    log.Println(strings.Repeat("-", 88))
    log.Println("Welcome to the Amazon DynamoDB getting started demo.")
    log.Println(strings.Repeat("-", 88))

    tableBasics := actions.TableBasics{TableName: tableName,
        DynamoDbClient: dynamodb.NewFromConfig(sdkConfig)}

    exists, err := tableBasics.TableExists(ctx)
    if err != nil {
        panic(err)
    }
    if !exists {
        log.Printf("Creating table %v...\n", tableName)
        _, err = tableBasics.CreateMovieTable(ctx)
        if err != nil {
            panic(err)
        }
    }
}
```

```

    } else {
        log.Printf("Created table %v.\n", tableName)
    }
} else {
    log.Printf("Table %v already exists.\n", tableName)
}

var customMovie actions.Movie
customMovie.Title = questioner.Ask("Enter a movie title to add to the table:",
    demotools.NotEmpty{})
customMovie.Year = questioner.AskInt("What year was it released?",
    demotools.NotEmpty{}, demotools.InIntRange{Lower: 1900, Upper: 2030})
customMovie.Info = map[string]interface{}{}
customMovie.Info["rating"] = questioner.AskFloat64(
    "Enter a rating between 1 and 10:",
    demotools.NotEmpty{}, demotools.InFloatRange{Lower: 1, Upper: 10})
customMovie.Info["plot"] = questioner.Ask("What's the plot? ",
    demotools.NotEmpty{})
err = tableBasics.AddMovie(ctx, customMovie)
if err == nil {
    log.Printf("Added %v to the movie table.\n", customMovie.Title)
}
log.Println(strings.Repeat("-", 88))

log.Printf("Let's update your movie. You previously rated it %v.\n",
    customMovie.Info["rating"])
customMovie.Info["rating"] = questioner.AskFloat64(
    "What new rating would you give it?",
    demotools.NotEmpty{}, demotools.InFloatRange{Lower: 1, Upper: 10})
log.Printf("You summarized the plot as '%v'.\n", customMovie.Info["plot"])
customMovie.Info["plot"] = questioner.Ask("What would you say now?",
    demotools.NotEmpty{})
attributes, err := tableBasics.UpdateMovie(ctx, customMovie)
if err == nil {
    log.Printf("Updated %v with new values.\n", customMovie.Title)
    for _, attVal := range attributes {
        for valKey, val := range attVal {
            log.Printf("\t%v: %v\n", valKey, val)
        }
    }
}
log.Println(strings.Repeat("-", 88))

log.Printf("Getting movie data from %v and adding 250 movies to the table...\n",

```

```

    movieSampler.GetURL())
movies := movieSampler.GetSampleMovies()
written, err := tableBasics.AddMovieBatch(ctx, movies, 250)
if err != nil {
    panic(err)
} else {
    log.Printf("Added %v movies to the table.\n", written)
}

show := 10
if show > written {
    show = written
}
log.Printf("The first %v movies in the table are:", show)
for index, movie := range movies[:show] {
    log.Printf("\t%v. %v\n", index+1, movie.Title)
}
movieIndex := questioner.AskInt(
    "Enter the number of a movie to get info about it: ",
    demotools.InIntRange{Lower: 1, Upper: show},
)
movie, err := tableBasics.GetMovie(ctx, movies[movieIndex-1].Title,
movies[movieIndex-1].Year)
if err == nil {
    log.Println(movie)
}
log.Println(strings.Repeat("-", 88))

log.Println("Let's get a list of movies released in a given year.")
releaseYear := questioner.AskInt("Enter a year between 1972 and 2018: ",
    demotools.InIntRange{Lower: 1972, Upper: 2018},
)
releases, err := tableBasics.Query(ctx, releaseYear)
if err == nil {
    if len(releases) == 0 {
        log.Printf("I couldn't find any movies released in %v!\n", releaseYear)
    } else {
        for _, movie = range releases {
            log.Println(movie)
        }
    }
}
log.Println(strings.Repeat("-", 88))

```

```

log.Println("Now let's scan for movies released in a range of years.")
startYear := questioner.AskInt("Enter a year: ",
    demotools.InIntRange{Lower: 1972, Upper: 2018})
endYear := questioner.AskInt("Enter another year: ",
    demotools.InIntRange{Lower: 1972, Upper: 2018})
releases, err = tableBasics.Scan(ctx, startYear, endYear)
if err == nil {
    if len(releases) == 0 {
        log.Printf("I couldn't find any movies released between %v and %v!\n", startYear,
endYear)
    } else {
        log.Printf("Found %v movies. In this list, the plot is <nil> because "+
            "we used a projection expression when scanning for items to return only "+
            "the title, year, and rating.\n", len(releases))
        for _, movie = range releases {
            log.Println(movie)
        }
    }
}
log.Println(strings.Repeat("-", 88))

var tables []string
if questioner.AskBool("Do you want to list all of your tables? (y/n) ", "y") {
    tables, err = tableBasics.ListTables(ctx)
    if err == nil {
        log.Printf("Found %v tables:", len(tables))
        for _, table := range tables {
            log.Printf("\t%v", table)
        }
    }
}
log.Println(strings.Repeat("-", 88))

log.Printf("Let's remove your movie '%v'.\n", customMovie.Title)
if questioner.AskBool("Do you want to delete it from the table? (y/n) ", "y") {
    err = tableBasics.DeleteMovie(ctx, customMovie)
}
if err == nil {
    log.Printf("Deleted %v.\n", customMovie.Title)
}

if questioner.AskBool("Delete the table, too? (y/n)", "y") {
    err = tableBasics.DeleteTable(ctx)
} else {

```



```

    log.Println("Don't forget to delete the table when you're done or you might " +
        "incur charges on your account.")
}
if err == nil {
    log.Printf("Deleted table %v.\n", tableBasics.TableName)
}

log.Println(strings.Repeat("-", 88))
log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}

```

Defina una estructura Película para utilizar en este ejemplo.

```

import (
    "archive/zip"
    "bytes"
    "encoding/json"
    "fmt"
    "io"
    "log"
    "net/http"

    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
// key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
// key,
// and Info is additional data.
type Movie struct {
    Title string          `dynamodbav:"title"`
    Year  int                `dynamodbav:"year"`
    Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.

```

```

func (movie Movie) GetKey() map[string]types.AttributeValue {
    title, err := attributevalue.Marshal(movie.Title)
    if err != nil {
        panic(err)
    }
    year, err := attributevalue.Marshal(movie.Year)
    if err != nil {
        panic(err)
    }
    return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
// example.
func (movie Movie) String() string {
    return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
        movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}

```

Crear una estructura y los métodos que llaman a las acciones de DynamoDB.

```

import (
    "context"
    "errors"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
// examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
    TableName      string
}

```

```

}

// TableExists determines whether a DynamoDB table exists.
func (basics TableBasics) TableExists(ctx context.Context) (bool, error) {
    exists := true
    _, err := basics.DynamoDbClient.DescribeTable(
        ctx, &dynamodb.DescribeTableInput{TableName: aws.String(basics.TableName)},
    )
    if err != nil {
        var notFoundEx *types.ResourceNotFoundException
        if errors.As(err, &notFoundEx) {
            log.Printf("Table %v does not exist.\n", basics.TableName)
            err = nil
        } else {
            log.Printf("Couldn't determine existence of table %v. Here's why: %v\n",
basics.TableName, err)
        }
        exists = false
    }
    return exists, err
}

// CreateMovieTable creates a DynamoDB table with a composite primary key defined as
// a string sort key named `title`, and a numeric partition key named `year`.
// This function uses NewTableExistsWaiter to wait for the table to be created by
// DynamoDB before it returns.
func (basics TableBasics) CreateMovieTable(ctx context.Context)
(*types.TableDescription, error) {
    var tableDesc *types.TableDescription
    table, err := basics.DynamoDbClient.CreateTable(ctx, &dynamodb.CreateTableInput{
        AttributeDefinitions: []types.AttributeDefinition{{
            AttributeName: aws.String("year"),
            AttributeType: types.ScalarAttributeTypeN,
        }}, {
            AttributeName: aws.String("title"),
            AttributeType: types.ScalarAttributeTypeS,
        }},
        KeySchema: []types.KeySchemaElement{{
            AttributeName: aws.String("year"),
            KeyType:      types.KeyTypeHash,

```

```

    }, {
        AttributeName: aws.String("title"),
        KeyType:      types.KeyTypeRange,
    }},
    TableName:      aws.String(basics.TableName),
    BillingMode:     types.BillingModePayPerRequest,
})
if err != nil {
    log.Printf("Couldn't create table %v. Here's why: %v\n", basics.TableName, err)
} else {
    waiter := dynamodb.NewTableExistsWaiter(basics.DynamoDbClient)
    err = waiter.Wait(ctx, &dynamodb.DescribeTableInput{
        TableName: aws.String(basics.TableName)}, 5*time.Minute)
    if err != nil {
        log.Printf("Wait for table exists failed. Here's why: %v\n", err)
    }
    tableDesc = table.TableDescription
    log.Printf("Ccreating table test")
}
return tableDesc, err
}

// ListTables lists the DynamoDB table names for the current account.
func (basics TableBasics) ListTables(ctx context.Context) ([]string, error) {
    var tableNames []string
    var output *dynamodb.ListTablesOutput
    var err error
    tablePaginator := dynamodb.NewListTablesPaginator(basics.DynamoDbClient,
        &dynamodb.ListTablesInput{})
    for tablePaginator.HasMorePages() {
        output, err = tablePaginator.NextPage(ctx)
        if err != nil {
            log.Printf("Couldn't list tables. Here's why: %v\n", err)
            break
        } else {
            tableNames = append(tableNames, output.TableNames...)
        }
    }
    return tableNames, err
}

```

```
// AddMovie adds a movie the DynamoDB table.
func (basics TableBasics) AddMovie(ctx context.Context, movie Movie) error {
    item, err := attributevalue.MarshalMap(movie)
    if err != nil {
        panic(err)
    }
    _, err = basics.DynamoDbClient.PutItem(ctx, &dynamodb.PutItemInput{
        TableName: aws.String(basics.TableName), Item: item,
    })
    if err != nil {
        log.Printf("Couldn't add item to table. Here's why: %v\n", err)
    }
    return err
}

// UpdateMovie updates the rating and plot of a movie that already exists in the
// DynamoDB table. This function uses the `expression` package to build the update
// expression.
func (basics TableBasics) UpdateMovie(ctx context.Context, movie Movie)
    (map[string]map[string]interface{}, error) {
    var err error
    var response *dynamodb.UpdateItemOutput
    var attributeMap map[string]map[string]interface{}
    update := expression.Set(expression.Name("info.rating"),
        expression.Value(movie.Info["rating"]))
    update.Set(expression.Name("info.plot"), expression.Value(movie.Info["plot"]))
    expr, err := expression.NewBuilder().WithUpdate(update).Build()
    if err != nil {
        log.Printf("Couldn't build expression for update. Here's why: %v\n", err)
    } else {
        response, err = basics.DynamoDbClient.UpdateItem(ctx, &dynamodb.UpdateItemInput{
            TableName:      aws.String(basics.TableName),
            Key:             movie.GetKey(),
            ExpressionAttributeNames: expr.Names(),
            ExpressionAttributeValues: expr.Values(),
            UpdateExpression: expr.Update(),
            ReturnValues:    types.ReturnValueUpdatedNew,
        })
        if err != nil {
            log.Printf("Couldn't update movie %v. Here's why: %v\n", movie.Title, err)
        } else {

```

```

    err = attributevalue.UnmarshalMap(response.Attributes, &attributeMap)
    if err != nil {
        log.Printf("Couldn't unmarshall update response. Here's why: %v\n", err)
    }
}
return attributeMap, err
}

// AddMovieBatch adds a slice of movies to the DynamoDB table. The function sends
// batches of 25 movies to DynamoDB until all movies are added or it reaches the
// specified maximum.
func (basics TableBasics) AddMovieBatch(ctx context.Context, movies []Movie,
    maxMovies int) (int, error) {
    var err error
    var item map[string]types.AttributeValue
    written := 0
    batchSize := 25 // DynamoDB allows a maximum batch size of 25 items.
    start := 0
    end := start + batchSize
    for start < maxMovies && start < len(movies) {
        var writeReqs []types.WriteRequest
        if end > len(movies) {
            end = len(movies)
        }
        for _, movie := range movies[start:end] {
            item, err = attributevalue.MarshalMap(movie)
            if err != nil {
                log.Printf("Couldn't marshal movie %v for batch writing. Here's why: %v\n",
movie.Title, err)
            } else {
                writeReqs = append(
                    writeReqs,
                    types.WriteRequest{PutRequest: &types.PutRequest{Item: item}},
                )
            }
        }
        _, err = basics.DynamoDbClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
            RequestItems: map[string][]types.WriteRequest{basics.TableName: writeReqs}})
        if err != nil {
            log.Printf("Couldn't add a batch of movies to %v. Here's why: %v\n",
basics.TableName, err)

```

```

    } else {
        written += len(writeReqs)
    }
    start = end
    end += batchSize
}

return written, err
}

// GetMovie gets movie data from the DynamoDB table by using the primary composite
// key
// made of title and year.
func (basics TableBasics) GetMovie(ctx context.Context, title string, year int)
(Movie, error) {
    movie := Movie{Title: title, Year: year}
    response, err := basics.DynamoDbClient.GetItem(ctx, &dynamodb.GetItemInput{
        Key: movie.GetKey(), TableName: aws.String(basics.TableName),
    })
    if err != nil {
        log.Printf("Couldn't get info about %v. Here's why: %v\n", title, err)
    } else {
        err = attributevalue.UnmarshalMap(response.Item, &movie)
        if err != nil {
            log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
        }
    }
    return movie, err
}

// Query gets all movies in the DynamoDB table that were released in the specified
// year.
// The function uses the `expression` package to build the key condition expression
// that is used in the query.
func (basics TableBasics) Query(ctx context.Context, releaseYear int) ([]Movie,
error) {
    var err error
    var response *dynamodb.QueryOutput
    var movies []Movie
    keyEx := expression.Key("year").Equal(expression.Value(releaseYear))

```

```

expr, err := expression.NewBuilder().WithKeyCondition(keyEx).Build()
if err != nil {
    log.Printf("Couldn't build expression for query. Here's why: %v\n", err)
} else {
    queryPaginator := dynamodb.NewQueryPaginator(basics.DynamoDbClient,
&dynamodb.QueryInput{
        TableName:           aws.String(basics.TableName),
        ExpressionAttributeNames:  expr.Names(),
        ExpressionAttributeValues: expr.Values(),
        KeyConditionExpression:   expr.KeyCondition(),
    })
    for queryPaginator.HasMorePages() {
        response, err = queryPaginator.NextPage(ctx)
        if err != nil {
            log.Printf("Couldn't query for movies released in %v. Here's why: %v\n",
releaseYear, err)
            break
        } else {
            var moviePage []Movie
            err = attributevalue.UnmarshalListOfMaps(response.Items, &moviePage)
            if err != nil {
                log.Printf("Couldn't unmarshal query response. Here's why: %v\n", err)
                break
            } else {
                movies = append(movies, moviePage...)
            }
        }
    }
}
return movies, err
}

```

```

// Scan gets all movies in the DynamoDB table that were released in a range of years
// and projects them to return a reduced set of fields.
// The function uses the `expression` package to build the filter and projection
// expressions.

```

```

func (basics TableBasics) Scan(ctx context.Context, startYear int, endYear int)
([]Movie, error) {
    var movies []Movie
    var err error
    var response *dynamodb.ScanOutput

```



```

filtEx := expression.Name("year").Between(expression.Value(startYear),
expression.Value(endYear))
projEx := expression.NamesList(
    expression.Name("year"), expression.Name("title"), expression.Name("info.rating"))
expr, err :=
expression.NewBuilder().WithFilter(filtEx).WithProjection(projEx).Build()
if err != nil {
    log.Printf("Couldn't build expressions for scan. Here's why: %v\n", err)
} else {
    scanPaginator := dynamodb.NewScanPaginator(basics.DynamoDbClient,
&dynamodb.ScanInput{
        TableName:            aws.String(basics.TableName),
        ExpressionAttributeNames:  expr.Names(),
        ExpressionAttributeValues: expr.Values(),
        FilterExpression:        expr.Filter(),
        ProjectionExpression:    expr.Projection(),
    })
    for scanPaginator.HasMorePages() {
        response, err = scanPaginator.NextPage(ctx)
        if err != nil {
            log.Printf("Couldn't scan for movies released between %v and %v. Here's why: %v\n",
                startYear, endYear, err)
            break
        } else {
            var moviePage []Movie
            err = attributevalue.UnmarshalListOfMaps(response.Items, &moviePage)
            if err != nil {
                log.Printf("Couldn't unmarshal query response. Here's why: %v\n", err)
                break
            } else {
                movies = append(movies, moviePage...)
            }
        }
    }
}
return movies, err
}

// DeleteMovie removes a movie from the DynamoDB table.
func (basics TableBasics) DeleteMovie(ctx context.Context, movie Movie) error {
    _, err := basics.DynamoDbClient.DeleteItem(ctx, &dynamodb.DeleteItemInput{

```

```
    TableName: aws.String(basics.TableName), Key: movie.GetKey(),
  })
  if err != nil {
    log.Printf("Couldn't delete %v from the table. Here's why: %v\n", movie.Title,
err)
  }
  return err
}

// DeleteTable deletes the DynamoDB table and all of its data.
func (basics TableBasics) DeleteTable(ctx context.Context) error {
  _, err := basics.DynamoDbClient.DeleteTable(ctx, &dynamodb.DeleteTableInput{
    TableName: aws.String(basics.TableName)})
  if err != nil {
    log.Printf("Couldn't delete table %v. Here's why: %v\n", basics.TableName, err)
  }
  return err
}
```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Go .
 - [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 Go V2

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Defina una estructura de receptor de funciones para el ejemplo.

```
import (
    "context"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// PartiQLRunner encapsulates the Amazon DynamoDB service actions used in the
// PartiQL examples. It contains a DynamoDB service client that is used to act on
// the
// specified table.
type PartiQLRunner struct {
    DynamoDbClient *dynamodb.Client
    TableName      string
}
```

Utilizar lotes de instrucciones INSERT para agregar elementos.

```
// AddMovieBatch runs a batch of PartiQL INSERT statements to add multiple movies to
// the
// DynamoDB table.
func (runner PartiQLRunner) AddMovieBatch(ctx context.Context, movies []Movie) error
{
    statementRequests := make([]types.BatchStatementRequest, len(movies))
    for index, movie := range movies {
        params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year,
movie.Info})
        if err != nil {
            panic(err)
        }
        statementRequests[index] = types.BatchStatementRequest{
            Statement: aws.String(fmt.Sprintf(
                "INSERT INTO \"%v\" VALUE {'title': ?, 'year': ?, 'info': ?}",
runner.TableName)),
            Parameters: params,
        }
    }

    _, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
&dynamodb.BatchExecuteStatementInput{
    Statements: statementRequests,
})
    if err != nil {
        log.Printf("Couldn't insert a batch of items with PartiQL. Here's why: %v\n", err)
    }
    return err
}
```

Utilizar lotes de instrucciones SELECT para obtener elementos.

```
// GetMovieBatch runs a batch of PartiQL SELECT statements to get multiple movies
// from
// the DynamoDB table by title and year.
func (runner PartiQLRunner) GetMovieBatch(ctx context.Context, movies []Movie)
([]Movie, error) {
    statementRequests := make([]types.BatchStatementRequest, len(movies))
    for index, movie := range movies {
        params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year})
```

```

    if err != nil {
        panic(err)
    }
    statementRequests[index] = types.BatchStatementRequest{
        Statement: aws.String(
            fmt.Sprintf("SELECT * FROM \"%v\" WHERE title=? AND year=?", runner.TableName)),
        Parameters: params,
    }
}

output, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
    &dynamodb.BatchExecuteStatementInput{
        Statements: statementRequests,
    })
var outMovies []Movie
if err != nil {
    log.Printf("Couldn't get a batch of items with PartiQL. Here's why: %v\n", err)
} else {
    for _, response := range output.Responses {
        var movie Movie
        err = attributevalue.UnmarshalMap(response.Item, &movie)
        if err != nil {
            log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
        } else {
            outMovies = append(outMovies, movie)
        }
    }
}
return outMovies, err
}

```

Utilizar lotes de instrucciones UPDATE para actualizar elementos.

```

// UpdateMovieBatch runs a batch of PartiQL UPDATE statements to update the rating
// of
// multiple movies that already exist in the DynamoDB table.
func (runner PartiQLRunner) UpdateMovieBatch(ctx context.Context, movies []Movie,
    ratings []float64) error {
    statementRequests := make([]types.BatchStatementRequest, len(movies))
    for index, movie := range movies {

```

```

    params, err := attributevalue.MarshalList([]interface{}{ratings[index],
movie.Title, movie.Year})
    if err != nil {
        panic(err)
    }
    statementRequests[index] = types.BatchStatementRequest{
        Statement: aws.String(
            fmt.Sprintf("UPDATE \"%v\" SET info.rating=? WHERE title=? AND year=?",
runner.TableName)),
        Parameters: params,
    }
}

_, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
&dynamodb.BatchExecuteStatementInput{
    Statements: statementRequests,
})
if err != nil {
    log.Printf("Couldn't update the batch of movies. Here's why: %v\n", err)
}
return err
}

```

Utilizar lotes de instrucciones DELETE para eliminar elementos.

```

// DeleteMovieBatch runs a batch of PartiQL DELETE statements to remove multiple
movies
// from the DynamoDB table.
func (runner PartiQLRunner) DeleteMovieBatch(ctx context.Context, movies []Movie)
error {
    statementRequests := make([]types.BatchStatementRequest, len(movies))
    for index, movie := range movies {
        params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year})
        if err != nil {
            panic(err)
        }
        statementRequests[index] = types.BatchStatementRequest{
            Statement: aws.String(
                fmt.Sprintf("DELETE FROM \"%v\" WHERE title=? AND year=?", runner.TableName)),
            Parameters: params,

```

```

    }
}

_, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
&dynamodb.BatchExecuteStatementInput{
    Statements: statementRequests,
})
if err != nil {
    log.Printf("Couldn't delete the batch of movies. Here's why: %v\n", err)
}
return err
}

```

Defina una estructura Película para utilizar en este ejemplo.

```

import (
    "archive/zip"
    "bytes"
    "encoding/json"
    "fmt"
    "io"
    "log"
    "net/http"

    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
// key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
// key,
// and Info is additional data.
type Movie struct {
    Title string          `dynamodbav:"title"`
    Year  int                 `dynamodbav:"year"`
    Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be

```

```
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
    title, err := attributevalue.Marshal(movie.Title)
    if err != nil {
        panic(err)
    }
    year, err := attributevalue.Marshal(movie.Year)
    if err != nil {
        panic(err)
    }
    return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
// example.
func (movie Movie) String() string {
    return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
        movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Para obtener más información sobre la API, consulta [BatchExecuteStatement](#) la Referencia AWS SDK para Go de la API.

BatchWriteItem

En el siguiente ejemplo de código, se muestra cómo utilizar BatchWriteItem.

SDK para Go V2

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 (
    "context"
    "errors"
```



```

"log"
"time"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
"github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
"github.com/aws/aws-sdk-go-v2/service/dynamodb"
"github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
// examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
    TableName      string
}

// AddMovieBatch adds a slice of movies to the DynamoDB table. The function sends
// batches of 25 movies to DynamoDB until all movies are added or it reaches the
// specified maximum.
func (basics TableBasics) AddMovieBatch(ctx context.Context, movies []Movie,
    maxMovies int) (int, error) {
    var err error
    var item map[string]types.AttributeValue
    written := 0
    batchSize := 25 // DynamoDB allows a maximum batch size of 25 items.
    start := 0
    end := start + batchSize
    for start < maxMovies && start < len(movies) {
        var writeReqs []types.WriteRequest
        if end > len(movies) {
            end = len(movies)
        }
        for _, movie := range movies[start:end] {
            item, err = attributevalue.MarshalMap(movie)
            if err != nil {
                log.Printf("Couldn't marshal movie %v for batch writing. Here's why: %v\n",
                    movie.Title, err)
            } else {
                writeReqs = append(
                    writeReqs,

```

```

        types.WriteRequest{PutRequest: &types.PutRequest{Item: item}},
    )
}
}
_, err = basics.DynamoDbClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
    RequestItems: map[string][]types.WriteRequest{basics.TableName: writeReqs})
if err != nil {
    log.Printf("Couldn't add a batch of movies to %v. Here's why: %v\n",
basics.TableName, err)
} else {
    written += len(writeReqs)
}
start = end
end += batchSize
}

return written, err
}

```

Defina una estructura Película para utilizar en este ejemplo.

```

import (
    "archive/zip"
    "bytes"
    "encoding/json"
    "fmt"
    "io"
    "log"
    "net/http"

    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
key,
// and Info is additional data.
type Movie struct {

```

```

Title string          `dynamodbav:"title"`
Year  int             `dynamodbav:"year"`
Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
    title, err := attributevalue.Marshal(movie.Title)
    if err != nil {
        panic(err)
    }
    year, err := attributevalue.Marshal(movie.Year)
    if err != nil {
        panic(err)
    }
    return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
// example.
func (movie Movie) String() string {
    return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
        movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}

```

- Para obtener más información sobre la API, consulta [BatchWriteItem](#) la Referencia AWS SDK para Go de la API.

CreateTable

En el siguiente ejemplo de código, se muestra cómo utilizar CreateTable.

SDK para Go V2

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 (
    "context"
    "errors"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
// examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
    TableName      string
}

// CreateMovieTable creates a DynamoDB table with a composite primary key defined as
// a string sort key named `title`, and a numeric partition key named `year`.
// This function uses NewTableExistsWaiter to wait for the table to be created by
// DynamoDB before it returns.
func (basics TableBasics) CreateMovieTable(ctx context.Context)
(*types.TableDescription, error) {
    var tableDesc *types.TableDescription
    table, err := basics.DynamoDbClient.CreateTable(ctx, &dynamodb.CreateTableInput{
        AttributeDefinitions: []types.AttributeDefinition{{
            AttributeName: aws.String("year"),
            AttributeType: types.ScalarAttributeTypeN,
        }}, {
            AttributeName: aws.String("title"),
            AttributeType: types.ScalarAttributeTypeS,
        }},
        KeySchema: []types.KeySchemaElement{{
            AttributeName: aws.String("year"),
            KeyType:      types.KeyTypeHash,
        }}, {

```

```

    AttributeName: aws.String("title"),
    KeyType:      types.KeyTypeRange,
  }},
  TableName:     aws.String(basics.TableName),
  BillingMode:    types.BillingModePayPerRequest,
})
if err != nil {
  log.Printf("Couldn't create table %v. Here's why: %v\n", basics.TableName, err)
} else {
  waiter := dynamodb.NewTableExistsWaiter(basics.DynamoDbClient)
  err = waiter.Wait(ctx, &dynamodb.DescribeTableInput{
    TableName: aws.String(basics.TableName)}, 5*time.Minute)
  if err != nil {
    log.Printf("Wait for table exists failed. Here's why: %v\n", err)
  }
  tableDesc = table.TableDescription
  log.Printf("Ccreating table test")
}
return tableDesc, err
}

```

- Para obtener más información sobre la API, consulta [CreateTable](#) la Referencia AWS SDK para Go de la API.

DeleteItem

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteItem.

SDK para Go V2

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 (
  "context"

```

```

"errors"
"log"
"time"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
"github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
"github.com/aws/aws-sdk-go-v2/service/dynamodb"
"github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
// examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
    TableName      string
}

// DeleteMovie removes a movie from the DynamoDB table.
func (basics TableBasics) DeleteMovie(ctx context.Context, movie Movie) error {
    _, err := basics.DynamoDbClient.DeleteItem(ctx, &dynamodb.DeleteItemInput{
        TableName: aws.String(basics.TableName), Key: movie.GetKey(),
    })
    if err != nil {
        log.Printf("Couldn't delete %v from the table. Here's why: %v\n", movie.Title,
            err)
    }
    return err
}

```

Defina una estructura Película para utilizar en este ejemplo.

```

import (
    "archive/zip"
    "bytes"
    "encoding/json"
    "fmt"

```

```

"io"
"log"
"net/http"

"github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
"github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
// key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
// key,
// and Info is additional data.
type Movie struct {
    Title string          `dynamodbav:"title"`
    Year  int                `dynamodbav:"year"`
    Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
    title, err := attributevalue.Marshal(movie.Title)
    if err != nil {
        panic(err)
    }
    year, err := attributevalue.Marshal(movie.Year)
    if err != nil {
        panic(err)
    }
    return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
// example.
func (movie Movie) String() string {
    return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
        movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}

```

- Para obtener más información sobre la API, consulta [DeleteItem](#) la Referencia AWS SDK para Go de la API.

DeleteTable

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteTable.

SDK para Go V2

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 (
    "context"
    "errors"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
// examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
    TableName      string
}

// DeleteTable deletes the DynamoDB table and all of its data.
func (basics TableBasics) DeleteTable(ctx context.Context) error {
    _, err := basics.DynamoDbClient.DeleteTable(ctx, &dynamodb.DeleteTableInput{
```



```
    TableName: aws.String(basics.TableName)})
if err != nil {
    log.Printf("Couldn't delete table %v. Here's why: %v\n", basics.TableName, err)
}
return err
}
```

- Para obtener más información sobre la API, consulta [DeleteTable](#) la Referencia AWS SDK para Go de la API.

DescribeTable

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeTable.

SDK para Go V2

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 (
    "context"
    "errors"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
// examples.
// It contains a DynamoDB service client that is used to act on the specified table.
```

```
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
    TableName      string
}

// TableExists determines whether a DynamoDB table exists.
func (basics TableBasics) TableExists(ctx context.Context) (bool, error) {
    exists := true
    _, err := basics.DynamoDbClient.DescribeTable(
        ctx, &dynamodb.DescribeTableInput{TableName: aws.String(basics.TableName)},
    )
    if err != nil {
        var notFoundEx *types.ResourceNotFoundException
        if errors.As(err, &notFoundEx) {
            log.Printf("Table %v does not exist.\n", basics.TableName)
            err = nil
        } else {
            log.Printf("Couldn't determine existence of table %v. Here's why: %v\n",
                basics.TableName, err)
        }
        exists = false
    }
    return exists, err
}
```

- Para obtener más información sobre la API, consulta [DescribeTable](#) la Referencia AWS SDK para Go de la API.

ExecuteStatement

En el siguiente ejemplo de código, se muestra cómo utilizar ExecuteStatement.

SDK para Go V2

 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](#).

Defina una estructura de receptor de funciones para el ejemplo.

```
import (  
    "context"  
    "fmt"  
    "log"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"  
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"  
    "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"  
)  
  
// PartiQLRunner encapsulates the Amazon DynamoDB service actions used in the  
// PartiQL examples. It contains a DynamoDB service client that is used to act on  
// the  
// specified table.  
type PartiQLRunner struct {  
    DynamoDbClient *dynamodb.Client  
    TableName      string  
}
```

Utilizar una instrucción INSERT para agregar un elemento.

```
// AddMovie runs a PartiQL INSERT statement to add a movie to the DynamoDB table.  
func (runner PartiQLRunner) AddMovie(ctx context.Context, movie Movie) error {  
    params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year,  
    movie.Info})  
    if err != nil {  
        panic(err)  
    }
```

```

}
_, err = runner.DynamoDbClient.ExecuteStatement(ctx,
&dynamodb.ExecuteStatementInput{
    Statement: aws.String(
        fmt.Sprintf("INSERT INTO \"%v\" VALUE {'title': ?, 'year': ?, 'info': ?}",
            runner.TableName)),
    Parameters: params,
})
if err != nil {
    log.Printf("Couldn't insert an item with PartiQL. Here's why: %v\n", err)
}
return err
}

```

Utilizar una instrucción SELECT para obtener un elemento.

```

// GetMovie runs a PartiQL SELECT statement to get a movie from the DynamoDB table
// by
// title and year.
func (runner PartiQLRunner) GetMovie(ctx context.Context, title string, year int)
(Movie, error) {
    var movie Movie
    params, err := attributevalue.MarshalList([]interface{}{title, year})
    if err != nil {
        panic(err)
    }
    response, err := runner.DynamoDbClient.ExecuteStatement(ctx,
&dynamodb.ExecuteStatementInput{
    Statement: aws.String(
        fmt.Sprintf("SELECT * FROM \"%v\" WHERE title=? AND year=?",
            runner.TableName)),
    Parameters: params,
})
    if err != nil {
        log.Printf("Couldn't get info about %v. Here's why: %v\n", title, err)
    } else {
        err = attributevalue.UnmarshalMap(response.Items[0], &movie)
        if err != nil {
            log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
        }
    }
}

```

```

}
return movie, err
}

```

Utilizar una instrucción SELECT para obtener una lista de elementos y proyectar los resultados.

```

// GetAllMovies runs a PartiQL SELECT statement to get all movies from the DynamoDB
// table.
// pageSize is not typically required and is used to show how to paginate the
// results.
// The results are projected to return only the title and rating of each movie.
func (runner PartiQLRunner) GetAllMovies(ctx context.Context, pageSize int32)
([]map[string]interface{}, error) {
    var output []map[string]interface{}
    var response *dynamodb.ExecuteStatementOutput
    var err error
    var nextToken *string
    for moreData := true; moreData; {
        response, err = runner.DynamoDbClient.ExecuteStatement(ctx,
            &dynamodb.ExecuteStatementInput{
                Statement: aws.String(
                    fmt.Sprintf("SELECT title, info.rating FROM \"%v\"", runner.TableName)),
                Limit:      aws.Int32(pageSize),
                NextToken: nextToken,
            })
        if err != nil {
            log.Printf("Couldn't get movies. Here's why: %v\n", err)
            moreData = false
        } else {
            var pageOutput []map[string]interface{}
            err = attributevalue.UnmarshalListOfMaps(response.Items, &pageOutput)
            if err != nil {
                log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
            } else {
                log.Printf("Got a page of length %v.\n", len(response.Items))
                output = append(output, pageOutput...)
            }
            nextToken = response.NextToken
            moreData = nextToken != nil
        }
    }
}

```

```

}
return output, err
}

```

Utilizar una instrucción UPDATE para actualizar un elemento.

```

// UpdateMovie runs a PartiQL UPDATE statement to update the rating of a movie that
// already exists in the DynamoDB table.
func (runner PartiQLRunner) UpdateMovie(ctx context.Context, movie Movie, rating
float64) error {
    params, err := attributevalue.MarshalList([]interface{}{rating, movie.Title,
movie.Year})
    if err != nil {
        panic(err)
    }
    _, err = runner.DynamoDbClient.ExecuteStatement(ctx,
&dynamodb.ExecuteStatementInput{
    Statement: aws.String(
        fmt.Sprintf("UPDATE \"%v\" SET info.rating=? WHERE title=? AND year=?",
            runner.TableName)),
    Parameters: params,
})
    if err != nil {
        log.Printf("Couldn't update movie %v. Here's why: %v\n", movie.Title, err)
    }
    return err
}

```

Utilizar una instrucción DELETE para eliminar un elemento.

```

// DeleteMovie runs a PartiQL DELETE statement to remove a movie from the DynamoDB
table.
func (runner PartiQLRunner) DeleteMovie(ctx context.Context, movie Movie) error {
    params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year})
    if err != nil {
        panic(err)
    }
}

```

```

_, err = runner.DynamoDbClient.ExecuteStatement(ctx,
&dynamodb.ExecuteStatementInput{
    Statement: aws.String(
        fmt.Sprintf("DELETE FROM \"%v\" WHERE title=? AND year=?",
            runner.TableName)),
    Parameters: params,
})
if err != nil {
    log.Printf("Couldn't delete %v from the table. Here's why: %v\n", movie.Title,
err)
}
return err
}

```

Defina una estructura Película para utilizar en este ejemplo.

```

import (
    "archive/zip"
    "bytes"
    "encoding/json"
    "fmt"
    "io"
    "log"
    "net/http"

    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
key,
// and Info is additional data.
type Movie struct {
    Title string          `dynamodbav:"title"`
    Year  int                 `dynamodbav:"year"`
    Info  map[string]interface{} `dynamodbav:"info"`
}

```

```
// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
    title, err := attributevalue.Marshal(movie.Title)
    if err != nil {
        panic(err)
    }
    year, err := attributevalue.Marshal(movie.Year)
    if err != nil {
        panic(err)
    }
    return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
// example.
func (movie Movie) String() string {
    return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
        movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Para obtener más información sobre la API, consulta [ExecuteStatement](#) la Referencia AWS SDK para Go de la API.

GetItem

En el siguiente ejemplo de código, se muestra cómo utilizar `GetItem`.

SDK para Go V2

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 (
    "context"
```



```

"errors"
"log"
"time"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
"github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
"github.com/aws/aws-sdk-go-v2/service/dynamodb"
"github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
// examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
    TableName      string
}

// GetMovie gets movie data from the DynamoDB table by using the primary composite
// key
// made of title and year.
func (basics TableBasics) GetMovie(ctx context.Context, title string, year int)
(Movie, error) {
    movie := Movie{Title: title, Year: year}
    response, err := basics.DynamoDbClient.GetItem(ctx, &dynamodb.GetItemInput{
        Key: movie.GetKey(), TableName: aws.String(basics.TableName),
    })
    if err != nil {
        log.Printf("Couldn't get info about %v. Here's why: %v\n", title, err)
    } else {
        err = attributevalue.UnmarshalMap(response.Item, &movie)
        if err != nil {
            log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
        }
    }
    return movie, err
}

```

Defina una estructura Película para utilizar en este ejemplo.

```
import (
    "archive/zip"
    "bytes"
    "encoding/json"
    "fmt"
    "io"
    "log"
    "net/http"

    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
// key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
// key,
// and Info is additional data.
type Movie struct {
    Title string          `dynamodbav:"title"`
    Year  int                 `dynamodbav:"year"`
    Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
    title, err := attributevalue.Marshal(movie.Title)
    if err != nil {
        panic(err)
    }
    year, err := attributevalue.Marshal(movie.Year)
    if err != nil {
        panic(err)
    }
    return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
// example.
func (movie Movie) String() string {
```

```
return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
    movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Para obtener más información sobre la API, consulta [GetItem](#) la Referencia AWS SDK para Go de la API.

ListTables

En el siguiente ejemplo de código, se muestra cómo utilizar ListTables.

SDK para Go V2

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 (
    "context"
    "errors"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
// examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
    TableName      string
}
```

```

}

// ListTables lists the DynamoDB table names for the current account.
func (basics TableBasics) ListTables(ctx context.Context) ([]string, error) {
    var tableNames []string
    var output *dynamodb.ListTablesOutput
    var err error
    tablePaginator := dynamodb.NewListTablesPaginator(basics.DynamoDbClient,
        &dynamodb.ListTablesInput{})
    for tablePaginator.HasMorePages() {
        output, err = tablePaginator.NextPage(ctx)
        if err != nil {
            log.Printf("Couldn't list tables. Here's why: %v\n", err)
            break
        } else {
            tableNames = append(tableNames, output.TableNames...)
        }
    }
    return tableNames, err
}

```

- Para obtener más información sobre la API, consulta [ListTables](#) la Referencia AWS SDK para Go de la API.

PutItem

En el siguiente ejemplo de código, se muestra cómo utilizar PutItem.

SDK para Go V2

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 (
    "context"
    "errors"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
// examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
    TableName      string
}

// AddMovie adds a movie the DynamoDB table.
func (basics TableBasics) AddMovie(ctx context.Context, movie Movie) error {
    item, err := attributevalue.MarshalMap(movie)
    if err != nil {
        panic(err)
    }
    _, err = basics.DynamoDbClient.PutItem(ctx, &dynamodb.PutItemInput{
        TableName: aws.String(basics.TableName), Item: item,
    })
    if err != nil {
        log.Printf("Couldn't add item to table. Here's why: %v\n", err)
    }
    return err
}

```

Defina una estructura Película para utilizar en este ejemplo.

```

import (
    "archive/zip"
    "bytes"
    "encoding/json"
    "fmt"
    "io"
    "log"
    "net/http"

    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
// key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
// key,
// and Info is additional data.
type Movie struct {
    Title string          `dynamodbav:"title"`
    Year  int                 `dynamodbav:"year"`
    Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
    title, err := attributevalue.Marshal(movie.Title)
    if err != nil {
        panic(err)
    }
    year, err := attributevalue.Marshal(movie.Year)
    if err != nil {
        panic(err)
    }
    return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
// example.
func (movie Movie) String() string {
    return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
        movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}

```

- Para obtener más información sobre la API, consulta [PutItem](#) la Referencia AWS SDK para Go de la API.

Query

En el siguiente ejemplo de código, se muestra cómo utilizar Query.

SDK para Go V2

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 (  
    "context"  
    "errors"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"  
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"  
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"  
    "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"  
)  
  
// TableBasics encapsulates the Amazon DynamoDB service actions used in the  
// examples.  
// It contains a DynamoDB service client that is used to act on the specified table.  
type TableBasics struct {  
    DynamoDbClient *dynamodb.Client  
    TableName      string  
}
```

```

// Query gets all movies in the DynamoDB table that were released in the specified
// year.
// The function uses the `expression` package to build the key condition expression
// that is used in the query.
func (basics TableBasics) Query(ctx context.Context, releaseYear int) ([]Movie,
error) {
    var err error
    var response *dynamodb.QueryOutput
    var movies []Movie
    keyEx := expression.Key("year").Equal(expression.Value(releaseYear))
    expr, err := expression.NewBuilder().WithKeyCondition(keyEx).Build()
    if err != nil {
        log.Printf("Couldn't build expression for query. Here's why: %v\n", err)
    } else {
        queryPaginator := dynamodb.NewQueryPaginator(basics.DynamoDbClient,
        &dynamodb.QueryInput{
            TableName:          aws.String(basics.TableName),
            ExpressionAttributeNames: expr.Names(),
            ExpressionAttributeValues: expr.Values(),
            KeyConditionExpression: expr.KeyCondition(),
        })
        for queryPaginator.HasMorePages() {
            response, err = queryPaginator.NextPage(ctx)
            if err != nil {
                log.Printf("Couldn't query for movies released in %v. Here's why: %v\n",
                releaseYear, err)
                break
            } else {
                var moviePage []Movie
                err = attributevalue.UnmarshalListOfMaps(response.Items, &moviePage)
                if err != nil {
                    log.Printf("Couldn't unmarshal query response. Here's why: %v\n", err)
                    break
                } else {
                    movies = append(movies, moviePage...)
                }
            }
        }
    }
    return movies, err
}

```


Defina una estructura Película para utilizar en este ejemplo.

```
import (
    "archive/zip"
    "bytes"
    "encoding/json"
    "fmt"
    "io"
    "log"
    "net/http"

    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
// key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
// key,
// and Info is additional data.
type Movie struct {
    Title string          `dynamodbav:"title"`
    Year  int                `dynamodbav:"year"`
    Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
    title, err := attributevalue.Marshal(movie.Title)
    if err != nil {
        panic(err)
    }
    year, err := attributevalue.Marshal(movie.Year)
    if err != nil {
        panic(err)
    }
    return map[string]types.AttributeValue{"title": title, "year": year}
}
```

```
// String returns the title, year, rating, and plot of a movie, formatted for the
// example.
func (movie Movie) String() string {
    return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
        movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Para obtener información sobre la API, consulte [Query](#) en la referencia de la API de AWS SDK para Go .

Scan

En el siguiente ejemplo de código, se muestra cómo utilizar Scan.

SDK para Go V2

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
import (
    "context"
    "errors"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
// examples.
// It contains a DynamoDB service client that is used to act on the specified table.
```

```

type TableBasics struct {
    DynamoDbClient *dynamodb.Client
    TableName      string
}

// Scan gets all movies in the DynamoDB table that were released in a range of years
// and projects them to return a reduced set of fields.
// The function uses the `expression` package to build the filter and projection
// expressions.
func (basics TableBasics) Scan(ctx context.Context, startYear int, endYear int)
    ([]Movie, error) {
    var movies []Movie
    var err error
    var response *dynamodb.ScanOutput
    filtEx := expression.Name("year").Between(expression.Value(startYear),
        expression.Value(endYear))
    projEx := expression.NamesList(
        expression.Name("year"), expression.Name("title"), expression.Name("info.rating"))
    expr, err :=
        expression.NewBuilder().WithFilter(filtEx).WithProjection(projEx).Build()
    if err != nil {
        log.Printf("Couldn't build expressions for scan. Here's why: %v\n", err)
    } else {
        scanPaginator := dynamodb.NewScanPaginator(basics.DynamoDbClient,
            &dynamodb.ScanInput{
                TableName:          aws.String(basics.TableName),
                ExpressionAttributeNames: expr.Names(),
                ExpressionAttributeValues: expr.Values(),
                FilterExpression:    expr.Filter(),
                ProjectionExpression: expr.Projection(),
            })
        for scanPaginator.HasMorePages() {
            response, err = scanPaginator.NextPage(ctx)
            if err != nil {
                log.Printf("Couldn't scan for movies released between %v and %v. Here's why: %v\n",
                    startYear, endYear, err)
                break
            } else {
                var moviePage []Movie
                err = attributevalue.UnmarshalListOfMaps(response.Items, &moviePage)
                if err != nil {

```

```

    log.Printf("Couldn't unmarshal query response. Here's why: %v\n", err)
    break
} else {
    movies = append(movies, moviePage...)
}
}
}
}
return movies, err
}

```

Defina una estructura Película para utilizar en este ejemplo.

```

import (
    "archive/zip"
    "bytes"
    "encoding/json"
    "fmt"
    "io"
    "log"
    "net/http"

    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
// key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
// key,
// and Info is additional data.
type Movie struct {
    Title string          `dynamodbav:"title"`
    Year  int                 `dynamodbav:"year"`
    Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {

```

```

title, err := attributevalue.Marshal(movie.Title)
if err != nil {
    panic(err)
}
year, err := attributevalue.Marshal(movie.Year)
if err != nil {
    panic(err)
}
return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
// example.
func (movie Movie) String() string {
    return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
        movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}

```

- Para obtener información sobre la API, consulte [Scan](#) en la referencia de la API de AWS SDK para Go .

UpdateItem

En el siguiente ejemplo de código, se muestra cómo utilizar UpdateItem.

SDK para Go V2

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```

import (
    "context"
    "errors"
    "log"
    "time"

```

```

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
"github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
"github.com/aws/aws-sdk-go-v2/service/dynamodb"
"github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// TableBasics encapsulates the Amazon DynamoDB service actions used in the
// examples.
// It contains a DynamoDB service client that is used to act on the specified table.
type TableBasics struct {
    DynamoDbClient *dynamodb.Client
    TableName      string
}

// UpdateMovie updates the rating and plot of a movie that already exists in the
// DynamoDB table. This function uses the `expression` package to build the update
// expression.
func (basics TableBasics) UpdateMovie(ctx context.Context, movie Movie)
    (map[string]map[string]interface{}, error) {
    var err error
    var response *dynamodb.UpdateItemOutput
    var attributeMap map[string]map[string]interface{}
    update := expression.Set(expression.Name("info.rating"),
        expression.Value(movie.Info["rating"]))
    update.Set(expression.Name("info.plot"), expression.Value(movie.Info["plot"]))
    expr, err := expression.NewBuilder().WithUpdate(update).Build()
    if err != nil {
        log.Printf("Couldn't build expression for update. Here's why: %v\n", err)
    } else {
        response, err = basics.DynamoDbClient.UpdateItem(ctx, &dynamodb.UpdateItemInput{
            TableName:      aws.String(basics.TableName),
            Key:            movie.GetKey(),
            ExpressionAttributeNames: expr.Names(),
            ExpressionAttributeValues: expr.Values(),
            UpdateExpression: expr.Update(),
            ReturnValues:    types.ReturnValueUpdatedNew,
        })
    }
    if err != nil {
        log.Printf("Couldn't update movie %v. Here's why: %v\n", movie.Title, err)
    } else {

```

```

    err = attributevalue.UnmarshalMap(response.Attributes, &attributeMap)
    if err != nil {
        log.Printf("Couldn't unmarshall update response. Here's why: %v\n", err)
    }
}
return attributeMap, err
}

```

Defina una estructura Película para utilizar en este ejemplo.

```

import (
    "archive/zip"
    "bytes"
    "encoding/json"
    "fmt"
    "io"
    "log"
    "net/http"

    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
// key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
// key,
// and Info is additional data.
type Movie struct {
    Title string          `dynamodbav:"title"`
    Year  int                 `dynamodbav:"year"`
    Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
    title, err := attributevalue.Marshal(movie.Title)
    if err != nil {

```

```
    panic(err)
}
year, err := attributevalue.Marshal(movie.Year)
if err != nil {
    panic(err)
}
return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
// example.
func (movie Movie) String() string {
    return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
        movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}
```

- Para obtener más información sobre la API, consulta [UpdateItem](#) la Referencia AWS SDK para Go de la API.

Escenarios

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 Go V2

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](#).

Ejecutar un escenario que crea una tabla y ejecuta lotes de consultas PartiQL.

```

import (
    "context"
    "fmt"
    "log"
    "strings"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/dynamodb/actions"
)

// RunPartiQLBatchScenario shows you how to use the AWS SDK for Go
// to run batches of PartiQL statements to query a table that stores data about
// movies.
//
// - Use batches of PartiQL statements to add, get, update, and delete data for
//   individual movies.
//
// This example creates an Amazon DynamoDB service client from the specified
// sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// This example creates and deletes a DynamoDB table to use during the scenario.
func RunPartiQLBatchScenario(ctx context.Context, sdkConfig aws.Config, tableName
    string) {
    defer func() {
        if r := recover(); r != nil {
            fmt.Printf("Something went wrong with the demo.")
        }
    }()

    log.Println(strings.Repeat("-", 88))
    log.Println("Welcome to the Amazon DynamoDB PartiQL batch demo.")
    log.Println(strings.Repeat("-", 88))

    tableBasics := actions.TableBasics{
        DynamoDbClient: dynamodb.NewFromConfig(sdkConfig),
        TableName:      tableName,
    }
    runner := actions.PartiQLRunner{

```

```

    DynamoDbClient: dynamodb.NewFromConfig(sdkConfig),
    TableName:      tableName,
}

exists, err := tableBasics.TableExists(ctx)
if err != nil {
    panic(err)
}
if !exists {
    log.Printf("Creating table %v...\n", tableName)
    _, err = tableBasics.CreateMovieTable(ctx)
    if err != nil {
        panic(err)
    } else {
        log.Printf("Created table %v.\n", tableName)
    }
} else {
    log.Printf("Table %v already exists.\n", tableName)
}
log.Println(strings.Repeat("-", 88))

currentYear, _, _ := time.Now().Date()
customMovies := []actions.Movie{
    Title: "House PartiQL",
    Year:  currentYear - 5,
    Info: map[string]interface{}{
        "plot":  "Wacky high jinks result from querying a mysterious database.",
        "rating": 8.5}}, {
    Title: "House PartiQL 2",
    Year:  currentYear - 3,
    Info: map[string]interface{}{
        "plot":  "Moderate high jinks result from querying another mysterious
database.",
        "rating": 6.5}}, {
    Title: "House PartiQL 3",
    Year:  currentYear - 1,
    Info: map[string]interface{}{
        "plot":  "Tepid high jinks result from querying yet another mysterious
database.",
        "rating": 2.5},
},
}

log.Printf("Inserting a batch of movies into table '%v'.\n", tableName)

```

```

err = runner.AddMovieBatch(ctx, customMovies)
if err == nil {
    log.Printf("Added %v movies to the table.\n", len(customMovies))
}
log.Println(strings.Repeat("-", 88))

log.Println("Getting data for a batch of movies.")
movies, err := runner.GetMovieBatch(ctx, customMovies)
if err == nil {
    for _, movie := range movies {
        log.Println(movie)
    }
}
log.Println(strings.Repeat("-", 88))

newRatings := []float64{7.7, 4.4, 1.1}
log.Println("Updating a batch of movies with new ratings.")
err = runner.UpdateMovieBatch(ctx, customMovies, newRatings)
if err == nil {
    log.Printf("Updated %v movies with new ratings.\n", len(customMovies))
}
log.Println(strings.Repeat("-", 88))

log.Println("Getting projected data from the table to verify our update.")
log.Println("Using a page size of 2 to demonstrate paging.")
projections, err := runner.GetAllMovies(ctx, 2)
if err == nil {
    log.Println("All movies:")
    for _, projection := range projections {
        log.Println(projection)
    }
}
log.Println(strings.Repeat("-", 88))

log.Println("Deleting a batch of movies.")
err = runner.DeleteMovieBatch(ctx, customMovies)
if err == nil {
    log.Printf("Deleted %v movies.\n", len(customMovies))
}

err = tableBasics.DeleteTable(ctx)
if err == nil {
    log.Printf("Deleted table %v.\n", tableBasics.TableName)
}

```

```
log.Println(strings.Repeat("-", 88))
log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}
```

Defina una estructura Película para utilizar en este ejemplo.

```
import (
    "archive/zip"
    "bytes"
    "encoding/json"
    "fmt"
    "io"
    "log"
    "net/http"

    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
// key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
// key,
// and Info is additional data.
type Movie struct {
    Title string          `dynamodbav:"title"`
    Year  int                 `dynamodbav:"year"`
    Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {
    title, err := attributevalue.Marshal(movie.Title)
    if err != nil {
        panic(err)
    }
    year, err := attributevalue.Marshal(movie.Year)
```

```

    if err != nil {
        panic(err)
    }
    return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
// example.
func (movie Movie) String() string {
    return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
        movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}

```

Crear una estructura y métodos que ejecuten instrucciones PartiQL.

```

import (
    "context"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// PartiQLRunner encapsulates the Amazon DynamoDB service actions used in the
// PartiQL examples. It contains a DynamoDB service client that is used to act on
// the
// specified table.
type PartiQLRunner struct {
    DynamoDbClient *dynamodb.Client
    TableName      string
}

// AddMovieBatch runs a batch of PartiQL INSERT statements to add multiple movies to
// the
// DynamoDB table.

```

```

func (runner PartiQLRunner) AddMovieBatch(ctx context.Context, movies []Movie) error
{
    statementRequests := make([]types.BatchStatementRequest, len(movies))
    for index, movie := range movies {
        params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year,
movie.Info})
        if err != nil {
            panic(err)
        }
        statementRequests[index] = types.BatchStatementRequest{
            Statement: aws.String(fmt.Sprintf(
                "INSERT INTO \"%v\" VALUE {'title': ?, 'year': ?, 'info': ?}",
runner.TableName)),
            Parameters: params,
        }
    }

    _, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
&dynamodb.BatchExecuteStatementInput{
    Statements: statementRequests,
})
    if err != nil {
        log.Printf("Couldn't insert a batch of items with PartiQL. Here's why: %v\n", err)
    }
    return err
}

// GetMovieBatch runs a batch of PartiQL SELECT statements to get multiple movies
// from
// the DynamoDB table by title and year.
func (runner PartiQLRunner) GetMovieBatch(ctx context.Context, movies []Movie)
([]Movie, error) {
    statementRequests := make([]types.BatchStatementRequest, len(movies))
    for index, movie := range movies {
        params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year})
        if err != nil {
            panic(err)
        }
        statementRequests[index] = types.BatchStatementRequest{
            Statement: aws.String(
                fmt.Sprintf("SELECT * FROM \"%v\" WHERE title=? AND year=?", runner.TableName)),
            Parameters: params,

```

```

    }
}

output, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
&dynamodb.BatchExecuteStatementInput{
    Statements: statementRequests,
})
var outMovies []Movie
if err != nil {
    log.Printf("Couldn't get a batch of items with PartiQL. Here's why: %v\n", err)
} else {
    for _, response := range output.Responses {
        var movie Movie
        err = attributevalue.UnmarshalMap(response.Item, &movie)
        if err != nil {
            log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
        } else {
            outMovies = append(outMovies, movie)
        }
    }
}
return outMovies, err
}

// GetAllMovies runs a PartiQL SELECT statement to get all movies from the DynamoDB
// table.
// pageSize is not typically required and is used to show how to paginate the
// results.
// The results are projected to return only the title and rating of each movie.
func (runner PartiQLRunner) GetAllMovies(ctx context.Context, pageSize int32)
([]map[string]interface{}, error) {
    var output []map[string]interface{}
    var response *dynamodb.ExecuteStatementOutput
    var err error
    var nextToken *string
    for moreData := true; moreData; {
        response, err = runner.DynamoDbClient.ExecuteStatement(ctx,
&dynamodb.ExecuteStatementInput{
            Statement: aws.String(
                fmt.Sprintf("SELECT title, info.rating FROM \"%v\"", runner.TableName)),
            Limit:      aws.Int32(pageSize),
            NextToken: nextToken,

```

```

    })
    if err != nil {
        log.Printf("Couldn't get movies. Here's why: %v\n", err)
        moreData = false
    } else {
        var pageOutput []map[string]interface{}
        err = attributevalue.UnmarshalListOfMaps(response.Items, &pageOutput)
        if err != nil {
            log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
        } else {
            log.Printf("Got a page of length %v.\n", len(response.Items))
            output = append(output, pageOutput...)
        }
        nextToken = response.NextToken
        moreData = nextToken != nil
    }
}
return output, err
}

// UpdateMovieBatch runs a batch of PartiQL UPDATE statements to update the rating
// of
// multiple movies that already exist in the DynamoDB table.
func (runner PartiQLRunner) UpdateMovieBatch(ctx context.Context, movies []Movie,
ratings []float64) error {
    statementRequests := make([]types.BatchStatementRequest, len(movies))
    for index, movie := range movies {
        params, err := attributevalue.MarshalList([]interface{}{ratings[index],
movie.Title, movie.Year})
        if err != nil {
            panic(err)
        }
        statementRequests[index] = types.BatchStatementRequest{
            Statement: aws.String(
                fmt.Sprintf("UPDATE \"%v\" SET info.rating=? WHERE title=? AND year=?",
runner.TableName)),
            Parameters: params,
        }
    }

    _, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
&dynamodb.BatchExecuteStatementInput{

```



```

    Statements: statementRequests,
  })
  if err != nil {
    log.Printf("Couldn't update the batch of movies. Here's why: %v\n", err)
  }
  return err
}

// DeleteMovieBatch runs a batch of PartiQL DELETE statements to remove multiple
// movies
// from the DynamoDB table.
func (runner PartiQLRunner) DeleteMovieBatch(ctx context.Context, movies []Movie)
error {
  statementRequests := make([]types.BatchStatementRequest, len(movies))
  for index, movie := range movies {
    params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year})
    if err != nil {
      panic(err)
    }
    statementRequests[index] = types.BatchStatementRequest{
      Statement: aws.String(
        fmt.Sprintf("DELETE FROM \"%v\" WHERE title=? AND year=?", runner.TableName)),
      Parameters: params,
    }
  }

  _, err := runner.DynamoDbClient.BatchExecuteStatement(ctx,
    &dynamodb.BatchExecuteStatementInput{
      Statements: statementRequests,
    })
  if err != nil {
    log.Printf("Couldn't delete the batch of movies. Here's why: %v\n", err)
  }
  return err
}

```

- Para obtener más información sobre la API, consulta [BatchExecuteStatement](#) la Referencia AWS SDK para Go 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 Go V2

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](#).

Ejecutar un escenario que crea una tabla y ejecuta consultas PartiQL.

```
import (  
    "context"  
    "fmt"  
    "log"  
    "strings"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"  
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/dynamodb/actions"  
)  
  
// RunPartiQLSingleScenario shows you how to use the AWS SDK for Go  
// to use PartiQL to query a table that stores data about movies.  
//  
// * Use PartiQL statements to add, get, update, and delete data for individual  
//   movies.  
//  
// This example creates an Amazon DynamoDB service client from the specified  
// sdkConfig so that  
// you can replace it with a mocked or stubbed config for unit testing.
```

```
//
// This example creates and deletes a DynamoDB table to use during the scenario.
func RunPartiQLSingleScenario(ctx context.Context, sdkConfig aws.Config, tableName
string) {
defer func() {
if r := recover(); r != nil {
fmt.Printf("Something went wrong with the demo.")
}
}()

log.Println(strings.Repeat("-", 88))
log.Println("Welcome to the Amazon DynamoDB PartiQL single action demo.")
log.Println(strings.Repeat("-", 88))

tableBasics := actions.TableBasics{
DynamoDbClient: dynamodb.NewFromConfig(sdkConfig),
TableName:      tableName,
}
runner := actions.PartiQLRunner{
DynamoDbClient: dynamodb.NewFromConfig(sdkConfig),
TableName:      tableName,
}

exists, err := tableBasics.TableExists(ctx)
if err != nil {
panic(err)
}
if !exists {
log.Printf("Creating table %v...\n", tableName)
_, err = tableBasics.CreateMovieTable(ctx)
if err != nil {
panic(err)
} else {
log.Printf("Created table %v.\n", tableName)
}
} else {
log.Printf("Table %v already exists.\n", tableName)
}
log.Println(strings.Repeat("-", 88))

currentYear, _, _ := time.Now().Date()
customMovie := actions.Movie{
Title: "24 Hour PartiQL People",
Year:  currentYear,
```

```
Info: map[string]interface{}{
    "plot":  "A group of data developers discover a new query language they can't
stop using.",
    "rating": 9.9,
},
}

log.Printf("Inserting movie '%v' released in %v.", customMovie.Title,
customMovie.Year)
err = runner.AddMovie(ctx, customMovie)
if err == nil {
    log.Printf("Added %v to the movie table.\n", customMovie.Title)
}
log.Println(strings.Repeat("-", 88))

log.Printf("Getting data for movie '%v' released in %v.", customMovie.Title,
customMovie.Year)
movie, err := runner.GetMovie(ctx, customMovie.Title, customMovie.Year)
if err == nil {
    log.Println(movie)
}
log.Println(strings.Repeat("-", 88))

newRating := 6.6
log.Printf("Updating movie '%v' with a rating of %v.", customMovie.Title,
newRating)
err = runner.UpdateMovie(ctx, customMovie, newRating)
if err == nil {
    log.Printf("Updated %v with a new rating.\n", customMovie.Title)
}
log.Println(strings.Repeat("-", 88))

log.Printf("Getting data again to verify the update.")
movie, err = runner.GetMovie(ctx, customMovie.Title, customMovie.Year)
if err == nil {
    log.Println(movie)
}
log.Println(strings.Repeat("-", 88))

log.Printf("Deleting movie '%v'.\n", customMovie.Title)
err = runner.DeleteMovie(ctx, customMovie)
if err == nil {
    log.Printf("Deleted %v.\n", customMovie.Title)
}
}
```

```

err = tableBasics.DeleteTable(ctx)
if err == nil {
    log.Printf("Deleted table %v.\n", tableBasics.TableName)
}

log.Println(strings.Repeat("-", 88))
log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}

```

Defina una estructura Película para utilizar en este ejemplo.

```

import (
    "archive/zip"
    "bytes"
    "encoding/json"
    "fmt"
    "io"
    "log"
    "net/http"

    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// Movie encapsulates data about a movie. Title and Year are the composite primary
// key
// of the movie in Amazon DynamoDB. Title is the sort key, Year is the partition
// key,
// and Info is additional data.
type Movie struct {
    Title string          `dynamodbav:"title"`
    Year  int                 `dynamodbav:"year"`
    Info  map[string]interface{} `dynamodbav:"info"`
}

// GetKey returns the composite primary key of the movie in a format that can be
// sent to DynamoDB.
func (movie Movie) GetKey() map[string]types.AttributeValue {

```

```

    title, err := attributevalue.Marshal(movie.Title)
    if err != nil {
        panic(err)
    }
    year, err := attributevalue.Marshal(movie.Year)
    if err != nil {
        panic(err)
    }
    return map[string]types.AttributeValue{"title": title, "year": year}
}

// String returns the title, year, rating, and plot of a movie, formatted for the
// example.
func (movie Movie) String() string {
    return fmt.Sprintf("%v\n\tReleased: %v\n\tRating: %v\n\tPlot: %v\n",
        movie.Title, movie.Year, movie.Info["rating"], movie.Info["plot"])
}

```

Crear una estructura y métodos que ejecuten instrucciones PartiQL.

```

import (
    "context"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// PartiQLRunner encapsulates the Amazon DynamoDB service actions used in the
// PartiQL examples. It contains a DynamoDB service client that is used to act on
// the
// specified table.
type PartiQLRunner struct {
    DynamoDbClient *dynamodb.Client
    TableName      string
}

```

```

// AddMovie runs a PartiQL INSERT statement to add a movie to the DynamoDB table.
func (runner PartiQLRunner) AddMovie(ctx context.Context, movie Movie) error {
    params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year,
        movie.Info})
    if err != nil {
        panic(err)
    }
    _, err = runner.DynamoDbClient.ExecuteStatement(ctx,
        &dynamodb.ExecuteStatementInput{
            Statement: aws.String(
                fmt.Sprintf("INSERT INTO \"%v\" VALUE {'title': ?, 'year': ?, 'info': ?}",
                    runner.TableName)),
            Parameters: params,
        })
    if err != nil {
        log.Printf("Couldn't insert an item with PartiQL. Here's why: %v\n", err)
    }
    return err
}

// GetMovie runs a PartiQL SELECT statement to get a movie from the DynamoDB table
// by
// title and year.
func (runner PartiQLRunner) GetMovie(ctx context.Context, title string, year int)
(Movie, error) {
    var movie Movie
    params, err := attributevalue.MarshalList([]interface{}{title, year})
    if err != nil {
        panic(err)
    }
    response, err := runner.DynamoDbClient.ExecuteStatement(ctx,
        &dynamodb.ExecuteStatementInput{
            Statement: aws.String(
                fmt.Sprintf("SELECT * FROM \"%v\" WHERE title=? AND year=?",
                    runner.TableName)),
            Parameters: params,
        })
    if err != nil {
        log.Printf("Couldn't get info about %v. Here's why: %v\n", title, err)
    } else {

```

```

    err = attributevalue.UnmarshalMap(response.Items[0], &movie)
    if err != nil {
        log.Printf("Couldn't unmarshal response. Here's why: %v\n", err)
    }
}
return movie, err
}

// UpdateMovie runs a PartiQL UPDATE statement to update the rating of a movie that
// already exists in the DynamoDB table.
func (runner PartiQLRunner) UpdateMovie(ctx context.Context, movie Movie, rating
float64) error {
    params, err := attributevalue.MarshalList([]interface{}{rating, movie.Title,
movie.Year})
    if err != nil {
        panic(err)
    }
    _, err = runner.DynamoDbClient.ExecuteStatement(ctx,
&dynamodb.ExecuteStatementInput{
    Statement: aws.String(
        fmt.Sprintf("UPDATE \"%v\" SET info.rating=? WHERE title=? AND year=?",
            runner.TableName)),
    Parameters: params,
})
    if err != nil {
        log.Printf("Couldn't update movie %v. Here's why: %v\n", movie.Title, err)
    }
    return err
}

// DeleteMovie runs a PartiQL DELETE statement to remove a movie from the DynamoDB
// table.
func (runner PartiQLRunner) DeleteMovie(ctx context.Context, movie Movie) error {
    params, err := attributevalue.MarshalList([]interface{}{movie.Title, movie.Year})
    if err != nil {
        panic(err)
    }
    _, err = runner.DynamoDbClient.ExecuteStatement(ctx,
&dynamodb.ExecuteStatementInput{
    Statement: aws.String(

```



```
    fmt.Sprintf("DELETE FROM \"%v\" WHERE title=? AND year=?",
        runner.TableName)),
    Parameters: params,
}))
if err != nil {
    log.Printf("Couldn't delete %v from the table. Here's why: %v\n", movie.Title,
        err)
}
return err
}
```

- Para obtener más información sobre la API, consulta [ExecuteStatement](#) la Referencia AWS SDK para Go de la API.

Ejemplos de tecnología sin servidor

Invocación de una función de Lambda desde un desencadenador de DynamoDB

El siguientes ejemplo de código muestra cómo implementar una función de Lambda que recibe un evento desencadenado al recibir registros de una secuencia de DynamoDB. La función recupera la carga útil de DynamoDB y registra el contenido del registro.

SDK para Go V2

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Consumo de un evento de DynamoDB con Lambda mediante Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
    "context"
```

```

"github.com/aws/aws-lambda-go/lambda"
"github.com/aws/aws-lambda-go/events"
"fmt"
)

func HandleRequest(ctx context.Context, event events.DynamoDBEvent) (*string, error)
{
    if len(event.Records) == 0 {
        return nil, fmt.Errorf("received empty event")
    }

    for _, record := range event.Records {
        LogDynamoDBRecord(record)
    }

    message := fmt.Sprintf("Records processed: %d", len(event.Records))
    return &message, nil
}

func main() {
    lambda.Start(HandleRequest)
}

func LogDynamoDBRecord(record events.DynamoDBEventRecord){
    fmt.Println(record.EventID)
    fmt.Println(record.EventName)
    fmt.Printf("%+v\n", record.Change)
}

```

Notificación de los errores de los elementos del lote de las funciones de Lambda con un desencadenador de DynamoDB

El siguiente ejemplo de código muestra cómo implementar una respuesta parcial por lotes para las funciones de Lambda que reciben eventos de una secuencia de DynamoDB. La función informa los errores de los elementos del lote en la respuesta y le indica a Lambda que vuelva a intentar esos mensajes más adelante.

SDK para Go V2

 Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Notificación de los errores de los elementos del lote de DynamoDB con Lambda mediante Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
    "context"
    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
)

type BatchItemFailure struct {
    ItemIdentifier string `json:"ItemIdentifier"`
}

type BatchResult struct {
    BatchItemFailures []BatchItemFailure `json:"BatchItemFailures"`
}

func HandleRequest(ctx context.Context, event events.DynamoDBEvent) (*BatchResult,
    error) {
    var batchItemFailures []BatchItemFailure
    curRecordSequenceNumber := ""

    for _, record := range event.Records {
        // Process your record
        curRecordSequenceNumber = record.Change.SequenceNumber
    }

    if curRecordSequenceNumber != "" {
        batchItemFailures = append(batchItemFailures, BatchItemFailure{ItemIdentifier:
            curRecordSequenceNumber})
    }
}
```

```
batchResult := BatchResult{
    BatchItemFailures: batchItemFailures,
}

return &batchResult, nil
}

func main() {
    lambda.Start(HandleRequest)
}
```

AWS contribuciones de la comunidad

Cómo crear y probar una aplicación sin servidor

El siguiente ejemplo de código muestra cómo crear y probar una aplicación sin servidor mediante API Gateway con Lambda y DynamoDB.

SDK para Go V2

Muestra cómo crear y probar una aplicación sin servidor que consta de una API Gateway con Lambda y DynamoDB mediante el SDK de Go.

Para obtener el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda

Ejemplos de IAM usando SDK para Go V2

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso de la AWS SDK para Go V2 con 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 Go V2

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](#).

```
package main

import (
    "context"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/iam"
)

// main uses the AWS SDK for Go (v2) to create an AWS Identity and Access Management
// (IAM)
```

```
// client and list up to 10 policies in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
        fmt.Println(err)
        return
    }
    iamClient := iam.NewFromConfig(sdkConfig)
    const maxPols = 10
    fmt.Printf("Let's list up to %v policies for your account.\n", maxPols)
    result, err := iamClient.ListPolicies(ctx, &iam.ListPoliciesInput{
        MaxItems: aws.Int32(maxPols),
    })
    if err != nil {
        fmt.Printf("Couldn't list policies for your account. Here's why: %v\n", err)
        return
    }
    if len(result.Policies) == 0 {
        fmt.Println("You don't have any policies!")
    } else {
        for _, policy := range result.Policies {
            fmt.Printf("\t\t%v\n", *policy.PolicyName)
        }
    }
}
```

- Para obtener más información sobre la API, consulta [ListPolicies](#) la Referencia AWS SDK para Go 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 Go V2

i 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](#).

Ejecute un escenario interactivo en un símbolo del sistema.

```
import (  
    "context"  
    "errors"  
    "fmt"  
    "log"  
    "math/rand"  
    "strings"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/config"  
    "github.com/aws/aws-sdk-go-v2/credentials"  
    "github.com/aws/aws-sdk-go-v2/service/iam"  
    "github.com/aws/aws-sdk-go-v2/service/iam/types"  
    "github.com/aws/aws-sdk-go-v2/service/s3"
```

```

"github.com/aws/aws-sdk-go-v2/service/sts"
"github.com/aws/smithy-go"
"github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
"github.com/awsdocs/aws-doc-sdk-examples/gov2/iam/actions"
)

// AssumeRoleScenario shows you how to use the AWS Identity and Access Management
// (IAM)
// service to perform the following actions:
//
// 1. Create a user who has no permissions.
// 2. Create a role that grants permission to list Amazon Simple Storage Service
//    (Amazon S3) buckets for the account.
// 3. Add a policy to let the user assume the role.
// 4. Try and fail to list buckets without permissions.
// 5. Assume the role and list S3 buckets using temporary credentials.
// 6. Delete the policy, role, and user.
type AssumeRoleScenario struct {
    sdkConfig      aws.Config
    accountWrapper actions.AccountWrapper
    policyWrapper  actions.PolicyWrapper
    roleWrapper    actions.RoleWrapper
    userWrapper    actions.UserWrapper
    questioner     demotools.IQuestioner
    helper         IScenarioHelper
    isTestRun      bool
}

// NewAssumeRoleScenario constructs an AssumeRoleScenario instance from a
// configuration.
// It uses the specified config to get an IAM client and create wrappers for the
// actions
// used in the scenario.
func NewAssumeRoleScenario(sdkConfig aws.Config, questioner demotools.IQuestioner,
    helper IScenarioHelper) AssumeRoleScenario {
    iamClient := iam.NewFromConfig(sdkConfig)
    return AssumeRoleScenario{
        sdkConfig:      sdkConfig,
        accountWrapper: actions.AccountWrapper{IamClient: iamClient},
        policyWrapper:  actions.PolicyWrapper{IamClient: iamClient},
        roleWrapper:    actions.RoleWrapper{IamClient: iamClient},
        userWrapper:    actions.UserWrapper{IamClient: iamClient},
        questioner:     questioner,
        helper:         helper,
    }
}

```



```

    }
}

// addTestOptions appends the API options specified in the original configuration to
// another configuration. This is used to attach the middleware stubber to clients
// that are constructed during the scenario, which is needed for unit testing.
func (scenario AssumeRoleScenario) addTestOptions(scenarioConfig *aws.Config) {
    if scenario.isTestRun {
        scenarioConfig.APIOptions = append(scenarioConfig.APIOptions,
            scenario.sdkConfig.APIOptions...)
    }
}

// Run runs the interactive scenario.
func (scenario AssumeRoleScenario) Run(ctx context.Context) {
    defer func() {
        if r := recover(); r != nil {
            log.Printf("Something went wrong with the demo.\n")
            log.Println(r)
        }
    }()

    log.Println(strings.Repeat("-", 88))
    log.Println("Welcome to the AWS Identity and Access Management (IAM) assume role
demo.")
    log.Println(strings.Repeat("-", 88))

    user := scenario.CreateUser(ctx)
    accessKey := scenario.CreateAccessKey(ctx, user)
    role := scenario.CreateRoleAndPolicies(ctx, user)
    noPermsConfig := scenario.ListBucketsWithoutPermissions(ctx, accessKey)
    scenario.ListBucketsWithAssumedRole(ctx, noPermsConfig, role)
    scenario.Cleanup(ctx, user, role)

    log.Println(strings.Repeat("-", 88))
    log.Println("Thanks for watching!")
    log.Println(strings.Repeat("-", 88))
}

// CreateUser creates a new IAM user. This user has no permissions.
func (scenario AssumeRoleScenario) CreateUser(ctx context.Context) *types.User {
    log.Println("Let's create an example user with no permissions.")
    userName := scenario.questioner.Ask("Enter a name for the example user:",
        demotools.NotEmpty{})
}

```

```

user, err := scenario.userWrapper.GetUser(ctx, userName)
if err != nil {
    panic(err)
}
if user == nil {
    user, err = scenario.userWrapper.CreateUser(ctx, userName)
    if err != nil {
        panic(err)
    }
    log.Printf("Created user %v.\n", *user.UserName)
} else {
    log.Printf("User %v already exists.\n", *user.UserName)
}
log.Println(strings.Repeat("-", 88))
return user
}

// CreateAccessKey creates an access key for the user.
func (scenario AssumeRoleScenario) CreateAccessKey(ctx context.Context, user
    *types.User) *types.AccessKey {
    accessKey, err := scenario.userWrapper.CreateAccessKeyPair(ctx, *user.UserName)
    if err != nil {
        panic(err)
    }
    log.Printf("Created access key %v for your user.", *accessKey.AccessKeyId)
    log.Println("Waiting a few seconds for your user to be ready...")
    scenario.helper.Pause(10)
    log.Println(strings.Repeat("-", 88))
    return accessKey
}

// CreateRoleAndPolicies creates a policy that grants permission to list S3 buckets
// for
// the current account and attaches the policy to a newly created role. It also adds
// an
// inline policy to the specified user that grants the user permission to assume the
// role.
func (scenario AssumeRoleScenario) CreateRoleAndPolicies(ctx context.Context, user
    *types.User) *types.Role {
    log.Println("Let's create a role and policy that grant permission to list S3
    buckets.")
    scenario.questioner.Ask("Press Enter when you're ready.")
    listBucketsRole, err := scenario.roleWrapper.CreateRole(ctx,
    scenario.helper.GetName(), *user.Arn)

```

```

if err != nil {
    panic(err)
}
log.Printf("Created role %v.\n", *listBucketsRole.RoleName)
listBucketsPolicy, err := scenario.policyWrapper.CreatePolicy(
    ctx, scenario.helper.GetName(), []string{"s3:ListAllMyBuckets"}, "arn:aws:s3:::*")
if err != nil {
    panic(err)
}
log.Printf("Created policy %v.\n", *listBucketsPolicy.PolicyName)
err = scenario.roleWrapper.AttachRolePolicy(ctx, *listBucketsPolicy.Arn,
    *listBucketsRole.RoleName)
if err != nil {
    panic(err)
}
log.Printf("Attached policy %v to role %v.\n", *listBucketsPolicy.PolicyName,
    *listBucketsRole.RoleName)
err = scenario.userWrapper.CreateUserPolicy(ctx, *user.UserName,
    scenario.helper.GetName(),
    []string{"sts:AssumeRole"}, *listBucketsRole.Arn)
if err != nil {
    panic(err)
}
log.Printf("Created an inline policy for user %v that lets the user assume the
role.\n",
    *user.UserName)
log.Println("Let's give AWS a few seconds to propagate these new resources and
connections...")
scenario.helper.Pause(10)
log.Println(strings.Repeat("-", 88))
return listBucketsRole
}

// ListBucketsWithoutPermissions creates an Amazon S3 client from the user's access
key
// credentials and tries to list buckets for the account. Because the user does not
have
// permission to perform this action, the action fails.
func (scenario AssumeRoleScenario) ListBucketsWithoutPermissions(ctx
context.Context, accessKey *types.AccessKey) *aws.Config {
    log.Println("Let's try to list buckets without permissions. This should return an
AccessDenied error.")
    scenario.questioner.Ask("Press Enter when you're ready.")
    noPermsConfig, err := config.LoadDefaultConfig(ctx,

```

```

    config.WithCredentialsProvider(credentials.NewStaticCredentialsProvider(
        *accessKey.AccessKeyId, *accessKey.SecretAccessKey, ""),
    ))
    if err != nil {
        panic(err)
    }

    // Add test options if this is a test run. This is needed only for testing
    purposes.
    scenario.addTestOptions(&noPermsConfig)

    s3Client := s3.NewFromConfig(noPermsConfig)
    _, err = s3Client.ListBuckets(ctx, &s3.ListBucketsInput{})
    if err != nil {
        // The SDK for Go does not model the AccessDenied error, so check ErrorCode
        directly.
        var ae smithy.APIError
        if errors.As(err, &ae) {
            switch ae.ErrorCode() {
            case "AccessDenied":
                log.Println("Got AccessDenied error, which is the expected result because\n" +
                    "the ListBuckets call was made without permissions.")
            default:
                log.Println("Expected AccessDenied, got something else.")
                panic(err)
            }
        }
    } else {
        log.Println("Expected AccessDenied error when calling ListBuckets without
        permissions,\n" +
            "but the call succeeded. Continuing the example anyway...")
    }
    log.Println(strings.Repeat("-", 88))
    return &noPermsConfig
}

// ListBucketsWithAssumedRole performs the following actions:
//
// 1. Creates an AWS Security Token Service (AWS STS) client from the config
//    created from
//    the user's access key credentials.
// 2. Gets temporary credentials by assuming the role that grants permission to
//    list the
//    buckets.

```

```
// 3. Creates an Amazon S3 client from the temporary credentials.
// 4. Lists buckets for the account. Because the temporary credentials are
//    generated by
//    assuming the role that grants permission, the action succeeds.
func (scenario AssumeRoleScenario) ListBucketsWithAssumedRole(ctx context.Context,
    noPermsConfig *aws.Config, role *types.Role) {
    log.Println("Let's assume the role that grants permission to list buckets and try
    again.")
    scenario.questioner.Ask("Press Enter when you're ready.")
    stsClient := sts.NewFromConfig(*noPermsConfig)
    tempCredentials, err := stsClient.AssumeRole(ctx, &sts.AssumeRoleInput{
        RoleArn:          role.Arn,
        RoleSessionName:  aws.String("AssumeRoleExampleSession"),
        DurationSeconds:   aws.Int32(900),
    })
    if err != nil {
        log.Printf("Couldn't assume role %v.\n", *role.RoleName)
        panic(err)
    }
    log.Printf("Assumed role %v, got temporary credentials.\n", *role.RoleName)
    assumeRoleConfig, err := config.LoadDefaultConfig(ctx,
        config.WithCredentialsProvider(credentials.NewStaticCredentialsProvider(
            *tempCredentials.Credentials.AccessKeyId,
            *tempCredentials.Credentials.SecretAccessKey,
            *tempCredentials.Credentials.SessionToken),
        ),
    )
    if err != nil {
        panic(err)
    }

    // Add test options if this is a test run. This is needed only for testing
    // purposes.
    scenario.addTestOptions(&assumeRoleConfig)

    s3Client := s3.NewFromConfig(assumeRoleConfig)
    result, err := s3Client.ListBuckets(ctx, &s3.ListBucketsInput{})
    if err != nil {
        log.Println("Couldn't list buckets with assumed role credentials.")
        panic(err)
    }
    log.Println("Successfully called ListBuckets with assumed role credentials, \n" +
        "here are some of them:")
    for i := 0; i < len(result.Buckets) && i < 5; i++ {
```

```

    log.Printf("\t%v\n", *result.Buckets[i].Name)
}
log.Println(strings.Repeat("-", 88))
}

// Cleanup deletes all resources created for the scenario.
func (scenario AssumeRoleScenario) Cleanup(ctx context.Context, user *types.User,
role *types.Role) {
    if scenario.questioner.AskBool(
        "Do you want to delete the resources created for this example? (y/n)", "y",
    ) {
        policies, err := scenario.roleWrapper.ListAttachedRolePolicies(ctx,
            *role.RoleName)
        if err != nil {
            panic(err)
        }
        for _, policy := range policies {
            err = scenario.roleWrapper.DetachRolePolicy(ctx, *role.RoleName,
                *policy.PolicyArn)
            if err != nil {
                panic(err)
            }
            err = scenario.policyWrapper.DeletePolicy(ctx, *policy.PolicyArn)
            if err != nil {
                panic(err)
            }
            log.Printf("Detached policy %v from role %v and deleted the policy.\n",
                *policy.PolicyName, *role.RoleName)
        }
        err = scenario.roleWrapper.DeleteRole(ctx, *role.RoleName)
        if err != nil {
            panic(err)
        }
        log.Printf("Deleted role %v.\n", *role.RoleName)

        userPols, err := scenario.userWrapper.ListUserPolicies(ctx, *user.UserName)
        if err != nil {
            panic(err)
        }
        for _, userPol := range userPols {
            err = scenario.userWrapper.DeleteUserPolicy(ctx, *user.UserName, userPol)
            if err != nil {
                panic(err)
            }
        }
    }
}

```

```

    log.Printf("Deleted policy %v from user %v.\n", userPol, *user.UserName)
}
keys, err := scenario.userWrapper.ListAccessKeys(ctx, *user.UserName)
if err != nil {
    panic(err)
}
for _, key := range keys {
    err = scenario.userWrapper.DeleteAccessKey(ctx, *user.UserName, *key.AccessKeyId)
    if err != nil {
        panic(err)
    }
    log.Printf("Deleted access key %v from user %v.\n", *key.AccessKeyId,
*user.UserName)
}
err = scenario.userWrapper.DeleteUser(ctx, *user.UserName)
if err != nil {
    panic(err)
}
log.Printf("Deleted user %v.\n", *user.UserName)
log.Println(strings.Repeat("-", 88))
}

}

// IScenarioHelper abstracts input and wait functions from a scenario so that they
// can be mocked for unit testing.
type IScenarioHelper interface {
    GetName() string
    Pause(secs int)
}

const rMax = 100000

type ScenarioHelper struct {
    Prefix string
    Random *rand.Rand
}

// GetName returns a unique name formed of a prefix and a random number.
func (helper *ScenarioHelper) GetName() string {
    return fmt.Sprintf("%v%v", helper.Prefix, helper.Random.Intn(rMax))
}

// Pause waits for the specified number of seconds.

```

```
func (helper ScenarioHelper) Pause(secs int) {  
    time.Sleep(time.Duration(secs) * time.Second)  
}
```

Defina una estructura que incluya las acciones de la cuenta.

```
import (  
    "context"  
    "log"  
  
    "github.com/aws/aws-sdk-go-v2/service/iam"  
    "github.com/aws/aws-sdk-go-v2/service/iam/types"  
)  
  
// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account  
// actions  
// used in the examples.  
// It contains an IAM service client that is used to perform account actions.  
type AccountWrapper struct {  
    iamClient *iam.Client  
}  
  
// GetAccountPasswordPolicy gets the account password policy for the current  
// account.  
// If no policy has been set, a NoSuchEntityException is error is returned.  
func (wrapper AccountWrapper) GetAccountPasswordPolicy(ctx context.Context)  
    (*types.PasswordPolicy, error) {  
    var pwPolicy *types.PasswordPolicy  
    result, err := wrapper.IamClient.GetAccountPasswordPolicy(ctx,  
        &iam.GetAccountPasswordPolicyInput{})  
    if err != nil {  
        log.Printf("Couldn't get account password policy. Here's why: %v\n", err)  
    } else {  
        pwPolicy = result.PasswordPolicy  
    }  
    return pwPolicy, err  
}
```



```
// ListSAMLProviders gets the SAML providers for the account.
func (wrapper AccountWrapper) ListSAMLProviders(ctx context.Context)
([]types.SAMLProviderListEntry, error) {
    var providers []types.SAMLProviderListEntry
    result, err := wrapper.IamClient.ListSAMLProviders(ctx,
        &iam.ListSAMLProvidersInput{})
    if err != nil {
        log.Printf("Couldn't list SAML providers. Here's why: %v\n", err)
    } else {
        providers = result.SAMLProviderList
    }
    return providers, err
}
```

Defina una estructura que incluya las acciones de la política.

```
import (
    "context"
    "encoding/json"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/iam"
    "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
// actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
    IamClient *iam.Client
}

// ListPolicies gets up to maxPolicies policies.
```

```

func (wrapper PolicyWrapper) ListPolicies(ctx context.Context, maxPolicies int32)
([]types.Policy, error) {
    var policies []types.Policy
    result, err := wrapper.IamClient.ListPolicies(ctx, &iam.ListPoliciesInput{
        MaxItems: aws.Int32(maxPolicies),
    })
    if err != nil {
        log.Printf("Couldn't list policies. Here's why: %v\n", err)
    } else {
        policies = result.Policies
    }
    return policies, err
}

// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
    Version    string
    Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
    Effect    string
    Action    []string
    Principal map[string]string `json:",omitempty"`
    Resource  *string             `json:",omitempty"`
}

// CreatePolicy creates a policy that grants a list of actions to the specified
// resource.
// PolicyDocument shows how to work with a policy document as a data structure and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper PolicyWrapper) CreatePolicy(ctx context.Context, policyName string,
    actions []string,
    resourceArn string) (*types.Policy, error) {
    var policy *types.Policy
    policyDoc := PolicyDocument{
        Version: "2012-10-17",
        Statement: []PolicyStatement{{
            Effect:    "Allow",
            Action:    actions,

```

```

    Resource: aws.String(resourceArn),
  }},
}
policyBytes, err := json.Marshal(policyDoc)
if err != nil {
    log.Printf("Couldn't create policy document for %v. Here's why: %v\n",
resourceArn, err)
    return nil, err
}
result, err := wrapper.IamClient.CreatePolicy(ctx, &iam.CreatePolicyInput{
    PolicyDocument: aws.String(string(policyBytes)),
    PolicyName:      aws.String(policyName),
})
if err != nil {
    log.Printf("Couldn't create policy %v. Here's why: %v\n", policyName, err)
} else {
    policy = result.Policy
}
return policy, err
}

// GetPolicy gets data about a policy.
func (wrapper PolicyWrapper) GetPolicy(ctx context.Context, policyArn string)
(*types.Policy, error) {
    var policy *types.Policy
    result, err := wrapper.IamClient.GetPolicy(ctx, &iam.GetPolicyInput{
        PolicyArn: aws.String(policyArn),
    })
    if err != nil {
        log.Printf("Couldn't get policy %v. Here's why: %v\n", policyArn, err)
    } else {
        policy = result.Policy
    }
    return policy, err
}

// DeletePolicy deletes a policy.
func (wrapper PolicyWrapper) DeletePolicy(ctx context.Context, policyArn string)
error {
    _, err := wrapper.IamClient.DeletePolicy(ctx, &iam.DeletePolicyInput{

```

```

    PolicyArn: aws.String(policyArn),
  })
  if err != nil {
    log.Printf("Couldn't delete policy %v. Here's why: %v\n", policyArn, err)
  }
  return err
}

```

Defina una estructura que incluya las acciones de rol.

```

import (
    "context"
    "encoding/json"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/iam"
    "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
    iamClient *iam.Client
}

// ListRoles gets up to maxRoles roles.
func (wrapper RoleWrapper) ListRoles(ctx context.Context, maxRoles int32)
    ([]types.Role, error) {
    var roles []types.Role
    result, err := wrapper.IamClient.ListRoles(ctx,
        &iam.ListRolesInput{MaxItems: aws.Int32(maxRoles)},
    )
    if err != nil {
        log.Printf("Couldn't list roles. Here's why: %v\n", err)
    } else {
        roles = result.Roles
    }
}

```

```

    }
    return roles, err
}

// CreateRole creates a role that trusts a specified user. The trusted user can
// assume
// the role to acquire its permissions.
// PolicyDocument shows how to work with a policy document as a data structure and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper RoleWrapper) CreateRole(ctx context.Context, roleName string,
    trustedUserArn string) (*types.Role, error) {
    var role *types.Role
    trustPolicy := PolicyDocument{
        Version: "2012-10-17",
        Statement: []PolicyStatement{{
            Effect:    "Allow",
            Principal: map[string]string{"AWS": trustedUserArn},
            Action:    []string{"sts:AssumeRole"},
        }},
    }
    policyBytes, err := json.Marshal(trustPolicy)
    if err != nil {
        log.Printf("Couldn't create trust policy for %v. Here's why: %v\n",
            trustedUserArn, err)
        return nil, err
    }
    result, err := wrapper.IamClient.CreateRole(ctx, &iam.CreateRoleInput{
        AssumeRolePolicyDocument: aws.String(string(policyBytes)),
        RoleName:                  aws.String(roleName),
    })
    if err != nil {
        log.Printf("Couldn't create role %v. Here's why: %v\n", roleName, err)
    } else {
        role = result.Role
    }
    return role, err
}

// GetRole gets data about a role.

```

```

func (wrapper RoleWrapper) GetRole(ctx context.Context, roleName string)
(*types.Role, error) {
    var role *types.Role
    result, err := wrapper.IamClient.GetRole(ctx,
        &iam.GetRoleInput{RoleName: aws.String(roleName)})
    if err != nil {
        log.Printf("Couldn't get role %v. Here's why: %v\n", roleName, err)
    } else {
        role = result.Role
    }
    return role, err
}

// CreateServiceLinkedRole creates a service-linked role that is owned by the
// specified service.
func (wrapper RoleWrapper) CreateServiceLinkedRole(ctx context.Context, serviceName
string, description string) (
    *types.Role, error) {
    var role *types.Role
    result, err := wrapper.IamClient.CreateServiceLinkedRole(ctx,
        &iam.CreateServiceLinkedRoleInput{
            AWSServiceName: aws.String(serviceName),
            Description:    aws.String(description),
        })
    if err != nil {
        log.Printf("Couldn't create service-linked role %v. Here's why: %v\n",
            serviceName, err)
    } else {
        role = result.Role
    }
    return role, err
}

// DeleteServiceLinkedRole deletes a service-linked role.
func (wrapper RoleWrapper) DeleteServiceLinkedRole(ctx context.Context, roleName
string) error {
    _, err := wrapper.IamClient.DeleteServiceLinkedRole(ctx,
        &iam.DeleteServiceLinkedRoleInput{
            RoleName: aws.String(roleName)},
    )
}

```

```
if err != nil {
    log.Printf("Couldn't delete service-linked role %v. Here's why: %v\n", roleName,
err)
}
return err
}

// AttachRolePolicy attaches a policy to a role.
func (wrapper RoleWrapper) AttachRolePolicy(ctx context.Context, policyArn string,
roleName string) error {
    _, err := wrapper.IamClient.AttachRolePolicy(ctx, &iam.AttachRolePolicyInput{
        PolicyArn: aws.String(policyArn),
        RoleName:  aws.String(roleName),
    })
    if err != nil {
        log.Printf("Couldn't attach policy %v to role %v. Here's why: %v\n", policyArn,
roleName, err)
    }
    return err
}

// ListAttachedRolePolicies lists the policies that are attached to the specified
role.
func (wrapper RoleWrapper) ListAttachedRolePolicies(ctx context.Context, roleName
string) ([]types.AttachedPolicy, error) {
    var policies []types.AttachedPolicy
    result, err := wrapper.IamClient.ListAttachedRolePolicies(ctx,
&iam.ListAttachedRolePoliciesInput{
        RoleName: aws.String(roleName),
    })
    if err != nil {
        log.Printf("Couldn't list attached policies for role %v. Here's why: %v\n",
roleName, err)
    } else {
        policies = result.AttachedPolicies
    }
    return policies, err
}
```

```
// DetachRolePolicy detaches a policy from a role.
func (wrapper RoleWrapper) DetachRolePolicy(ctx context.Context, roleName string,
policyArn string) error {
    _, err := wrapper.IamClient.DetachRolePolicy(ctx, &iam.DetachRolePolicyInput{
        PolicyArn: aws.String(policyArn),
        RoleName:  aws.String(roleName),
    })
    if err != nil {
        log.Printf("Couldn't detach policy from role %v. Here's why: %v\n", roleName, err)
    }
    return err
}

// ListRolePolicies lists the inline policies for a role.
func (wrapper RoleWrapper) ListRolePolicies(ctx context.Context, roleName string)
([]string, error) {
    var policies []string
    result, err := wrapper.IamClient.ListRolePolicies(ctx, &iam.ListRolePoliciesInput{
        RoleName: aws.String(roleName),
    })
    if err != nil {
        log.Printf("Couldn't list policies for role %v. Here's why: %v\n", roleName, err)
    } else {
        policies = result.PolicyNames
    }
    return policies, err
}

// DeleteRole deletes a role. All attached policies must be detached before a
// role can be deleted.
func (wrapper RoleWrapper) DeleteRole(ctx context.Context, roleName string) error {
    _, err := wrapper.IamClient.DeleteRole(ctx, &iam.DeleteRoleInput{
        RoleName: aws.String(roleName),
    })
    if err != nil {
        log.Printf("Couldn't delete role %v. Here's why: %v\n", roleName, err)
    }
    return err
}
```


Defina una estructura que incluya las acciones del usuario.

```
import (
    "context"
    "encoding/json"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/iam"
    "github.com/aws/aws-sdk-go-v2/service/iam/types"
    "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
    iamClient *iam.Client
}

// ListUsers gets up to maxUsers number of users.
func (wrapper UserWrapper) ListUsers(ctx context.Context, maxUsers int32)
    ([]types.User, error) {
    var users []types.User
    result, err := wrapper.iamClient.ListUsers(ctx, &iam.ListUsersInput{
        MaxItems: aws.Int32(maxUsers),
    })
    if err != nil {
        log.Printf("Couldn't list users. Here's why: %v\n", err)
    } else {
        users = result.Users
    }
    return users, err
}
```

```
// GetUser gets data about a user.
func (wrapper UserWrapper) GetUser(ctx context.Context, userName string)
(*types.User, error) {
    var user *types.User
    result, err := wrapper.IamClient.GetUser(ctx, &iam.GetUserInput{
        UserName: aws.String(userName),
    })
    if err != nil {
        var apiError smithy.APIError
        if errors.As(err, &apiError) {
            switch apiError.(type) {
            case *types.NoSuchEntityException:
                log.Printf("User %v does not exist.\n", userName)
                err = nil
            default:
                log.Printf("Couldn't get user %v. Here's why: %v\n", userName, err)
            }
        }
    } else {
        user = result.User
    }
    return user, err
}

// CreateUser creates a new user with the specified name.
func (wrapper UserWrapper) CreateUser(ctx context.Context, userName string)
(*types.User, error) {
    var user *types.User
    result, err := wrapper.IamClient.CreateUser(ctx, &iam.CreateUserInput{
        UserName: aws.String(userName),
    })
    if err != nil {
        log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
    } else {
        user = result.User
    }
    return user, err
}
```

```
// CreateUserPolicy adds an inline policy to a user. This example creates a policy
// that
// grants a list of actions on a specified role.
// PolicyDocument shows how to work with a policy document as a data structure and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper UserWrapper) CreateUserPolicy(ctx context.Context, userName string,
policyName string, actions []string,
roleArn string) error {
policyDoc := PolicyDocument{
Version: "2012-10-17",
Statement: []PolicyStatement{{
Effect: "Allow",
Action: actions,
Resource: aws.String(roleArn),
}},
}
policyBytes, err := json.Marshal(policyDoc)
if err != nil {
log.Printf("Couldn't create policy document for %v. Here's why: %v\n", roleArn,
err)
return err
}
_, err = wrapper.IamClient.PutUserPolicy(ctx, &iam.PutUserPolicyInput{
PolicyDocument: aws.String(string(policyBytes)),
PolicyName: aws.String(policyName),
UserName: aws.String(userName),
})
if err != nil {
log.Printf("Couldn't create policy for user %v. Here's why: %v\n", userName, err)
}
return err
}

// ListUserPolicies lists the inline policies for the specified user.
func (wrapper UserWrapper) ListUserPolicies(ctx context.Context, userName string)
([]string, error) {
var policies []string
result, err := wrapper.IamClient.ListUserPolicies(ctx, &iam.ListUserPoliciesInput{
UserName: aws.String(userName),
})
if err != nil {
log.Printf("Couldn't list policies for user %v. Here's why: %v\n", userName, err)
}
```

```
} else {
    policies = result.PolicyNames
}
return policies, err
}

// DeleteUserPolicy deletes an inline policy from a user.
func (wrapper UserWrapper) DeleteUserPolicy(ctx context.Context, userName string,
    policyName string) error {
    _, err := wrapper.IamClient.DeleteUserPolicy(ctx, &iam.DeleteUserPolicyInput{
        PolicyName: aws.String(policyName),
        UserName:   aws.String(userName),
    })
    if err != nil {
        log.Printf("Couldn't delete policy from user %v. Here's why: %v\n", userName, err)
    }
    return err
}

// DeleteUser deletes a user.
func (wrapper UserWrapper) DeleteUser(ctx context.Context, userName string) error {
    _, err := wrapper.IamClient.DeleteUser(ctx, &iam.DeleteUserInput{
        UserName: aws.String(userName),
    })
    if err != nil {
        log.Printf("Couldn't delete user %v. Here's why: %v\n", userName, err)
    }
    return err
}

// CreateAccessKeyPair creates an access key for a user. The returned access key
// contains
// the ID and secret credentials needed to use the key.
func (wrapper UserWrapper) CreateAccessKeyPair(ctx context.Context, userName string)
    (*types.AccessKey, error) {
    var key *types.AccessKey
    result, err := wrapper.IamClient.CreateAccessKey(ctx, &iam.CreateAccessKeyInput{
        UserName: aws.String(userName)})
}
```

```
if err != nil {
    log.Printf("Couldn't create access key pair for user %v. Here's why: %v\n",
        userName, err)
} else {
    key = result.AccessKey
}
return key, err
}

// DeleteAccessKey deletes an access key from a user.
func (wrapper UserWrapper) DeleteAccessKey(ctx context.Context, userName string,
    keyId string) error {
    _, err := wrapper.IamClient.DeleteAccessKey(ctx, &iam.DeleteAccessKeyInput{
        AccessKeyId: aws.String(keyId),
        UserName:    aws.String(userName),
    })
    if err != nil {
        log.Printf("Couldn't delete access key %v. Here's why: %v\n", keyId, err)
    }
    return err
}

// ListAccessKeys lists the access keys for the specified user.
func (wrapper UserWrapper) ListAccessKeys(ctx context.Context, userName string)
    ([]types.AccessKeyMetadata, error) {
    var keys []types.AccessKeyMetadata
    result, err := wrapper.IamClient.ListAccessKeys(ctx, &iam.ListAccessKeysInput{
        UserName: aws.String(userName),
    })
    if err != nil {
        log.Printf("Couldn't list access keys for user %v. Here's why: %v\n", userName,
            err)
    } else {
        keys = result.AccessKeyMetadata
    }
    return keys, err
}
```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK para Go .
 - [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 Go V2

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
import (  
    "context"  
    "encoding/json"  
    "log"  
  
    "github.com/aws/aws-sdk-go-v2/aws"
```

```

    "github.com/aws/aws-sdk-go-v2/service/iam"
    "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
    iamClient *iam.Client
}

// AttachRolePolicy attaches a policy to a role.
func (wrapper RoleWrapper) AttachRolePolicy(ctx context.Context, policyArn string,
    roleName string) error {
    _, err := wrapper.IamClient.AttachRolePolicy(ctx, &iam.AttachRolePolicyInput{
        PolicyArn: aws.String(policyArn),
        RoleName:  aws.String(roleName),
    })
    if err != nil {
        log.Printf("Couldn't attach policy %v to role %v. Here's why: %v\n", policyArn,
            roleName, err)
    }
    return err
}

```

- Para obtener más información sobre la API, consulta [AttachRolePolicy](#) la Referencia AWS SDK para Go de la API.

CreateAccessKey

En el siguiente ejemplo de código, se muestra cómo utilizar CreateAccessKey.

SDK para Go V2

 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 (  
    "context"  
    "encoding/json"  
    "errors"  
    "log"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/iam"  
    "github.com/aws/aws-sdk-go-v2/service/iam/types"  
    "github.com/aws/smithy-go"  
)  
  
// UserWrapper encapsulates user actions used in the examples.  
// It contains an IAM service client that is used to perform user actions.  
type UserWrapper struct {  
    iamClient *iam.Client  
}  
  
// CreateAccessKeyPair creates an access key for a user. The returned access key  
// contains  
// the ID and secret credentials needed to use the key.  
func (wrapper UserWrapper) CreateAccessKeyPair(ctx context.Context, userName string)  
(*types.AccessKey, error) {  
    var key *types.AccessKey  
    result, err := wrapper.IamClient.CreateAccessKey(ctx, &iam.CreateAccessKeyInput{  
        UserName: aws.String(userName)})  
    if err != nil {  
        log.Printf("Couldn't create access key pair for user %v. Here's why: %v\n",  
            userName, err)  
    } else {  
        key = result.AccessKey  
    }  
}
```



```
}  
    return key, err  
}
```

- Para obtener más información sobre la API, consulta [CreateAccessKey](#) la Referencia AWS SDK para Go de la API.

CreatePolicy

En el siguiente ejemplo de código, se muestra cómo utilizar CreatePolicy.

SDK para Go V2

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 (  
    "context"  
    "encoding/json"  
    "log"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/iam"  
    "github.com/aws/aws-sdk-go-v2/service/iam/types"  
)  
  
// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy  
// actions  
// used in the examples.  
// It contains an IAM service client that is used to perform policy actions.  
type PolicyWrapper struct {  
    iamClient *iam.Client  
}
```

```
// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
    Version    string
    Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
    Effect    string
    Action    []string
    Principal map[string]string `json:",omitempty"`
    Resource  *string            `json:",omitempty"`
}

// CreatePolicy creates a policy that grants a list of actions to the specified
// resource.
// PolicyDocument shows how to work with a policy document as a data structure and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper PolicyWrapper) CreatePolicy(ctx context.Context, policyName string,
    actions []string,
    resourceArn string) (*types.Policy, error) {
    var policy *types.Policy
    policyDoc := PolicyDocument{
        Version: "2012-10-17",
        Statement: []PolicyStatement{{
            Effect:    "Allow",
            Action:    actions,
            Resource: aws.String(resourceArn),
        }},
    }
    policyBytes, err := json.Marshal(policyDoc)
    if err != nil {
        log.Printf("Couldn't create policy document for %v. Here's why: %v\n",
            resourceArn, err)
        return nil, err
    }
    result, err := wrapper.IamClient.CreatePolicy(ctx, &iam.CreatePolicyInput{
        PolicyDocument: aws.String(string(policyBytes)),
        PolicyName:     aws.String(policyName),
    })
    if err != nil {
        log.Printf("Couldn't create policy %v. Here's why: %v\n", policyName, err)
    }
}
```

```
} else {  
    policy = result.Policy  
}  
return policy, err  
}
```

- Para obtener más información sobre la API, consulta [CreatePolicy](#) la Referencia AWS SDK para Go de la API.

CreateRole

En el siguiente ejemplo de código, se muestra cómo utilizar CreateRole.

SDK para Go V2

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 (  
    "context"  
    "encoding/json"  
    "log"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/iam"  
    "github.com/aws/aws-sdk-go-v2/service/iam/types"  
)  
  
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions  
// used in the examples.  
// It contains an IAM service client that is used to perform role actions.  
type RoleWrapper struct {  
    iamClient *iam.Client  
}
```

```
// CreateRole creates a role that trusts a specified user. The trusted user can
// assume
// the role to acquire its permissions.
// PolicyDocument shows how to work with a policy document as a data structure and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper RoleWrapper) CreateRole(ctx context.Context, roleName string,
    trustedUserArn string) (*types.Role, error) {
    var role *types.Role
    trustPolicy := PolicyDocument{
        Version: "2012-10-17",
        Statement: []PolicyStatement{{
            Effect:    "Allow",
            Principal: map[string]string{"AWS": trustedUserArn},
            Action:    []string{"sts:AssumeRole"},
        }},
    }
    policyBytes, err := json.Marshal(trustPolicy)
    if err != nil {
        log.Printf("Couldn't create trust policy for %v. Here's why: %v\n",
            trustedUserArn, err)
        return nil, err
    }
    result, err := wrapper.IamClient.CreateRole(ctx, &iam.CreateRoleInput{
        AssumeRolePolicyDocument: aws.String(string(policyBytes)),
        RoleName:                  aws.String(roleName),
    })
    if err != nil {
        log.Printf("Couldn't create role %v. Here's why: %v\n", roleName, err)
    } else {
        role = result.Role
    }
    return role, err
}
```

- Para obtener más información sobre la API, consulta [CreateRole](#) la Referencia AWS SDK para Go de la API.

CreateServiceLinkedRole

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateServiceLinkedRole`.

SDK para Go V2

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 (
    "context"
    "encoding/json"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/iam"
    "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
    iamClient *iam.Client
}

// CreateServiceLinkedRole creates a service-linked role that is owned by the
// specified service.
func (wrapper RoleWrapper) CreateServiceLinkedRole(ctx context.Context, serviceName
    string, description string) (
    *types.Role, error) {
    var role *types.Role
    result, err := wrapper.IamClient.CreateServiceLinkedRole(ctx,
    &iam.CreateServiceLinkedRoleInput{
        AWSServiceName: aws.String(serviceName),
        Description:     aws.String(description),
    })
}
```

```
if err != nil {
    log.Printf("Couldn't create service-linked role %v. Here's why: %v\n",
        serviceName, err)
} else {
    role = result.Role
}
return role, err
}
```

- Para obtener más información sobre la API, consulta [CreateServiceLinkedRole](#) la Referencia AWS SDK para Go de la API.

CreateUser

En el siguiente ejemplo de código, se muestra cómo utilizar CreateUser.

SDK para Go V2

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 (
    "context"
    "encoding/json"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/iam"
    "github.com/aws/aws-sdk-go-v2/service/iam/types"
    "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
```

```

type UserWrapper struct {
    iamClient *iam.Client
}

// CreateUser creates a new user with the specified name.
func (wrapper UserWrapper) CreateUser(ctx context.Context, userName string)
(*types.User, error) {
    var user *types.User
    result, err := wrapper.IamClient.CreateUser(ctx, &iam.CreateUserInput{
        UserName: aws.String(userName),
    })
    if err != nil {
        log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
    } else {
        user = result.User
    }
    return user, err
}

```

- Para obtener más información sobre la API, consulta [CreateUser](#) la Referencia AWS SDK para Go de la API.

DeleteAccessKey

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteAccessKey.

SDK para Go V2

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 (
    "context"

```

```

"encoding/json"
"errors"
"log"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/service/iam"
"github.com/aws/aws-sdk-go-v2/service/iam/types"
"github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
    iamClient *iam.Client
}

// DeleteAccessKey deletes an access key from a user.
func (wrapper UserWrapper) DeleteAccessKey(ctx context.Context, userName string,
    keyId string) error {
    _, err := wrapper.IamClient.DeleteAccessKey(ctx, &iam.DeleteAccessKeyInput{
        AccessKeyId: aws.String(keyId),
        UserName:    aws.String(userName),
    })
    if err != nil {
        log.Printf("Couldn't delete access key %v. Here's why: %v\n", keyId, err)
    }
    return err
}

```

- Para obtener más información sobre la API, consulta [DeleteAccessKey](#) la Referencia AWS SDK para Go de la API.

DeletePolicy

En el siguiente ejemplo de código, se muestra cómo utilizar DeletePolicy.

SDK para Go V2

 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 (  
    "context"  
    "encoding/json"  
    "log"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/iam"  
    "github.com/aws/aws-sdk-go-v2/service/iam/types"  
)  
  
// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy  
// actions  
// used in the examples.  
// It contains an IAM service client that is used to perform policy actions.  
type PolicyWrapper struct {  
    IamClient *iam.Client  
}  
  
// DeletePolicy deletes a policy.  
func (wrapper PolicyWrapper) DeletePolicy(ctx context.Context, policyArn string)  
    error {  
    _, err := wrapper.IamClient.DeletePolicy(ctx, &iam.DeletePolicyInput{  
        PolicyArn: aws.String(policyArn),  
    })  
    if err != nil {  
        log.Printf("Couldn't delete policy %v. Here's why: %v\n", policyArn, err)  
    }  
    return err  
}
```

- Para obtener más información sobre la API, consulta [DeletePolicy](#) la Referencia AWS SDK para Go de la API.

DeleteRole

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteRole.

SDK para Go V2

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 (
    "context"
    "encoding/json"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/iam"
    "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
    iamClient *iam.Client
}

// DeleteRole deletes a role. All attached policies must be detached before a
// role can be deleted.
func (wrapper RoleWrapper) DeleteRole(ctx context.Context, roleName string) error {
    _, err := wrapper.IamClient.DeleteRole(ctx, &iam.DeleteRoleInput{
        RoleName: aws.String(roleName),
    })
    if err != nil {
```

```
    log.Printf("Couldn't delete role %v. Here's why: %v\n", roleName, err)
}
return err
}
```

- Para obtener más información sobre la API, consulta [DeleteRole](#) la Referencia AWS SDK para Go de la API.

DeleteServiceLinkedRole

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteServiceLinkedRole.

SDK para Go V2

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 (
    "context"
    "encoding/json"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/iam"
    "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
    iamClient *iam.Client
}
```

```
// DeleteServiceLinkedRole deletes a service-linked role.
func (wrapper RoleWrapper) DeleteServiceLinkedRole(ctx context.Context, roleName
string) error {
    _, err := wrapper.IamClient.DeleteServiceLinkedRole(ctx,
    &iam.DeleteServiceLinkedRoleInput{
        RoleName: aws.String(roleName)},
    )
    if err != nil {
        log.Printf("Couldn't delete service-linked role %v. Here's why: %v\n", roleName,
err)
    }
    return err
}
```

- Para obtener más información sobre la API, consulta [DeleteServiceLinkedRole](#) la Referencia AWS SDK para Go de la API.

DeleteUser

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteUser.

SDK para Go V2

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 (
    "context"
    "encoding/json"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/iam"
```

```

"github.com/aws/aws-sdk-go-v2/service/iam/types"
"github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
    iamClient *iam.Client
}

// DeleteUser deletes a user.
func (wrapper UserWrapper) DeleteUser(ctx context.Context, userName string) error {
    _, err := wrapper.IamClient.DeleteUser(ctx, &iam.DeleteUserInput{
        UserName: aws.String(userName),
    })
    if err != nil {
        log.Printf("Couldn't delete user %v. Here's why: %v\n", userName, err)
    }
    return err
}

```

- Para obtener más información sobre la API, consulta [DeleteUser](#) la Referencia AWS SDK para Go de la API.

DeleteUserPolicy

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteUserPolicy.

SDK para Go V2

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 (
    "context"
    "encoding/json"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/iam"
    "github.com/aws/aws-sdk-go-v2/service/iam/types"
    "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
    iamClient *iam.Client
}

// DeleteUserPolicy deletes an inline policy from a user.
func (wrapper UserWrapper) DeleteUserPolicy(ctx context.Context, userName string,
    policyName string) error {
    _, err := wrapper.IamClient.DeleteUserPolicy(ctx, &iam.DeleteUserPolicyInput{
        PolicyName: aws.String(policyName),
        UserName:   aws.String(userName),
    })
    if err != nil {
        log.Printf("Couldn't delete policy from user %v. Here's why: %v\n", userName, err)
    }
    return err
}

```

- Para obtener más información sobre la API, consulta [DeleteUserPolicy](#) la Referencia AWS SDK para Go de la API.

DetachRolePolicy

En el siguiente ejemplo de código, se muestra cómo utilizar DetachRolePolicy.

SDK para Go V2

 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 (  
    "context"  
    "encoding/json"  
    "log"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/iam"  
    "github.com/aws/aws-sdk-go-v2/service/iam/types"  
)  
  
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions  
// used in the examples.  
// It contains an IAM service client that is used to perform role actions.  
type RoleWrapper struct {  
    iamClient *iam.Client  
}  
  
// DetachRolePolicy detaches a policy from a role.  
func (wrapper RoleWrapper) DetachRolePolicy(ctx context.Context, roleName string,  
    policyArn string) error {  
    _, err := wrapper.iamClient.DetachRolePolicy(ctx, &iam.DetachRolePolicyInput{  
        PolicyArn: aws.String(policyArn),  
        RoleName:  aws.String(roleName),  
    })  
    if err != nil {  
        log.Printf("Couldn't detach policy from role %v. Here's why: %v\n", roleName, err)  
    }  
    return err  
}
```

- Para obtener más información sobre la API, consulta [DetachRolePolicy](#) la Referencia AWS SDK para Go de la API.

GetAccountPasswordPolicy

En el siguiente ejemplo de código, se muestra cómo utilizar `GetAccountPasswordPolicy`.

SDK para Go V2

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 (
    "context"
    "log"

    "github.com/aws/aws-sdk-go-v2/service/iam"
    "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account
// actions
// used in the examples.
// It contains an IAM service client that is used to perform account actions.
type AccountWrapper struct {
    iamClient *iam.Client
}

// GetAccountPasswordPolicy gets the account password policy for the current
// account.
// If no policy has been set, a NoSuchEntityException is error is returned.
func (wrapper AccountWrapper) GetAccountPasswordPolicy(ctx context.Context)
(*types.PasswordPolicy, error) {
    var pwPolicy *types.PasswordPolicy
    result, err := wrapper.IamClient.GetAccountPasswordPolicy(ctx,
        &iam.GetAccountPasswordPolicyInput{})
}
```



```
if err != nil {
    log.Printf("Couldn't get account password policy. Here's why: %v\n", err)
} else {
    pwPolicy = result.PasswordPolicy
}
return pwPolicy, err
}
```

- Para obtener más información sobre la API, consulta [GetAccountPasswordPolicy](#) la Referencia AWS SDK para Go de la API.

GetPolicy

En el siguiente ejemplo de código, se muestra cómo utilizar GetPolicy.

SDK para Go V2

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 (
    "context"
    "encoding/json"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/iam"
    "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
// actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
```

```

    iamClient *iam.Client
}

// GetPolicy gets data about a policy.
func (wrapper PolicyWrapper) GetPolicy(ctx context.Context, policyArn string)
(*types.Policy, error) {
    var policy *types.Policy
    result, err := wrapper.IamClient.GetPolicy(ctx, &iam.GetPolicyInput{
        PolicyArn: aws.String(policyArn),
    })
    if err != nil {
        log.Printf("Couldn't get policy %v. Here's why: %v\n", policyArn, err)
    } else {
        policy = result.Policy
    }
    return policy, err
}

```

- Para obtener más información sobre la API, consulta [GetPolicy](#) la Referencia AWS SDK para Go de la API.

GetRole

En el siguiente ejemplo de código, se muestra cómo utilizar GetRole.

SDK para Go V2

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 (
    "context"
    "encoding/json"

```

```

"log"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/service/iam"
"github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
    iamClient *iam.Client
}

// GetRole gets data about a role.
func (wrapper RoleWrapper) GetRole(ctx context.Context, roleName string)
(*types.Role, error) {
    var role *types.Role
    result, err := wrapper.IamClient.GetRole(ctx,
        &iam.GetRoleInput{RoleName: aws.String(roleName)})
    if err != nil {
        log.Printf("Couldn't get role %v. Here's why: %v\n", roleName, err)
    } else {
        role = result.Role
    }
    return role, err
}

```

- Para obtener más información sobre la API, consulta [GetRole](#) la Referencia AWS SDK para Go de la API.

GetUser

En el siguiente ejemplo de código, se muestra cómo utilizar GetUser.

SDK para Go V2

 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 (  
    "context"  
    "encoding/json"  
    "errors"  
    "log"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/iam"  
    "github.com/aws/aws-sdk-go-v2/service/iam/types"  
    "github.com/aws/smithy-go"  
)  
  
// UserWrapper encapsulates user actions used in the examples.  
// It contains an IAM service client that is used to perform user actions.  
type UserWrapper struct {  
    iamClient *iam.Client  
}  
  
// GetUser gets data about a user.  
func (wrapper UserWrapper) GetUser(ctx context.Context, userName string)  
    (*types.User, error) {  
    var user *types.User  
    result, err := wrapper.iamClient.GetUser(ctx, &iam.GetUserInput{  
        UserName: aws.String(userName),  
    })  
    if err != nil {  
        var apiError smithy.APIError  
        if errors.As(err, &apiError) {  
            switch apiError.(type) {  
            case *types.NoSuchEntityException:  
                log.Printf("User %v does not exist.\n", userName)
```

```
    err = nil
    default:
        log.Printf("Couldn't get user %v. Here's why: %v\n", userName, err)
    }
}
} else {
    user = result.User
}
return user, err
}
```

- Para obtener más información sobre la API, consulta [GetUser](#) la Referencia AWS SDK para Go de la API.

ListAccessKeys

En el siguiente ejemplo de código, se muestra cómo utilizar ListAccessKeys.

SDK para Go V2

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 (
    "context"
    "encoding/json"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/iam"
    "github.com/aws/aws-sdk-go-v2/service/iam/types"
    "github.com/aws/smithy-go"
)
```

```
// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
    iamClient *iam.Client
}

// ListAccessKeys lists the access keys for the specified user.
func (wrapper UserWrapper) ListAccessKeys(ctx context.Context, userName string)
([]types.AccessKeyMetadata, error) {
    var keys []types.AccessKeyMetadata
    result, err := wrapper.IamClient.ListAccessKeys(ctx, &iam.ListAccessKeysInput{
        UserName: aws.String(userName),
    })
    if err != nil {
        log.Printf("Couldn't list access keys for user %v. Here's why: %v\n", userName,
            err)
    } else {
        keys = result.AccessKeyMetadata
    }
    return keys, err
}
```

- Para obtener más información sobre la API, consulta [ListAccessKeys](#) la Referencia AWS SDK para Go de la API.

ListAttachedRolePolicies

En el siguiente ejemplo de código, se muestra cómo utilizar ListAttachedRolePolicies.

SDK para Go V2

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 (
    "context"
    "encoding/json"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/iam"
    "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
    iamClient *iam.Client
}

// ListAttachedRolePolicies lists the policies that are attached to the specified
// role.
func (wrapper RoleWrapper) ListAttachedRolePolicies(ctx context.Context, roleName
string) ([]types.AttachedPolicy, error) {
    var policies []types.AttachedPolicy
    result, err := wrapper.IamClient.ListAttachedRolePolicies(ctx,
    &iam.ListAttachedRolePoliciesInput{
        RoleName: aws.String(roleName),
    })
    if err != nil {
        log.Printf("Couldn't list attached policies for role %v. Here's why: %v\n",
        roleName, err)
    } else {
        policies = result.AttachedPolicies
    }
    return policies, err
}

```

- Para obtener más información sobre la API, consulta [ListAttachedRolePolicies](#) la Referencia AWS SDK para Go de la API.

ListGroups

En el siguiente ejemplo de código, se muestra cómo utilizar ListGroups.

SDK para Go V2

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 (
    "context"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/iam"
    "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// GroupWrapper encapsulates AWS Identity and Access Management (IAM) group actions
// used in the examples.
// It contains an IAM service client that is used to perform group actions.
type GroupWrapper struct {
    iamClient *iam.Client
}

// ListGroups lists up to maxGroups number of groups.
func (wrapper GroupWrapper) ListGroups(ctx context.Context, maxGroups int32)
    ([]types.Group, error) {
    var groups []types.Group
    result, err := wrapper.iamClient.ListGroups(ctx, &iam.ListGroupsInput{
        MaxItems: aws.Int32(maxGroups),
    })
    if err != nil {
        log.Printf("Couldn't list groups. Here's why: %v\n", err)
    } else {
        groups = result.Groups
    }
}
```



```
    return groups, err
}
```

- Para obtener más información sobre la API, consulta [ListGroups](#) la Referencia AWS SDK para Go de la API.

ListPolicies

En el siguiente ejemplo de código, se muestra cómo utilizar `ListPolicies`.

SDK para Go V2

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 (
    "context"
    "encoding/json"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/iam"
    "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
// actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
    iamClient *iam.Client
}
```

```
// ListPolicies gets up to maxPolicies policies.
func (wrapper PolicyWrapper) ListPolicies(ctx context.Context, maxPolicies int32)
([]types.Policy, error) {
    var policies []types.Policy
    result, err := wrapper.IamClient.ListPolicies(ctx, &iam.ListPoliciesInput{
        MaxItems: aws.Int32(maxPolicies),
    })
    if err != nil {
        log.Printf("Couldn't list policies. Here's why: %v\n", err)
    } else {
        policies = result.Policies
    }
    return policies, err
}
```

- Para obtener más información sobre la API, consulta [ListPolicies](#) la Referencia AWS SDK para Go de la API.

ListRolePolicies

En el siguiente ejemplo de código, se muestra cómo utilizar ListRolePolicies.

SDK para Go V2

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 (
    "context"
    "encoding/json"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/iam"
    "github.com/aws/aws-sdk-go-v2/service/iam/types"
)
```

```

)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
    iamClient *iam.Client
}

// ListRolePolicies lists the inline policies for a role.
func (wrapper RoleWrapper) ListRolePolicies(ctx context.Context, roleName string)
([]string, error) {
    var policies []string
    result, err := wrapper.IamClient.ListRolePolicies(ctx, &iam.ListRolePoliciesInput{
        RoleName: aws.String(roleName),
    })
    if err != nil {
        log.Printf("Couldn't list policies for role %v. Here's why: %v\n", roleName, err)
    } else {
        policies = result.PolicyNames
    }
    return policies, err
}

```

- Para obtener más información sobre la API, consulta [ListRolePolicies](#) la Referencia AWS SDK para Go de la API.

ListRoles

En el siguiente ejemplo de código, se muestra cómo utilizar `ListRoles`.

SDK para Go V2

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 (  
    "context"  
    "encoding/json"  
    "log"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/iam"  
    "github.com/aws/aws-sdk-go-v2/service/iam/types"  
)  
  
// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions  
// used in the examples.  
// It contains an IAM service client that is used to perform role actions.  
type RoleWrapper struct {  
    iamClient *iam.Client  
}  
  
// ListRoles gets up to maxRoles roles.  
func (wrapper RoleWrapper) ListRoles(ctx context.Context, maxRoles int32)  
    ([]types.Role, error) {  
    var roles []types.Role  
    result, err := wrapper.IamClient.ListRoles(ctx,  
        &iam.ListRolesInput{MaxItems: aws.Int32(maxRoles)},  
    )  
    if err != nil {  
        log.Printf("Couldn't list roles. Here's why: %v\n", err)  
    } else {  
        roles = result.Roles  
    }  
    return roles, err  
}
```

- Para obtener más información sobre la API, consulta [ListRoles](#) la Referencia AWS SDK para Go de la API.

ListSAMLProviders

En el siguiente ejemplo de código, se muestra cómo utilizar ListSAMLProviders.

SDK para Go V2

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 (
    "context"
    "log"

    "github.com/aws/aws-sdk-go-v2/service/iam"
    "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account
// actions
// used in the examples.
// It contains an IAM service client that is used to perform account actions.
type AccountWrapper struct {
    iamClient *iam.Client
}

// ListSAMLProviders gets the SAML providers for the account.
func (wrapper AccountWrapper) ListSAMLProviders(ctx context.Context)
    ([]types.SAMLProviderListEntry, error) {
    var providers []types.SAMLProviderListEntry
    result, err := wrapper.IamClient.ListSAMLProviders(ctx,
        &iam.ListSAMLProvidersInput{})
    if err != nil {
        log.Printf("Couldn't list SAML providers. Here's why: %v\n", err)
    } else {
        providers = result.SAMLProviderList
    }
    return providers, err
}
```

```
}
```

- Para obtener más información sobre la API, consulta la [lista SAMLProviders](#) en la referencia AWS SDK para Go de la API.

ListUserPolicies

En el siguiente ejemplo de código, se muestra cómo utilizar ListUserPolicies.

SDK para Go V2

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 (  
    "context"  
    "encoding/json"  
    "errors"  
    "log"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/iam"  
    "github.com/aws/aws-sdk-go-v2/service/iam/types"  
    "github.com/aws/smithy-go"  
)  
  
// UserWrapper encapsulates user actions used in the examples.  
// It contains an IAM service client that is used to perform user actions.  
type UserWrapper struct {  
    iamClient *iam.Client  
}  
  
// ListUserPolicies lists the inline policies for the specified user.
```

```
func (wrapper UserWrapper) ListUserPolicies(ctx context.Context, userName string)
([]string, error) {
    var policies []string
    result, err := wrapper.IamClient.ListUserPolicies(ctx, &iam.ListUserPoliciesInput{
        UserName: aws.String(userName),
    })
    if err != nil {
        log.Printf("Couldn't list policies for user %v. Here's why: %v\n", userName, err)
    } else {
        policies = result.PolicyNames
    }
    return policies, err
}
```

- Para obtener más información sobre la API, consulta [ListUserPolicies](#) la Referencia AWS SDK para Go de la API.

ListUsers

En el siguiente ejemplo de código, se muestra cómo utilizar `ListUsers`.

SDK para Go V2

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 (
    "context"
    "encoding/json"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/iam"
    "github.com/aws/aws-sdk-go-v2/service/iam/types"
)
```

```
"github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
    iamClient *iam.Client
}

// ListUsers gets up to maxUsers number of users.
func (wrapper UserWrapper) ListUsers(ctx context.Context, maxUsers int32)
([]types.User, error) {
    var users []types.User
    result, err := wrapper.IamClient.ListUsers(ctx, &iam.ListUsersInput{
        MaxItems: aws.Int32(maxUsers),
    })
    if err != nil {
        log.Printf("Couldn't list users. Here's why: %v\n", err)
    } else {
        users = result.Users
    }
    return users, err
}
```

- Para obtener más información sobre la API, consulta [ListUsers](#) la Referencia AWS SDK para Go de la API.

PutUserPolicy

En el siguiente ejemplo de código, se muestra cómo utilizar PutUserPolicy.

SDK para Go V2

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 (
    "context"
    "encoding/json"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/iam"
    "github.com/aws/aws-sdk-go-v2/service/iam/types"
    "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
    iamClient *iam.Client
}

// CreateUserPolicy adds an inline policy to a user. This example creates a policy
// that
// grants a list of actions on a specified role.
// PolicyDocument shows how to work with a policy document as a data structure and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper UserWrapper) CreateUserPolicy(ctx context.Context, userName string,
    policyName string, actions []string,
    roleArn string) error {
    policyDoc := PolicyDocument{
        Version: "2012-10-17",
        Statement: []PolicyStatement{{
            Effect: "Allow",
            Action: actions,
            Resource: aws.String(roleArn),
        }},
    }
    policyBytes, err := json.Marshal(policyDoc)
    if err != nil {
        log.Printf("Couldn't create policy document for %v. Here's why: %v\n", roleArn,
            err)
        return err
    }
}

```

```
_, err = wrapper.IamClient.PutUserPolicy(ctx, &iam.PutUserPolicyInput{
    PolicyDocument: aws.String(string(policyBytes)),
    PolicyName:     aws.String(policyName),
    UserName:       aws.String(userName),
})
if err != nil {
    log.Printf("Couldn't create policy for user %v. Here's why: %v\n", userName, err)
}
return err
}
```

- Para obtener más información sobre la API, consulta [PutUserPolicy](#) la Referencia AWS SDK para Go de la API.

Ejemplos de Kinesis usando SDK para Go V2

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar situaciones comunes mediante el uso de la AWS SDK para Go V2 con Kinesis.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Ejemplos de tecnología sin servidor](#)

Ejemplos de tecnología sin servidor

Invocar una función de Lambda desde un desencadenador de Kinesis

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento activado al recibir registros de un flujo de Kinesis. La función recupera la carga útil de Kinesis, la decodifica desde Base64 y registra el contenido del registro.

SDK para Go V2

 Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Uso de un evento de Kinesis con Lambda mediante Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
    "context"
    "log"

    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, kinesisEvent events.KinesisEvent) error {
    if len(kinesisEvent.Records) == 0 {
        log.Printf("empty Kinesis event received")
        return nil
    }

    for _, record := range kinesisEvent.Records {
        log.Printf("processed Kinesis event with EventId: %v", record.EventID)
        recordDataBytes := record.Kinesis.Data
        recordDataText := string(recordDataBytes)
        log.Printf("record data: %v", recordDataText)
        // TODO: Do interesting work based on the new data
    }
    log.Printf("successfully processed %v records", len(kinesisEvent.Records))
    return nil
}

func main() {
    lambda.Start(handler)
}
```

Notificación de los errores de los elementos del lote de las funciones de Lambda mediante un desencadenador de Kinesis

En el siguiente ejemplo de código se muestra cómo implementar una respuesta por lotes parcial para funciones de Lambda que reciben eventos de un flujo de Kinesis. La función informa los errores de los elementos del lote en la respuesta y le indica a Lambda que vuelva a intentar esos mensajes más adelante.

SDK para Go V2

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Notificación de los errores de los elementos del lote de Kinesis con Lambda mediante Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
    "context"
    "fmt"
    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, kinesisEvent events.KinesisEvent)
    (map[string]interface{}, error) {
    batchItemFailures := []map[string]interface{}{}

    for _, record := range kinesisEvent.Records {
        curRecordSequenceNumber := ""

        // Process your record
        if /* Your record processing condition here */ {
            curRecordSequenceNumber = record.Kinesis.SequenceNumber
        }
    }

    return batchItemFailures, nil
}
```

```
}

// Add a condition to check if the record processing failed
if curRecordSequenceNumber != "" {
    batchItemFailures = append(batchItemFailures, map[string]interface{}{
        "itemIdentifier": curRecordSequenceNumber})
    }
}

kinesisBatchResponse := map[string]interface{}{
    "batchItemFailures": batchItemFailures,
}
return kinesisBatchResponse, nil
}

func main() {
    lambda.Start(handler)
}
```

Ejemplos de Lambda usando SDK para Go V2

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso de la AWS SDK para Go V2 con 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.

AWS Las contribuciones de la comunidad son ejemplos que fueron creados y mantenidos por varios equipos de distintos AWS equipos. Para enviar comentarios, utilice el mecanismo previsto en los repositorios vinculados.

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](#)
- [Ejemplos de tecnología sin servidor](#)
- [AWS contribuciones de la comunidad](#)

Introducción

Introducción a Lambda

El siguiente ejemplo de código muestra cómo empezar a utilizar Lambda.

SDK para Go V2

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](#).

```
package main

import (
    "context"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/lambda"
)

// main uses the AWS SDK for Go (v2) to create an AWS Lambda client and list up to
// 10
// functions in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
```

```
func main() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
        fmt.Println(err)
        return
    }
    lambdaClient := lambda.NewFromConfig(sdkConfig)

    maxItems := 10
    fmt.Printf("Let's list up to %v functions for your account.\n", maxItems)
    result, err := lambdaClient.ListFunctions(ctx, &lambda.ListFunctionsInput{
        MaxItems: aws.Int32(int32(maxItems)),
    })
    if err != nil {
        fmt.Printf("Couldn't list functions for your account. Here's why: %v\n", err)
        return
    }
    if len(result.Functions) == 0 {
        fmt.Println("You don't have any functions!")
    } else {
        for _, function := range result.Functions {
            fmt.Printf("\t\t%v\n", *function.FunctionName)
        }
    }
}
```

- Para obtener más información sobre la API, consulta [ListFunctions](#) la Referencia AWS SDK para Go 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 Go V2

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](#).

Cree una situación interactiva que le muestre cómo empezar a usar las funciones de Lambda.

```
import (  
    "archive/zip"  
    "bytes"  
    "context"  
    "encoding/base64"  
    "encoding/json"  
    "errors"  
    "fmt"  
    "log"  
    "os"  
    "strings"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/iam"  
    iamtypes "github.com/aws/aws-sdk-go-v2/service/iam/types"  
    "github.com/aws/aws-sdk-go-v2/service/lambda"  
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"  
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/lambda/actions"  
)  
  
// GetStartedFunctionsScenario shows you how to use AWS Lambda to perform the  
following
```



```
// actions:
//
// 1. Create an AWS Identity and Access Management (IAM) role and Lambda function,
//    then upload handler code.
// 2. Invoke the function with a single parameter and get results.
// 3. Update the function code and configure with an environment variable.
// 4. Invoke the function with new parameters and get results. Display the returned
//    execution log.
// 5. List the functions for your account, then clean up resources.
type GetStartedFunctionsScenario struct {
    sdkConfig      aws.Config
    functionWrapper actions.FunctionWrapper
    questioner      demotools.IQuestioner
    helper          IScenarioHelper
    isTestRun       bool
}

// NewGetStartedFunctionsScenario constructs a GetStartedFunctionsScenario instance
// from a configuration.
// It uses the specified config to get a Lambda client and create wrappers for the
// actions
// used in the scenario.
func NewGetStartedFunctionsScenario(sdkConfig aws.Config, questioner
    demotools.IQuestioner,
    helper IScenarioHelper) GetStartedFunctionsScenario {
    lambdaClient := lambda.NewFromConfig(sdkConfig)
    return GetStartedFunctionsScenario{
        sdkConfig:      sdkConfig,
        functionWrapper: actions.FunctionWrapper{LambdaClient: lambdaClient},
        questioner:      questioner,
        helper:          helper,
    }
}

// Run runs the interactive scenario.
func (scenario GetStartedFunctionsScenario) Run(ctx context.Context) {
    defer func() {
        if r := recover(); r != nil {
            log.Printf("Something went wrong with the demo.\n")
        }
    }()

    log.Println(strings.Repeat("-", 88))
    log.Println("Welcome to the AWS Lambda get started with functions demo.")
}
```

```

log.Println(strings.Repeat("-", 88))

role := scenario.GetOrCreateRole(ctx)
funcName := scenario.CreateFunction(ctx, role)
scenario.InvokeIncrement(ctx, funcName)
scenario.UpdateFunction(ctx, funcName)
scenario.InvokeCalculator(ctx, funcName)
scenario.ListFunctions(ctx)
scenario.Cleanup(ctx, role, funcName)

log.Println(strings.Repeat("-", 88))
log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}

// GetOrCreateRole checks whether the specified role exists and returns it if it
// does.
// Otherwise, a role is created that specifies Lambda as a trusted principal.
// The AWSLambdaBasicExecutionRole managed policy is attached to the role and the
// role
// is returned.
func (scenario GetStartedFunctionsScenario) GetOrCreateRole(ctx context.Context)
    *iamtypes.Role {
    var role *iamtypes.Role
    iamClient := iam.NewFromConfig(scenario.sdkConfig)
    log.Println("First, we need an IAM role that Lambda can assume.")
    roleName := scenario.questioner.Ask("Enter a name for the role:",
    demotools.NotEmpty{})
    getOutput, err := iamClient.GetRole(ctx, &iam.GetRoleInput{
        RoleName: aws.String(roleName)})
    if err != nil {
        var noSuch *iamtypes.NoSuchEntityException
        if errors.As(err, &noSuch) {
            log.Printf("Role %v doesn't exist. Creating it....\n", roleName)
        } else {
            log.Panicf("Couldn't check whether role %v exists. Here's why: %v\n",
                roleName, err)
        }
    } else {
        role = getOutput.Role
        log.Printf("Found role %v.\n", *role.RoleName)
    }
    if role == nil {
        trustPolicy := PolicyDocument{

```

```

    Version: "2012-10-17",
    Statement: []PolicyStatement{{
        Effect:      "Allow",
        Principal: map[string]string{"Service": "lambda.amazonaws.com"},
        Action:      []string{"sts:AssumeRole"},
    }},
}
policyArn := "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
createOutput, err := iamClient.CreateRole(ctx, &iam.CreateRoleInput{
    AssumeRolePolicyDocument: aws.String(trustPolicy.String()),
    RoleName:                  aws.String(roleName),
})
if err != nil {
    log.Panicf("Couldn't create role %v. Here's why: %v\n", roleName, err)
}
role = createOutput.Role
_, err = iamClient.AttachRolePolicy(ctx, &iam.AttachRolePolicyInput{
    PolicyArn: aws.String(policyArn),
    RoleName:  aws.String(roleName),
})
if err != nil {
    log.Panicf("Couldn't attach a policy to role %v. Here's why: %v\n", roleName,
err)
}
log.Printf("Created role %v.\n", *role.RoleName)
log.Println("Let's give AWS a few seconds to propagate resources...")
scenario.helper.Pause(10)
}
log.Println(strings.Repeat("-", 88))
return role
}

// CreateFunction creates a Lambda function and uploads a handler written in Python.
// The code for the Python handler is packaged as a []byte in .zip format.
func (scenario GetStartedFunctionsScenario) CreateFunction(ctx context.Context, role
*iamtypes.Role) string {
    log.Println("Let's create a function that increments a number.\n" +
        "The function uses the 'lambda_handler_basic.py' script found in the \n" +
        "'handlers' directory of this project.")
    funcName := scenario.questioner.Ask("Enter a name for the Lambda function:",
demotools.NotEmpty{})
    zipPackage := scenario.helper.CreateDeploymentPackage("lambda_handler_basic.py",
fmt.Sprintf("%v.py", funcName))
    log.Printf("Creating function %v and waiting for it to be ready.", funcName)

```

```

funcState := scenario.functionWrapper.CreateFunction(ctx, funcName,
fmt.Sprintf("%v.lambda_handler", funcName),
    role.Arn, zipPackage)
log.Printf("Your function is %v.", funcState)
log.Println(strings.Repeat("-", 88))
return funcName
}

// InvokeIncrement invokes a Lambda function that increments a number. The function
// parameters are contained in a Go struct that is used to serialize the parameters
// to
// a JSON payload that is passed to the function.
// The result payload is deserialized into a Go struct that contains an int value.
func (scenario GetStartedFunctionsScenario) InvokeIncrement(ctx context.Context,
    funcName string) {
    parameters := actions.IncrementParameters{Action: "increment"}
    log.Println("Let's invoke our function. This function increments a number.")
    parameters.Number = scenario.questioner.AskInt("Enter a number to increment:",
        demotools.NotEmpty{})
    log.Printf("Invoking %v with %v...\n", funcName, parameters.Number)
    invokeOutput := scenario.functionWrapper.Invoke(ctx, funcName, parameters, false)
    var payload actions.LambdaResultInt
    err := json.Unmarshal(invokeOutput.Payload, &payload)
    if err != nil {
        log.Panicf("Couldn't unmarshal payload from invoking %v. Here's why: %v\n",
            funcName, err)
    }
    log.Printf("Invoking %v with %v returned %v.\n", funcName, parameters.Number,
        payload)
    log.Println(strings.Repeat("-", 88))
}

// UpdateFunction updates the code for a Lambda function by uploading a simple
// arithmetic
// calculator written in Python. The code for the Python handler is packaged as a
// []byte in .zip format.
// After the code is updated, the configuration is also updated with a new log
// level that instructs the handler to log additional information.
func (scenario GetStartedFunctionsScenario) UpdateFunction(ctx context.Context,
    funcName string) {
    log.Println("Let's update the function to an arithmetic calculator.\n" +
        "The function uses the 'lambda_handler_calculator.py' script found in the\n" +
        "'handlers' directory of this project.")
    scenario.questioner.Ask("Press Enter when you're ready.")
}

```

```

log.Println("Creating deployment package...")
zipPackage :=
scenario.helper.CreateDeploymentPackage("lambda_handler_calculator.py",
    fmt.Sprintf("%v.py", funcName))
log.Println("...and updating the Lambda function and waiting for it to be ready.")
funcState := scenario.functionWrapper.UpdateFunctionCode(ctx, funcName, zipPackage)
log.Printf("Updated function %v. Its current state is %v.", funcName, funcState)
log.Println("This function uses an environment variable to control logging level.")
log.Println("Let's set it to DEBUG to get the most logging.")
scenario.functionWrapper.UpdateFunctionConfiguration(ctx, funcName,
    map[string]string{"LOG_LEVEL": "DEBUG"})
log.Println(strings.Repeat("-", 88))
}

// InvokeCalculator invokes the Lambda calculator function. The parameters are
// stored in a
// Go struct that is used to serialize the parameters to a JSON payload. That
// payload is then passed
// to the function.
// The result payload is deserialized to a Go struct that stores the result as
// either an
// int or float32, depending on the kind of operation that was specified.
func (scenario GetStartedFunctionsScenario) InvokeCalculator(ctx context.Context,
    funcName string) {
    wantInvoke := true
    choices := []string{"plus", "minus", "times", "divided-by"}
    for wantInvoke {
        choice := scenario.questioner.AskChoice("Select an arithmetic operation:\n",
            choices)
        x := scenario.questioner.AskInt("Enter a value for x:", demotools.NotEmpty{})
        y := scenario.questioner.AskInt("Enter a value for y:", demotools.NotEmpty{})
        log.Printf("Invoking %v %v %v...", x, choices[choice], y)
        calcParameters := actions.CalculatorParameters{
            Action: choices[choice],
            X:      x,
            Y:      y,
        }
        invokeOutput := scenario.functionWrapper.Invoke(ctx, funcName, calcParameters,
            true)
        var payload any
        if choice == 3 { // divide-by results in a float.
            payload = actions.LambdaResultFloat{}
        } else {
            payload = actions.LambdaResultInt{}
        }
    }
}

```

```

}
err := json.Unmarshal(invokeOutput.Payload, &payload)
if err != nil {
    log.Panicf("Couldn't unmarshal payload from invoking %v. Here's why: %v\n",
        funcName, err)
}
log.Printf("Invoking %v with %v %v %v returned %v.\n", funcName,
    calcParameters.X, calcParameters.Action, calcParameters.Y, payload)
scenario.questioner.Ask("Press Enter to see the logs from the call.")
logRes, err := base64.StdEncoding.DecodeString(*invokeOutput.LogResult)
if err != nil {
    log.Panicf("Couldn't decode log result. Here's why: %v\n", err)
}
log.Println(string(logRes))
wantInvoke = scenario.questioner.AskBool("Do you want to calculate again? (y/n)",
    "y")
}
log.Println(strings.Repeat("-", 88))
}

// ListFunctions lists up to the specified number of functions for your account.
func (scenario GetStartedFunctionsScenario) ListFunctions(ctx context.Context) {
    count := scenario.questioner.AskInt(
        "Let's list functions for your account. How many do you want to see?",
        demotools.NotEmpty{})
    functions := scenario.functionWrapper.ListFunctions(ctx, count)
    log.Printf("Found %v functions:", len(functions))
    for _, function := range functions {
        log.Printf("\t%v", *function.FunctionName)
    }
    log.Println(strings.Repeat("-", 88))
}

// Cleanup removes the IAM and Lambda resources created by the example.
func (scenario GetStartedFunctionsScenario) Cleanup(ctx context.Context, role
    *iamtypes.Role, funcName string) {
    if scenario.questioner.AskBool("Do you want to clean up resources created for this
    example? (y/n)",
        "y") {
        iamClient := iam.NewFromConfig(scenario.sdkConfig)
        policiesOutput, err := iamClient.ListAttachedRolePolicies(ctx,
            &iam.ListAttachedRolePoliciesInput{RoleName: role.RoleName})
        if err != nil {
            log.Panicf("Couldn't get policies attached to role %v. Here's why: %v\n",

```

```

    *role.RoleName, err)
}
for _, policy := range policiesOutput.AttachedPolicies {
    _, err = iamClient.DetachRolePolicy(ctx, &iam.DetachRolePolicyInput{
        PolicyArn: policy.PolicyArn, RoleName: role.RoleName,
    })
    if err != nil {
        log.Panicf("Couldn't detach policy %v from role %v. Here's why: %v\n",
            *policy.PolicyArn, *role.RoleName, err)
    }
}
_, err = iamClient.DeleteRole(ctx, &iam.DeleteRoleInput{RoleName: role.RoleName})
if err != nil {
    log.Panicf("Couldn't delete role %v. Here's why: %v\n", *role.RoleName, err)
}
log.Printf("Deleted role %v.\n", *role.RoleName)

scenario.functionWrapper.DeleteFunction(ctx, funcName)
log.Printf("Deleted function %v.\n", funcName)
} else {
    log.Println("Okay. Don't forget to delete the resources when you're done with
them.")
}
}

// IScenarioHelper abstracts I/O and wait functions from a scenario so that they
// can be mocked for unit testing.
type IScenarioHelper interface {
    Pause(secs int)
    CreateDeploymentPackage(sourceFile string, destinationFile string) *bytes.Buffer
}

// ScenarioHelper lets the caller specify the path to Lambda handler functions.
type ScenarioHelper struct {
    HandlerPath string
}

// Pause waits for the specified number of seconds.
func (helper *ScenarioHelper) Pause(secs int) {
    time.Sleep(time.Duration(secs) * time.Second)
}

// CreateDeploymentPackage creates an AWS Lambda deployment package from a source
file. The

```

```
// deployment package is stored in .zip format in a bytes.Buffer. The buffer can be
// used to pass a []byte to Lambda when creating the function.
// The specified destinationFile is the name to give the file when it's deployed to
// Lambda.
func (helper *ScenarioHelper) CreateDeploymentPackage(sourceFile string,
destinationFile string) *bytes.Buffer {
    var err error
    buffer := &bytes.Buffer{}
    writer := zip.NewWriter(buffer)
    zFile, err := writer.Create(destinationFile)
    if err != nil {
        log.Panicf("Couldn't create destination archive %v. Here's why: %v\n",
destinationFile, err)
    }
    sourceBody, err := os.ReadFile(fmt.Sprintf("%v/%v", helper.HandlerPath,
sourceFile))
    if err != nil {
        log.Panicf("Couldn't read handler source file %v. Here's why: %v\n",
sourceFile, err)
    } else {
        _, err = zFile.Write(sourceBody)
        if err != nil {
            log.Panicf("Couldn't write handler %v to zip archive. Here's why: %v\n",
sourceFile, err)
        }
    }
    err = writer.Close()
    if err != nil {
        log.Panicf("Couldn't close zip writer. Here's why: %v\n", err)
    }
    return buffer
}
```

Cree una estructura que ajuste las acciones individuales de Lambda.

```
import (
    "bytes"
    "context"
    "encoding/json"
    "errors"
```



```

"log"
"time"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/service/lambda"
"github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
    LambdaClient *lambda.Client
}

// GetFunction gets data about the Lambda function specified by functionName.
func (wrapper FunctionWrapper) GetFunction(ctx context.Context, functionName string)
types.State {
    var state types.State
    funcOutput, err := wrapper.LambdaClient.GetFunction(ctx, &lambda.GetFunctionInput{
        FunctionName: aws.String(functionName),
    })
    if err != nil {
        log.Panicf("Couldn't get function %v. Here's why: %v\n", functionName, err)
    } else {
        state = funcOutput.Configuration.State
    }
    return state
}

// CreateFunction creates a new Lambda function from code contained in the
// zipPackage
// buffer. The specified handlerName must match the name of the file and function
// contained in the uploaded code. The role specified by iamRoleArn is assumed by
// Lambda and grants specific permissions.
// When the function already exists, types.StateActive is returned.
// When the function is created, a lambda.FunctionActiveV2Waiter is used to wait
// until the
// function is active.
func (wrapper FunctionWrapper) CreateFunction(ctx context.Context, functionName
string, handlerName string,
iamRoleArn *string, zipPackage *bytes.Buffer) types.State {

```

```

var state types.State
_, err := wrapper.LambdaClient.CreateFunction(ctx, &lambda.CreateFunctionInput{
    Code:          &types.FunctionCode{ZipFile: zipPackage.Bytes()},
    FunctionName:  aws.String(functionName),
    Role:          iamRoleArn,
    Handler:       aws.String(handlerName),
    Publish:       true,
    Runtime:       types.RuntimePython39,
})
if err != nil {
    var resConflict *types.ResourceConflictException
    if errors.As(err, &resConflict) {
        log.Printf("Function %v already exists.\n", functionName)
        state = types.StateActive
    } else {
        log.Panicf("Couldn't create function %v. Here's why: %v\n", functionName, err)
    }
} else {
    waiter := lambda.NewFunctionActiveV2Waiter(wrapper.LambdaClient)
    funcOutput, err := waiter.WaitForOutput(ctx, &lambda.GetFunctionInput{
        FunctionName: aws.String(functionName)}, 1*time.Minute)
    if err != nil {
        log.Panicf("Couldn't wait for function %v to be active. Here's why: %v\n",
functionName, err)
    } else {
        state = funcOutput.Configuration.State
    }
}
return state
}

// UpdateFunctionCode updates the code for the Lambda function specified by
// functionName.
// The existing code for the Lambda function is entirely replaced by the code in the
// zipPackage buffer. After the update action is called, a
// lambda.FunctionUpdatedV2Waiter
// is used to wait until the update is successful.
func (wrapper FunctionWrapper) UpdateFunctionCode(ctx context.Context, functionName
string, zipPackage *bytes.Buffer) types.State {
    var state types.State
    _, err := wrapper.LambdaClient.UpdateFunctionCode(ctx,
&lambda.UpdateFunctionCodeInput{

```

```

    FunctionName: aws.String(functionName), ZipFile: zipPackage.Bytes(),
})
if err != nil {
    log.Panicf("Couldn't update code for function %v. Here's why: %v\n", functionName,
err)
} else {
    waiter := lambda.NewFunctionUpdatedV2Waiter(wrapper.LambdaClient)
    funcOutput, err := waiter.WaitForOutput(ctx, &lambda.GetFunctionInput{
        FunctionName: aws.String(functionName)}, 1*time.Minute)
    if err != nil {
        log.Panicf("Couldn't wait for function %v to be active. Here's why: %v\n",
functionName, err)
    } else {
        state = funcOutput.Configuration.State
    }
}
return state
}

// UpdateFunctionConfiguration updates a map of environment variables configured for
// the Lambda function specified by functionName.
func (wrapper FunctionWrapper) UpdateFunctionConfiguration(ctx context.Context,
functionName string, envVars map[string]string) {
_, err := wrapper.LambdaClient.UpdateFunctionConfiguration(ctx,
&lambda.UpdateFunctionConfigurationInput{
    FunctionName: aws.String(functionName),
    Environment: &types.Environment{Variables: envVars},
})
if err != nil {
    log.Panicf("Couldn't update configuration for %v. Here's why: %v", functionName,
err)
}
}

// ListFunctions lists up to maxItems functions for the account. This function uses
a
// lambda.ListFunctionsPaginator to paginate the results.
func (wrapper FunctionWrapper) ListFunctions(ctx context.Context, maxItems int)
[]types.FunctionConfiguration {
    var functions []types.FunctionConfiguration

```

```

paginator := lambda.NewListFunctionsPaginator(wrapper.LambdaClient,
&lambda.ListFunctionsInput{
    MaxItems: aws.Int32(int32(maxItems)),
})
for paginator.HasMorePages() && len(functions) < maxItems {
    pageOutput, err := paginator.NextPage(ctx)
    if err != nil {
        log.Panicf("Couldn't list functions for your account. Here's why: %v\n", err)
    }
    functions = append(functions, pageOutput.Functions...)
}
return functions
}

// DeleteFunction deletes the Lambda function specified by functionName.
func (wrapper FunctionWrapper) DeleteFunction(ctx context.Context, functionName
string) {
    _, err := wrapper.LambdaClient.DeleteFunction(ctx, &lambda.DeleteFunctionInput{
        FunctionName: aws.String(functionName),
    })
    if err != nil {
        log.Panicf("Couldn't delete function %v. Here's why: %v\n", functionName, err)
    }
}

// Invoke invokes the Lambda function specified by functionName, passing the
parameters
// as a JSON payload. When getLog is true, types.LogTypeTail is specified, which
tells
// Lambda to include the last few log lines in the returned result.
func (wrapper FunctionWrapper) Invoke(ctx context.Context, functionName string,
parameters any, getLog bool) *lambda.InvokeOutput {
    logType := types.LogTypeNone
    if getLog {
        logType = types.LogTypeTail
    }
    payload, err := json.Marshal(parameters)
    if err != nil {
        log.Panicf("Couldn't marshal parameters to JSON. Here's why %v\n", err)
    }
}

```

```

invokeOutput, err := wrapper.LambdaClient.Invoke(ctx, &lambda.InvokeInput{
    FunctionName: aws.String(functionName),
    LogType:      logType,
    Payload:      payload,
})
if err != nil {
    log.Panicf("Couldn't invoke function %v. Here's why: %v\n", functionName, err)
}
return invokeOutput
}

// IncrementParameters is used to serialize parameters to the increment Lambda
// handler.
type IncrementParameters struct {
    Action string `json:"action"`
    Number int    `json:"number"`
}

// CalculatorParameters is used to serialize parameters to the calculator Lambda
// handler.
type CalculatorParameters struct {
    Action string `json:"action"`
    X      int    `json:"x"`
    Y      int    `json:"y"`
}

// LambdaResultInt is used to deserialize an int result from a Lambda handler.
type LambdaResultInt struct {
    Result int `json:"result"`
}

// LambdaResultFloat is used to deserialize a float32 result from a Lambda handler.
type LambdaResultFloat struct {
    Result float32 `json:"result"`
}

```

Defina un controlador de Lambda que aumente un número.

```
import logging
```

```
logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    """
    Accepts an action and a single number, performs the specified action on the
    number,
    and returns the result. The only allowable action is 'increment'.

    :param event: The event dict that contains the parameters sent when the function
                  is invoked.
    :param context: The context in which the function is called.
    :return: The result of the action.
    """
    result = None
    action = event.get("action")
    if action == "increment":
        result = event.get("number", 0) + 1
        logger.info("Calculated result of %s", result)
    else:
        logger.error("%s is not a valid action.", action)

    response = {"result": result}
    return response
```

Defina un segundo controlador de Lambda que realice operaciones aritméticas.

```
import logging
import os

logger = logging.getLogger()

# Define a list of Python lambda functions that are called by this AWS Lambda
function.
ACTIONS = {
    "plus": lambda x, y: x + y,
    "minus": lambda x, y: x - y,
```

```

    "times": lambda x, y: x * y,
    "divided-by": lambda x, y: x / y,
}

def lambda_handler(event, context):
    """
    Accepts an action and two numbers, performs the specified action on the numbers,
    and returns the result.

    :param event: The event dict that contains the parameters sent when the function
                  is invoked.
    :param context: The context in which the function is called.
    :return: The result of the specified action.
    """
    # Set the log level based on a variable configured in the Lambda environment.
    logger.setLevel(os.environ.get("LOG_LEVEL", logging.INFO))
    logger.debug("Event: %s", event)

    action = event.get("action")
    func = ACTIONS.get(action)
    x = event.get("x")
    y = event.get("y")
    result = None
    try:
        if func is not None and x is not None and y is not None:
            result = func(x, y)
            logger.info("%s %s %s is %s", x, action, y, result)
        else:
            logger.error("I can't calculate %s %s %s.", x, action, y)
    except ZeroDivisionError:
        logger.warning("I can't divide %s by 0!", x)

    response = {"result": result}
    return response

```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK para Go .
 - [CreateFunction](#)

- [DeleteFunction](#)
- [GetFunction](#)
- [Invoke](#)
- [ListFunctions](#)
- [UpdateFunctionCode](#)
- [UpdateFunctionConfiguration](#)

Acciones

CreateFunction

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateFunction`.

SDK para Go V2

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
import (  
    "bytes"  
    "context"  
    "encoding/json"  
    "errors"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/lambda"  
    "github.com/aws/aws-sdk-go-v2/service/lambda/types"  
)  
  
// FunctionWrapper encapsulates function actions used in the examples.  
// It contains an AWS Lambda service client that is used to perform user actions.  
type FunctionWrapper struct {  
    LambdaClient *lambda.Client
```



```

}

// CreateFunction creates a new Lambda function from code contained in the
// zipPackage
// buffer. The specified handlerName must match the name of the file and function
// contained in the uploaded code. The role specified by iamRoleArn is assumed by
// Lambda and grants specific permissions.
// When the function already exists, types.StateActive is returned.
// When the function is created, a lambda.FunctionActiveV2Waiter is used to wait
// until the
// function is active.
func (wrapper FunctionWrapper) CreateFunction(ctx context.Context, functionName
string, handlerName string,
iamRoleArn *string, zipPackage *bytes.Buffer) types.State {
var state types.State
_, err := wrapper.LambdaClient.CreateFunction(ctx, &lambda.CreateFunctionInput{
Code:      &types.FunctionCode{ZipFile: zipPackage.Bytes()},
FunctionName: aws.String(functionName),
Role:      iamRoleArn,
Handler:   aws.String(handlerName),
Publish:   true,
Runtime:   types.RuntimePython39,
})
if err != nil {
var resConflict *types.ResourceConflictException
if errors.As(err, &resConflict) {
log.Printf("Function %v already exists.\n", functionName)
state = types.StateActive
} else {
log.Panicf("Couldn't create function %v. Here's why: %v\n", functionName, err)
}
} else {
waiter := lambda.NewFunctionActiveV2Waiter(wrapper.LambdaClient)
funcOutput, err := waiter.WaitForOutput(ctx, &lambda.GetFunctionInput{
FunctionName: aws.String(functionName)}, 1*time.Minute)
if err != nil {
log.Panicf("Couldn't wait for function %v to be active. Here's why: %v\n",
functionName, err)
} else {
state = funcOutput.Configuration.State
}
}
}

```

```
    return state
}
```

- Para obtener más información sobre la API, consulta [CreateFunction](#) la Referencia AWS SDK para Go de la API.

DeleteFunction

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteFunction`.

SDK para Go V2

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 (
    "bytes"
    "context"
    "encoding/json"
    "errors"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/lambda"
    "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
    LambdaClient *lambda.Client
}
```

```
// DeleteFunction deletes the Lambda function specified by functionName.
func (wrapper FunctionWrapper) DeleteFunction(ctx context.Context, functionName
string) {
    _, err := wrapper.LambdaClient.DeleteFunction(ctx, &lambda.DeleteFunctionInput{
        FunctionName: aws.String(functionName),
    })
    if err != nil {
        log.Panicf("Couldn't delete function %v. Here's why: %v\n", functionName, err)
    }
}
```

- Para obtener más información sobre la API, consulta [DeleteFunction](#) la Referencia AWS SDK para Go de la API.

GetFunction

En el siguiente ejemplo de código, se muestra cómo utilizar `GetFunction`.

SDK para Go V2

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 (
    "bytes"
    "context"
    "encoding/json"
    "errors"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/lambda"
    "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)
```

```
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
    LambdaClient *lambda.Client
}

// GetFunction gets data about the Lambda function specified by functionName.
func (wrapper FunctionWrapper) GetFunction(ctx context.Context, functionName string)
types.State {
    var state types.State
    funcOutput, err := wrapper.LambdaClient.GetFunction(ctx, &lambda.GetFunctionInput{
        FunctionName: aws.String(functionName),
    })
    if err != nil {
        log.Panicf("Couldn't get function %v. Here's why: %v\n", functionName, err)
    } else {
        state = funcOutput.Configuration.State
    }
    return state
}
```

- Para obtener más información sobre la API, consulta [GetFunction](#) la Referencia AWS SDK para Go de la API.

Invoke

En el siguiente ejemplo de código, se muestra cómo utilizar Invoke.

SDK para Go V2

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 (
    "bytes"
    "context"
    "encoding/json"
    "errors"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/lambda"
    "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
    LambdaClient *lambda.Client
}

// Invoke invokes the Lambda function specified by functionName, passing the
// parameters
// as a JSON payload. When getLog is true, types.LogTypeTail is specified, which
// tells
// Lambda to include the last few log lines in the returned result.
func (wrapper FunctionWrapper) Invoke(ctx context.Context, functionName string,
    parameters any, getLog bool) *lambda.InvokeOutput {
    logType := types.LogTypeNone
    if getLog {
        logType = types.LogTypeTail
    }
    payload, err := json.Marshal(parameters)
    if err != nil {
        log.Panicf("Couldn't marshal parameters to JSON. Here's why %v\n", err)
    }
    invokeOutput, err := wrapper.LambdaClient.Invoke(ctx, &lambda.InvokeInput{
        FunctionName: aws.String(functionName),
        LogType:      logType,
        Payload:      payload,
    })
    if err != nil {

```

```
    log.Panicf("Couldn't invoke function %v. Here's why: %v\n", functionName, err)
}
return invokeOutput
}
```

- Para obtener información sobre la API, consulte [Invocar](#) en la Referencia de la API de AWS SDK para Go .

ListFunctions

En el siguiente ejemplo de código, se muestra cómo utilizar `ListFunctions`.

SDK para Go V2

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
import (
    "bytes"
    "context"
    "encoding/json"
    "errors"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/lambda"
    "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
    LambdaClient *lambda.Client
}
```

```
// ListFunctions lists up to maxItems functions for the account. This function uses
// a
// lambda.ListFunctionsPaginator to paginate the results.
func (wrapper FunctionWrapper) ListFunctions(ctx context.Context, maxItems int)
[]types.FunctionConfiguration {
    var functions []types.FunctionConfiguration
    paginator := lambda.NewListFunctionsPaginator(wrapper.LambdaClient,
        &lambda.ListFunctionsInput{
            MaxItems: aws.Int32(int32(maxItems)),
        })
    for paginator.HasMorePages() && len(functions) < maxItems {
        pageOutput, err := paginator.NextPage(ctx)
        if err != nil {
            log.Panicf("Couldn't list functions for your account. Here's why: %v\n", err)
        }
        functions = append(functions, pageOutput.Functions...)
    }
    return functions
}
```

- Para obtener más información sobre la API, consulta [ListFunctions](#) la Referencia AWS SDK para Go de la API.

UpdateFunctionCode

En el siguiente ejemplo de código, se muestra cómo utilizar UpdateFunctionCode.

SDK para Go V2

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 (
    "bytes"
    "context"
    "encoding/json"
    "errors"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/lambda"
    "github.com/aws/aws-sdk-go-v2/service/lambda/types"
)

// FunctionWrapper encapsulates function actions used in the examples.
// It contains an AWS Lambda service client that is used to perform user actions.
type FunctionWrapper struct {
    LambdaClient *lambda.Client
}

// UpdateFunctionCode updates the code for the Lambda function specified by
// functionName.
// The existing code for the Lambda function is entirely replaced by the code in the
// zipPackage buffer. After the update action is called, a
// lambda.FunctionUpdatedV2Waiter
// is used to wait until the update is successful.
func (wrapper FunctionWrapper) UpdateFunctionCode(ctx context.Context, functionName
    string, zipPackage *bytes.Buffer) types.State {
    var state types.State
    _, err := wrapper.LambdaClient.UpdateFunctionCode(ctx,
        &lambda.UpdateFunctionCodeInput{
            FunctionName: aws.String(functionName), ZipFile: zipPackage.Bytes(),
        })
    if err != nil {
        log.Panicf("Couldn't update code for function %v. Here's why: %v\n", functionName,
            err)
    } else {
        waiter := lambda.NewFunctionUpdatedV2Waiter(wrapper.LambdaClient)
        funcOutput, err := waiter.WaitForOutput(ctx, &lambda.GetFunctionInput{
            FunctionName: aws.String(functionName)}, 1*time.Minute)
        if err != nil {
            log.Panicf("Couldn't wait for function %v to be active. Here's why: %v\n",
                functionName, err)
        }
    }
}

```



```
    } else {  
        state = funcOutput.Configuration.State  
    }  
}  
return state  
}
```

- Para obtener más información sobre la API, consulta [UpdateFunctionCode](#) la Referencia AWS SDK para Go de la API.

UpdateFunctionConfiguration

En el siguiente ejemplo de código, se muestra cómo utilizar UpdateFunctionConfiguration.

SDK para Go V2

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 (  
    "bytes"  
    "context"  
    "encoding/json"  
    "errors"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/lambda"  
    "github.com/aws/aws-sdk-go-v2/service/lambda/types"  
)  
  
// FunctionWrapper encapsulates function actions used in the examples.  
// It contains an AWS Lambda service client that is used to perform user actions.  
type FunctionWrapper struct {
```

```
LambdaClient *lambda.Client
}

// UpdateFunctionConfiguration updates a map of environment variables configured for
// the Lambda function specified by functionName.
func (wrapper FunctionWrapper) UpdateFunctionConfiguration(ctx context.Context,
    functionName string, envVars map[string]string) {
    _, err := wrapper.LambdaClient.UpdateFunctionConfiguration(ctx,
        &lambda.UpdateFunctionConfigurationInput{
            FunctionName: aws.String(functionName),
            Environment: &types.Environment{Variables: envVars},
        })
    if err != nil {
        log.Panicf("Couldn't update configuration for %v. Here's why: %v", functionName,
            err)
    }
}
```

- Para obtener más información sobre la API, consulta [UpdateFunctionConfiguration](#) la Referencia AWS SDK para Go de la API.

Escenarios

Confirmación de manera automática a los usuarios conocidos con una función de Lambda

En el siguiente ejemplo de código, se muestra cómo confirmar de manera automática los usuarios conocidos de Amazon Cognito con una función de Lambda.

- Configure un grupo de usuarios para que llame a una función de Lambda para el desencadenador PreSignUp.
- Inscripción de un usuario mediante Amazon Cognito
- La función de Lambda escanea una tabla de DynamoDB y confirma de manera automática los usuarios conocidos.
- Inicie sesión con el nuevo usuario y, luego, elimine los recursos.

SDK para Go V2

 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](#).

Ejecutar un escenario interactivo en un símbolo del sistema.

```
import (
    "context"
    "errors"
    "log"
    "strings"
    "user_pools_and_lambda_triggers/actions"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// AutoConfirm separates the steps of this scenario into individual functions so
// that
// they are simpler to read and understand.
type AutoConfirm struct {
    helper      IScenarioHelper
    questioner  demotools.IQuestioner
    resources   Resources
    cognitoActor *actions.CognitoActions
}

// NewAutoConfirm constructs a new auto confirm runner.
func NewAutoConfirm(sdkConfig aws.Config, questioner demotools.IQuestioner, helper
IScenarioHelper) AutoConfirm {
    scenario := AutoConfirm{
        helper:      helper,
        questioner:  questioner,
        resources:   Resources{},
        cognitoActor: &actions.CognitoActions{CognitoClient:
cognitoidentityprovider.NewFromConfig(sdkConfig)},
```

```

}
scenario.resources.init(scenario.cognitoActor, questioner)
return scenario
}

// AddPreSignUpTrigger adds a Lambda handler as an invocation target for the
// PreSignUp trigger.
func (runner *AutoConfirm) AddPreSignUpTrigger(ctx context.Context, userPoolId
string, functionArn string) {
log.Printf("Let's add a Lambda function to handle the PreSignUp trigger from
Cognito.\n" +
"This trigger happens when a user signs up, and lets your function take action
before the main Cognito\n" +
"sign up processing occurs.\n")
err := runner.cognitoActor.UpdateTriggers(
ctx, userPoolId,
actions.TriggerInfo{Trigger: actions.PreSignUp, HandlerArn:
aws.String(functionArn)})
if err != nil {
panic(err)
}
log.Printf("Lambda function %v added to user pool %v to handle the PreSignUp
trigger.\n",
functionArn, userPoolId)
}

// SignUpUser signs up a user from the known user table with a password you specify.
func (runner *AutoConfirm) SignUpUser(ctx context.Context, clientId string,
usersTable string) (string, string) {
log.Println("Let's sign up a user to your Cognito user pool. When the user's email
matches an email in the\n" +
"DynamoDB known users table, it is automatically verified and the user is
confirmed.")

knownUsers, err := runner.helper.GetKnownUsers(ctx, usersTable)
if err != nil {
panic(err)
}
userChoice := runner.questioner.AskChoice("Which user do you want to use?\n",
knownUsers.UserNameList())
user := knownUsers.Users[userChoice]

var signedUp bool
var userConfirmed bool

```

```

password := runner.questioner.AskPassword("Enter a password that has at least eight
characters, uppercase, lowercase, numbers and symbols.\n"+
    "(the password will not display as you type):", 8)
for !signedUp {
    log.Printf("Signing up user '%v' with email '%v' to Cognito.\n", user.UserName,
user.Email)
    userConfirmed, err = runner.cognitoActor.SignUp(ctx, clientId, user.UserName,
password, user.Email)
    if err != nil {
        var invalidPassword *types.InvalidPasswordException
        if errors.As(err, &invalidPassword) {
            password = runner.questioner.AskPassword("Enter another password:", 8)
        } else {
            panic(err)
        }
    } else {
        signedUp = true
    }
}
log.Printf("User %v signed up, confirmed = %v.\n", user.UserName, userConfirmed)

log.Println(strings.Repeat("-", 88))

return user.UserName, password
}

// SignInUser signs in a user.
func (runner *AutoConfirm) SignInUser(ctx context.Context, clientId string, userName
string, password string) string {
    runner.questioner.Ask("Press Enter when you're ready to continue.")
    log.Printf("Let's sign in as %v...\n", userName)
    authResult, err := runner.cognitoActor.SignIn(ctx, clientId, userName, password)
    if err != nil {
        panic(err)
    }
    log.Printf("Successfully signed in. Your access token starts with: %v...\n",
(*authResult.AccessToken)[:10])
    log.Println(strings.Repeat("-", 88))
    return *authResult.AccessToken
}

// Run runs the scenario.
func (runner *AutoConfirm) Run(ctx context.Context, stackName string) {
    defer func() {

```

```

    if r := recover(); r != nil {
        log.Println("Something went wrong with the demo.")
        runner.resources.Cleanup(ctx)
    }
}()

log.Println(strings.Repeat("-", 88))
log.Printf("Welcome\n")

log.Println(strings.Repeat("-", 88))

stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
if err != nil {
    panic(err)
}
runner.resources.userPoolId = stackOutputs["UserPoolId"]
runner.helper.PopulateUserTable(ctx, stackOutputs["TableName"])

runner.AddPreSignUpTrigger(ctx, stackOutputs["UserPoolId"],
    stackOutputs["AutoConfirmFunctionArn"])
runner.resources.triggers = append(runner.resources.triggers, actions.PreSignUp)
userName, password := runner.SignUpUser(ctx, stackOutputs["UserPoolClientId"],
    stackOutputs["TableName"])
runner.helper.ListRecentLogEvents(ctx, stackOutputs["AutoConfirmFunction"])
runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
    runner.SignInUser(ctx, stackOutputs["UserPoolClientId"], userName, password))

runner.resources.Cleanup(ctx)

log.Println(strings.Repeat("-", 88))
log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}

```

Controle el desencadenador PreSignUp con una función de Lambda.

```

import (
    "context"
    "log"
    "os"

```

```

"github.com/aws/aws-lambda-go/events"
"github.com/aws/aws-lambda-go/lambda"
"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/config"
"github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
"github.com/aws/aws-sdk-go-v2/service/dynamodb"
dynamodbtypes "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
// format.
type UserInfo struct {
    Username string `dynamodbav:"Username"`
    UserEmail string `dynamodbav:"UserEmail"`
}

// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
    userEmail, err := attributevalue.Marshal(user.UserEmail)
    if err != nil {
        panic(err)
    }
    return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
    dynamoClient *dynamodb.Client
}

// HandleRequest handles the PreSignUp event by looking up a user in an Amazon
// DynamoDB table and
// specifying whether they should be confirmed and verified.
func (h *handler) HandleRequest(ctx context.Context, event
events.CognitoEventUserPoolsPreSignup) (events.CognitoEventUserPoolsPreSignup,
error) {
    log.Printf("Received presignup from %v for user '%v'", event.TriggerSource,
event.Username)
    if event.TriggerSource != "PreSignUp_SignUp" {
        // Other trigger sources, such as PreSignUp_AdminInitiateAuth, ignore the response
        from this handler.
        return event, nil
    }
}

```

```

}
tableName := os.Getenv(TABLE_NAME)
user := UserInfo{
    UserEmail: event.Request.UserAttributes["email"],
}
log.Printf("Looking up email %v in table %v.\n", user.UserEmail, tableName)
output, err := h.dynamoClient.GetItem(ctx, &dynamodb.GetItemInput{
    Key:      user.GetKey(),
    TableName: aws.String(tableName),
})
if err != nil {
    log.Printf("Error looking up email %v.\n", user.UserEmail)
    return event, err
}
if output.Item == nil {
    log.Printf("Email %v not found. Email verification is required.\n",
user.UserEmail)
    return event, err
}

err = attributevalue.UnmarshalMap(output.Item, &user)
if err != nil {
    log.Printf("Couldn't unmarshal DynamoDB item. Here's why: %v\n", err)
    return event, err
}

if user.UserName != event.UserName {
    log.Printf("UserEmail %v found, but stored UserName '%v' does not match supplied
UserName '%v'. Verification is required.\n",
        user.UserEmail, user.UserName, event.UserName)
} else {
    log.Printf("UserEmail %v found with matching UserName %v. User is confirmed.\n",
user.UserEmail, user.UserName)
    event.Response.AutoConfirmUser = true
    event.Response.AutoVerifyEmail = true
}

return event, err
}

func main() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {

```



```

    log.Panicln(err)
}
h := handler{
    dynamoClient: dynamodb.NewFromConfig(sdkConfig),
}
lambda.Start(h.HandleRequest)
}

```

Cree una estructura que lleve a cabo las tareas habituales.

```

import (
    "context"
    "log"
    "strings"
    "time"
    "user_pools_and_lambda_triggers/actions"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cloudformation"
    "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this example.
type IScenarioHelper interface {
    Pause(secs int)
    GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs, error)
    PopulateUserTable(ctx context.Context, tableName string)
    GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
    AddKnownUser(ctx context.Context, tableName string, user actions.User)
    ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this example.
type ScenarioHelper struct {
    questioner demotools.IQuestioner
    dynamoActor *actions.DynamoActions
}

```

```

    cfnActor      *actions.CloudFormationActions
    cwlActor      *actions.CloudWatchLogsActions
    isTestRun     bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
    ScenarioHelper {
    scenario := ScenarioHelper{
        questioner: questioner,
        dynamoActor: &actions.DynamoActions{DynamoClient:
            dynamodb.NewFromConfig(sdkConfig)},
        cfnActor:      &actions.CloudFormationActions{CfnClient:
            cloudformation.NewFromConfig(sdkConfig)},
        cwlActor:      &actions.CloudWatchLogsActions{CwlClient:
            cloudwatchlogs.NewFromConfig(sdkConfig)},
    }
    return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
    if !helper.isTestRun {
        time.Sleep(time.Duration(secs) * time.Second)
    }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
    structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName string)
    (actions.StackOutputs, error) {
    return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
    string) {
    log.Printf("First, let's add some users to the DynamoDB %v table we'll use for this
        example.\n", tableName)
    err := helper.dynamoActor.PopulateTable(ctx, tableName)
    if err != nil {
        panic(err)
    }
}

```

```

// GetKnownUsers gets the users from the known users table in a structured format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
(actions.UserList, error) {
    knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
    if err != nil {
        log.Printf("Couldn't get known users from table %v. Here's why: %v\n", tableName,
err)
    }
    return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
user actions.User) {
    log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users table...
\n",
        user.UserName, user.UserEmail)
    err := helper.dynamoActor.AddUser(ctx, tableName, user)
    if err != nil {
        panic(err)
    }
}

// ListRecentLogEvents gets the most recent log stream and events for the specified
Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context, functionName
string) {
    log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
    helper.Pause(10)
    log.Println("Okay, let's check the logs to find what's happened recently with your
Lambda function.")
    logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
    if err != nil {
        panic(err)
    }
    log.Printf("Getting some recent events from log stream %v\n",
*logStream.LogStreamName)
    events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
*logStream.LogStreamName, 10)
    if err != nil {
        panic(err)
    }
    for _, event := range events {

```

```

    log.Printf("\t%v", *event.Message)
}
log.Println(strings.Repeat("-", 88))
}

```

Cree una estructura que ajuste las acciones de Amazon Cognito.

```

import (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
    CognitoClient *cognitoidentityprovider.Client
}

// Trigger and TriggerInfo define typed data for updating an Amazon Cognito trigger.
type Trigger int

const (
    PreSignUp Trigger = iota
    UserMigration
    PostAuthentication
)

type TriggerInfo struct {
    Trigger    Trigger
    HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger is
// specified with a `nil` value,
// it is removed from the user pool.

```

```

func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId string,
    triggers ...TriggerInfo) error {
    output, err := actor.CognitoClient.DescribeUserPool(ctx,
        &cognitoidentityprovider.DescribeUserPoolInput{
            UserPoolId: aws.String(userPoolId),
        })
    if err != nil {
        log.Printf("Couldn't get info about user pool %v. Here's why: %v\n", userPoolId,
            err)
        return err
    }
    lambdaConfig := output.UserPool.LambdaConfig
    for _, trigger := range triggers {
        switch trigger.Trigger {
            case PreSignUp:
                lambdaConfig.PreSignUp = trigger.HandlerArn
            case UserMigration:
                lambdaConfig.UserMigration = trigger.HandlerArn
            case PostAuthentication:
                lambdaConfig.PostAuthentication = trigger.HandlerArn
        }
    }
    _, err = actor.CognitoClient.UpdateUserPool(ctx,
        &cognitoidentityprovider.UpdateUserPoolInput{
            UserPoolId:    aws.String(userPoolId),
            LambdaConfig: lambdaConfig,
        })
    if err != nil {
        log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
    }
    return err
}

```

// SignUp signs up a user with Amazon Cognito.

```

func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
    string, password string, userEmail string) (bool, error) {
    confirmed := false
    output, err := actor.CognitoClient.SignUp(ctx,
        &cognitoidentityprovider.SignUpInput{
            ClientId: aws.String(clientId),
            Password: aws.String(password),
            Username: aws.String(userName),

```

```

    UserAttributes: []types.AttributeType{
        {Name: aws.String("email"), Value: aws.String(userEmail)},
    },
})
if err != nil {
    var invalidPassword *types.InvalidPasswordException
    if errors.As(err, &invalidPassword) {
        log.Println(*invalidPassword.Message)
    } else {
        log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
    }
} else {
    confirmed = output.UserConfirmed
}
return confirmed, err
}

// SignIn signs in a user to Amazon Cognito using a username and password
// authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
string, password string) (*types.AuthenticationResultType, error) {
    var authResult *types.AuthenticationResultType
    output, err := actor.CognitoClient.InitiateAuth(ctx,
    &cognitoidentityprovider.InitiateAuthInput{
        AuthFlow:      "USER_PASSWORD_AUTH",
        ClientId:      aws.String(clientId),
        AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
    })
    if err != nil {
        var resetRequired *types.PasswordResetRequiredException
        if errors.As(err, &resetRequired) {
            log.Println(*resetRequired.Message)
        } else {
            log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
        }
    } else {
        authResult = output.AuthenticationResult
    }
    return authResult, err
}

```

```

// ForgotPassword starts a password recovery flow for a user. This flow typically
// sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
    userName string) (*types.CodeDeliveryDetailsType, error) {
    output, err := actor.CognitoClient.ForgotPassword(ctx,
        &cognitoidentityprovider.ForgotPasswordInput{
            ClientId: aws.String(clientId),
            Username: aws.String(userName),
        })
    if err != nil {
        log.Printf("Couldn't start password reset for user '%v'. Here's why: %v\n",
            userName, err)
    }
    return output.CodeDeliveryDetails, err
}

// ConfirmForgotPassword confirms a user with a confirmation code and a new
// password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
    string, code string, userName string, password string) error {
    _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
        &cognitoidentityprovider.ConfirmForgotPasswordInput{
            ClientId:      aws.String(clientId),
            ConfirmationCode: aws.String(code),
            Password:     aws.String(password),
            Username:     aws.String(userName),
        })
    if err != nil {
        var invalidPassword *types.InvalidPasswordException
        if errors.As(err, &invalidPassword) {
            log.Println(*invalidPassword.Message)
        } else {
            log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
        }
    }
    return err
}

```

```
// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken string)
    error {
    _, err := actor.CognitoClient.DeleteUser(ctx,
    &cognitoidentityprovider.DeleteUserInput{
        AccessToken: aws.String(userAccessToken),
    })
    if err != nil {
        log.Printf("Couldn't delete user. Here's why: %v\n", err)
    }
    return err
}

// AdminCreateUser uses administrator credentials to add a user to a user pool. This
    method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId string,
    userName string, userEmail string) error {
    _, err := actor.CognitoClient.AdminCreateUser(ctx,
    &cognitoidentityprovider.AdminCreateUserInput{
        UserPoolId:      aws.String(userPoolId),
        Username:        aws.String(userName),
        MessageAction:   types.MessageActionTypeSuppress,
        UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
aws.String(userEmail)}}},
    })
    if err != nil {
        var userExists *types.UsernameExistsException
        if errors.As(err, &userExists) {
            log.Printf("User %v already exists in the user pool.", userName)
            err = nil
        } else {
            log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
        }
    }
    return err
}

// AdminSetUserPassword uses administrator credentials to set a password for a user
    without requiring a
```



```
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
string, userName string, password string) error {
_, err := actor.CognitoClient.AdminSetUserPassword(ctx,
&cognitoidentityprovider.AdminSetUserPasswordInput{
    Password:    aws.String(password),
    UserPoolId:  aws.String(userPoolId),
    Username:    aws.String(userName),
    Permanent:   true,
})
if err != nil {
    var invalidPassword *types.InvalidPasswordException
    if errors.As(err, &invalidPassword) {
        log.Println(*invalidPassword.Message)
    } else {
        log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName, err)
    }
}
return err
}
```

Cree una estructura que ajuste las acciones de DynamoDB.

```
import (
    "context"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
// actions
// used in the examples.
type DynamoActions struct {
    DynamoClient *dynamodb.Client
}
```

```

// User defines structured user data.
type User struct {
    Username string
    UserEmail string
    LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
    UserPoolId string
    ClientId string
    Time string
}

// UserList defines a list of users.
type UserList struct {
    Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of strings.
func (users *UserList) UserNameList() []string {
    names := make([]string, len(users.Users))
    for i := 0; i < len(users.Users); i++ {
        names[i] = users.Users[i].Username
    }
    return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
    error {
    var err error
    var item map[string]types.AttributeValue
    var writeReqs []types.WriteRequest
    for i := 1; i < 4; i++ {
        item, err = attributevalue.MarshalMap(User{Username: fmt.Sprintf("test_user_%v",
i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
        if err != nil {
            log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n", err)
            return err
        }
        writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
&types.PutRequest{Item: item}})
    }
}

```

```

}
_, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
    RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
})
if err != nil {
    log.Printf("Couldn't populate table %v with users. Here's why: %v\n", tableName,
err)
}
return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
error) {
    var userList UserList
    output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
        TableName: aws.String(tableName),
    })
    if err != nil {
        log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName, err)
    } else {
        err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
        if err != nil {
            log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
        }
    }
    return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user User)
error {
    userItem, err := attributevalue.MarshalMap(user)
    if err != nil {
        log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
    }
    _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
        Item:      userItem,
        TableName: aws.String(tableName),
    })
    if err != nil {
        log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
    }
    return err
}

```

```
}
```

Crea una estructura que agrupe las acciones de CloudWatch Logs.

```
import (
    "context"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
    "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

type CloudWatchLogsActions struct {
    CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
    functionName string) (types.LogStream, error) {
    var logStream types.LogStream
    logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
    output, err := actor.CwlClient.DescribeLogStreams(ctx,
        &cloudwatchlogs.DescribeLogStreamsInput{
            Descending:  aws.Bool(true),
            Limit:       aws.Int32(1),
            LogGroupName: aws.String(logGroupName),
            OrderBy:     types.OrderByLastEventTime,
        })
    if err != nil {
        log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
            logGroupName, err)
    } else {
        logStream = output.LogStreams[0]
    }
    return logStream, err
}
```

```
// GetLogEvents gets the most recent eventCount events from the specified log
// stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
string, logStreamName string, eventCount int32) (
[]types.OutputLogEvent, error) {
var events []types.OutputLogEvent
logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
output, err := actor.CwlClient.GetLogEvents(ctx, &cloudwatchlogs.GetLogEventsInput{
    LogStreamName: aws.String(logStreamName),
    Limit:         aws.Int32(eventCount),
    LogGroupName:  aws.String(logGroupName),
})
if err != nil {
    log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
logStreamName, err)
} else {
    events = output.Events
}
return events, err
}
```

Crea una estructura que agrupe las acciones. CloudFormation

```
import (
    "context"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
    CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
// structured format.
```

```
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName string)
StackOutputs {
    output, err := actor.CfnClient.DescribeStacks(ctx,
        &cloudformation.DescribeStacksInput{
            StackName: aws.String(stackName),
        })
    if err != nil || len(output.Stacks) == 0 {
        log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
            stackName, err)
    }
    stackOutputs := StackOutputs{}
    for _, out := range output.Stacks[0].Outputs {
        stackOutputs[*out.OutputKey] = *out.OutputValue
    }
    return stackOutputs
}
```

Eliminación de recursos.

```
import (
    "context"
    "log"
    "user_pools_and_lambda_triggers/actions"

    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
    userPoolId      string
    userAccessTokens []string
    triggers        []actions.Trigger

    cognitoActor *actions.CognitoActions
    questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
    demotools.IQuestioner) {
```

```

resources.userAccessTokens = []string{}
resources.triggers = []actions.Trigger{}
resources.cognitoActor = cognitoActor
resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
defer func() {
    if r := recover(); r != nil {
        log.Printf("Something went wrong during cleanup.\n%v\n", r)
        log.Println("Use the AWS Management Console to remove any remaining resources \n"
+
        "that were created for this scenario.")
    }
}()

wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
resources that were created "+
"during this demo (y/n)?", "y")
if wantDelete {
    for _, accessToken := range resources.userAccessTokens {
        err := resources.cognitoActor.DeleteUser(ctx, accessToken)
        if err != nil {
            log.Println("Couldn't delete user during cleanup.")
            panic(err)
        }
        log.Println("Deleted user.")
    }
    triggerList := make([]actions.TriggerInfo, len(resources.triggers))
    for i := 0; i < len(resources.triggers); i++ {
        triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i], HandlerArn:
nil}
    }
    err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
triggerList...)
    if err != nil {
        log.Println("Couldn't update Cognito triggers during cleanup.")
        panic(err)
    }
    log.Println("Removed Cognito triggers from user pool.")
} else {
    log.Println("Be sure to remove resources when you're done with them to avoid
unexpected charges!")
}

```

```
}  
}
```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Go .
 - [DeleteUser](#)
 - [InitiateAuth](#)
 - [SignUp](#)
 - [UpdateUserPool](#)

Migración en forma automática los usuarios conocidos con una función de Lambda

En el siguiente ejemplo de código, se muestra cómo migrar de manera automática los usuarios conocidos de Amazon Cognito con una función de Lambda.

- Configure un grupo de usuarios para que llame a una función de Lambda para el desencadenador `MigrateUser`.
- Inicie sesión en Amazon Cognito con un nombre de usuario y un correo electrónico que no estén en el grupo de usuarios.
- La función de Lambda escanea una tabla de DynamoDB y migra de manera automática los usuarios conocidos al grupo de usuarios.
- Realice el flujo en caso de olvido de contraseña para restablecer la contraseña respecto del usuario migrado.
- Inicie sesión como un nuevo usuario y, a continuación, elimine los recursos.

SDK para Go V2

 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](#).

Ejecutar un escenario interactivo en un símbolo del sistema.


```

import (
    "context"
    "errors"
    "fmt"
    "log"
    "strings"
    "user_pools_and_lambda_triggers/actions"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// MigrateUser separates the steps of this scenario into individual functions so
// that
// they are simpler to read and understand.
type MigrateUser struct {
    helper      IScenarioHelper
    questioner  demotools.IQuestioner
    resources   Resources
    cognitoActor *actions.CognitoActions
}

// NewMigrateUser constructs a new migrate user runner.
func NewMigrateUser(sdkConfig aws.Config, questioner demotools.IQuestioner, helper
    IScenarioHelper) MigrateUser {
    scenario := MigrateUser{
        helper:      helper,
        questioner:  questioner,
        resources:   Resources{},
        cognitoActor: &actions.CognitoActions{CognitoClient:
            cognitoidentityprovider.NewFromConfig(sdkConfig)},
    }
    scenario.resources.init(scenario.cognitoActor, questioner)
    return scenario
}

// AddMigrateUserTrigger adds a Lambda handler as an invocation target for the
// MigrateUser trigger.
func (runner *MigrateUser) AddMigrateUserTrigger(ctx context.Context, userPoolId
    string, functionArn string) {

```

```

log.Printf("Let's add a Lambda function to handle the MigrateUser trigger from
Cognito.\n" +
    "This trigger happens when an unknown user signs in, and lets your function take
action before Cognito\n" +
    "rejects the user.\n\n")
err := runner.cognitoActor.UpdateTriggers(
    ctx, userPoolId,
    actions.TriggerInfo{Trigger: actions.UserMigration, HandlerArn:
aws.String(functionArn)})
if err != nil {
    panic(err)
}
log.Printf("Lambda function %v added to user pool %v to handle the MigrateUser
trigger.\n",
    functionArn, userPoolId)

log.Println(strings.Repeat("-", 88))
}

// SignInUser adds a new user to the known users table and signs that user in to
Amazon Cognito.
func (runner *MigrateUser) SignInUser(ctx context.Context, usersTable string,
clientId string) (bool, actions.User) {
    log.Println("Let's sign in a user to your Cognito user pool. When the username and
email matches an entry in the\n" +
        "DynamoDB known users table, the email is automatically verified and the user is
migrated to the Cognito user pool.")

    user := actions.User{}
    user.UserName = runner.questioner.Ask("\nEnter a username:")
    user.UserEmail = runner.questioner.Ask("\nEnter an email that you own. This email
will be used to confirm user migration\n" +
        "during this example:")

    runner.helper.AddKnownUser(ctx, usersTable, user)

    var err error
    var resetRequired *types.PasswordResetRequiredException
    var authResult *types.AuthenticationResultType
    signedIn := false
    for !signedIn && resetRequired == nil {
        log.Printf("Signing in to Cognito as user '%v'. The expected result is a
PasswordResetRequiredException.\n\n", user.UserName)
        authResult, err = runner.cognitoActor.SignIn(ctx, clientId, user.UserName, "_")
    }

```

```

    if err != nil {
        if errors.As(err, &resetRequired) {
            log.Printf("\nUser '%v' is not in the Cognito user pool but was found in the
DynamoDB known users table.\n"+
                "User migration is started and a password reset is required.", user.UserName)
        } else {
            panic(err)
        }
    } else {
        log.Printf("User '%v' successfully signed in. This is unexpected and probably
means you have not\n"+
            "cleaned up a previous run of this scenario, so the user exist in the Cognito
user pool.\n"+
            "You can continue this example and select to clean up resources, or manually
remove\n"+
            "the user from your user pool and try again.", user.UserName)
        runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
*authResult.AccessToken)
        signedIn = true
    }
}

log.Println(strings.Repeat("-", 88))
return resetRequired != nil, user
}

// ResetPassword starts a password recovery flow.
func (runner *MigrateUser) ResetPassword(ctx context.Context, clientId string, user
actions.User) {
    wantCode := runner.questioner.AskBool(fmt.Sprintf("In order to migrate the user to
Cognito, you must be able to receive a confirmation\n"+
        "code by email at %v. Do you want to send a code (y/n)?", user.UserEmail), "y")
    if !wantCode {
        log.Println("To complete this example and successfully migrate a user to Cognito,
you must enter an email\n" +
            "you own that can receive a confirmation code.")
        return
    }
    codeDelivery, err := runner.cognitoActor.ForgotPassword(ctx, clientId,
user.UserName)
    if err != nil {
        panic(err)
    }
    log.Printf("\nA confirmation code has been sent to %v.", *codeDelivery.Destination)

```

```

code := runner.questioner.Ask("Check your email and enter it here:")

confirmed := false
password := runner.questioner.AskPassword("\nEnter a password that has at least
eight characters, uppercase, lowercase, numbers and symbols.\n"+
"(the password will not display as you type):", 8)
for !confirmed {
    log.Printf("\nConfirming password reset for user '%v'.\n", user.UserName)
    err = runner.cognitoActor.ConfirmForgotPassword(ctx, clientId, code,
user.UserName, password)
    if err != nil {
        var invalidPassword *types.InvalidPasswordException
        if errors.As(err, &invalidPassword) {
            password = runner.questioner.AskPassword("\nEnter another password:", 8)
        } else {
            panic(err)
        }
    } else {
        confirmed = true
    }
}
log.Printf("User '%v' successfully confirmed and migrated.\n", user.UserName)
log.Println("Signing in with your username and password...")
authResult, err := runner.cognitoActor.SignIn(ctx, clientId, user.UserName,
password)
if err != nil {
    panic(err)
}
log.Printf("Successfully signed in. Your access token starts with: %v...\n",
(*authResult.AccessToken)[:10])
runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
*authResult.AccessToken)

log.Println(strings.Repeat("-", 88))
}

// Run runs the scenario.
func (runner *MigrateUser) Run(ctx context.Context, stackName string) {
    defer func() {
        if r := recover(); r != nil {
            log.Println("Something went wrong with the demo.")
            runner.resources.Cleanup(ctx)
        }
    }()
}

```

```

log.Println(strings.Repeat("-", 88))
log.Printf("Welcome\n")

log.Println(strings.Repeat("-", 88))

stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
if err != nil {
    panic(err)
}
runner.resources.userPoolId = stackOutputs["UserPoolId"]

runner.AddMigrateUserTrigger(ctx, stackOutputs["UserPoolId"],
stackOutputs["MigrateUserFunctionArn"])
runner.resources.triggers = append(runner.resources.triggers,
actions.UserMigration)
resetNeeded, user := runner.SignInUser(ctx, stackOutputs["TableName"],
stackOutputs["UserPoolClientId"])
if resetNeeded {
    runner.helper.ListRecentLogEvents(ctx, stackOutputs["MigrateUserFunction"])
    runner.ResetPassword(ctx, stackOutputs["UserPoolClientId"], user)
}

runner.resources.Cleanup(ctx)

log.Println(strings.Repeat("-", 88))
log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}

```

Controle el desencadenador MigrateUser con una función de Lambda.

```

import (
    "context"
    "log"
    "os"

    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
    "github.com/aws/aws-sdk-go-v2/aws"

```

```

"github.com/aws/aws-sdk-go-v2/config"
"github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
"github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
"github.com/aws/aws-sdk-go-v2/service/dynamodb"
)

const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
// format.
type UserInfo struct {
    Username string `dynamodbav:"Username"`
    Email string `dynamodbav:"Email"`
}

type handler struct {
    dynamoClient *dynamodb.Client
}

// HandleRequest handles the MigrateUser event by looking up a user in an Amazon
// DynamoDB table and
// specifying whether they should be migrated to the user pool.
func (h *handler) HandleRequest(ctx context.Context, event
events.CognitoEventUserPoolsMigrateUser) (events.CognitoEventUserPoolsMigrateUser,
error) {
    log.Printf("Received migrate trigger from %v for user '%v'", event.TriggerSource,
event.Username)
    if event.TriggerSource != "UserMigration_Authentication" {
        return event, nil
    }
    tableName := os.Getenv(TABLE_NAME)
    user := UserInfo{
        Username: event.Username,
    }
    log.Printf("Looking up user '%v' in table %v.\n", user.Username, tableName)
    filterEx := expression.Name("Username").Equal(expression.Value(user.Username))
    expr, err := expression.NewBuilder().WithFilter(filterEx).Build()
    if err != nil {
        log.Printf("Error building expression to query for user '%v'.\n", user.Username)
        return event, err
    }
    output, err := h.dynamoClient.Scan(ctx, &dynamodb.ScanInput{
        TableName:          aws.String(tableName),
        FilterExpression:    expr.Filter(),
    })
}

```

```

    ExpressionAttributeNames: expr.Names(),
    ExpressionAttributeValues: expr.Values(),
  })
  if err != nil {
    log.Printf("Error looking up user '%v'.\n", user.UserName)
    return event, err
  }
  if len(output.Items) == 0 {
    log.Printf("User '%v' not found, not migrating user.\n", user.UserName)
    return event, err
  }

  var users []UserInfo
  err = attributevalue.UnmarshalListOfMaps(output.Items, &users)
  if err != nil {
    log.Printf("Couldn't unmarshal DynamoDB items. Here's why: %v\n", err)
    return event, err
  }

  user = users[0]
  log.Printf("UserName '%v' found with email %v. User is migrated and must reset password.\n", user.UserName, user.UserEmail)
  event.CognitoEventUserPoolsMigrateUserResponse.UserAttributes = map[string]string{
    "email":          user.UserEmail,
    "email_verified": "true", // email_verified is required for the forgot password flow.
  }
  event.CognitoEventUserPoolsMigrateUserResponse.FinalUserStatus = "RESET_REQUIRED"
  event.CognitoEventUserPoolsMigrateUserResponse.MessageAction = "SUPPRESS"

  return event, err
}

func main() {
  ctx := context.Background()
  sdkConfig, err := config.LoadDefaultConfig(ctx)
  if err != nil {
    log.Panicln(err)
  }
  h := handler{
    dynamoClient: dynamodb.NewFromConfig(sdkConfig),
  }
  lambda.Start(h.HandleRequest)
}

```

Cree una estructura que lleve a cabo las tareas habituales.

```
import (
    "context"
    "log"
    "strings"
    "time"
    "user_pools_and_lambda_triggers/actions"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cloudformation"
    "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this example.
type IScenarioHelper interface {
    Pause(secs int)
    GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
        error)
    PopulateUserTable(ctx context.Context, tableName string)
    GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
    AddKnownUser(ctx context.Context, tableName string, user actions.User)
    ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
// example.
type ScenarioHelper struct {
    questioner demotools.IQuestioner
    dynamoActor *actions.DynamoActions
    cfnActor     *actions.CloudFormationActions
    cwActor      *actions.CloudWatchLogsActions
    isTestRun    bool
}

// NewScenarioHelper constructs a new scenario helper.
```



```

func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
ScenarioHelper {
    scenario := ScenarioHelper{
        questioner: questioner,
        dynamoActor: &actions.DynamoActions{DynamoClient:
dynamodb.NewFromConfig(sdkConfig)},
        cfnActor: &actions.CloudFormationActions{CfnClient:
cloudformation.NewFromConfig(sdkConfig)},
        cwlActor: &actions.CloudWatchLogsActions{CwlClient:
cloudwatchlogs.NewFromConfig(sdkConfig)},
    }
    return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
    if !helper.isTestRun {
        time.Sleep(time.Duration(secs) * time.Second)
    }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName string)
(actions.StackOutputs, error) {
    return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
string) {
    log.Printf("First, let's add some users to the DynamoDB %v table we'll use for this
example.\n", tableName)
    err := helper.dynamoActor.PopulateTable(ctx, tableName)
    if err != nil {
        panic(err)
    }
}

// GetKnownUsers gets the users from the known users table in a structured format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
(actions.UserList, error) {
    knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
    if err != nil {

```

```

    log.Printf("Couldn't get known users from table %v. Here's why: %v\n", tableName,
err)
}
return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
user actions.User) {
    log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users table...
\n",
        user.UserName, user.UserEmail)
    err := helper.dynamoActor.AddUser(ctx, tableName, user)
    if err != nil {
        panic(err)
    }
}

// ListRecentLogEvents gets the most recent log stream and events for the specified
Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context, functionName
string) {
    log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
    helper.Pause(10)
    log.Println("Okay, let's check the logs to find what's happened recently with your
Lambda function.")
    logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
    if err != nil {
        panic(err)
    }
    log.Printf("Getting some recent events from log stream %v\n",
*logStream.LogStreamName)
    events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
*logStream.LogStreamName, 10)
    if err != nil {
        panic(err)
    }
    for _, event := range events {
        log.Printf("\t%v", *event.Message)
    }
    log.Println(strings.Repeat("-", 88))
}

```

Cree una estructura que ajuste las acciones de Amazon Cognito.

```
import (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
    CognitoClient *cognitoidentityprovider.Client
}

// Trigger and TriggerInfo define typed data for updating an Amazon Cognito trigger.
type Trigger int

const (
    PreSignUp Trigger = iota
    UserMigration
    PostAuthentication
)

type TriggerInfo struct {
    Trigger    Trigger
    HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger is
// specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId string,
    triggers ...TriggerInfo) error {
    output, err := actor.CognitoClient.DescribeUserPool(ctx,
        &cognitoidentityprovider.DescribeUserPoolInput{
            UserPoolId: aws.String(userPoolId),
```

```

    })
    if err != nil {
        log.Printf("Couldn't get info about user pool %v. Here's why: %v\n", userPoolId,
            err)
        return err
    }
    lambdaConfig := output.UserPool.LambdaConfig
    for _, trigger := range triggers {
        switch trigger.Trigger {
        case PreSignUp:
            lambdaConfig.PreSignUp = trigger.HandlerArn
        case UserMigration:
            lambdaConfig.UserMigration = trigger.HandlerArn
        case PostAuthentication:
            lambdaConfig.PostAuthentication = trigger.HandlerArn
        }
    }
    _, err = actor.CognitoClient.UpdateUserPool(ctx,
        &cognitoidentityprovider.UpdateUserPoolInput{
            UserPoolId:    aws.String(userPoolId),
            LambdaConfig: lambdaConfig,
        })
    if err != nil {
        log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
    }
    return err
}

// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
    string, password string, userEmail string) (bool, error) {
    confirmed := false
    output, err := actor.CognitoClient.SignUp(ctx,
        &cognitoidentityprovider.SignUpInput{
            ClientId: aws.String(clientId),
            Password: aws.String(password),
            Username: aws.String(userName),
            UserAttributes: []types.AttributeType{
                {Name: aws.String("email"), Value: aws.String(userEmail)},
            },
        })
    if err != nil {

```

```

    var invalidPassword *types.InvalidPasswordException
    if errors.As(err, &invalidPassword) {
        log.Println(*invalidPassword.Message)
    } else {
        log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
    }
} else {
    confirmed = output.UserConfirmed
}
return confirmed, err
}

// SignIn signs in a user to Amazon Cognito using a username and password
// authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
string, password string) (*types.AuthenticationResultType, error) {
    var authResult *types.AuthenticationResultType
    output, err := actor.CognitoClient.InitiateAuth(ctx,
        &cognitoidentityprovider.InitiateAuthInput{
            AuthFlow:      "USER_PASSWORD_AUTH",
            ClientId:      aws.String(clientId),
            AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
        })
    if err != nil {
        var resetRequired *types.PasswordResetRequiredException
        if errors.As(err, &resetRequired) {
            log.Println(*resetRequired.Message)
        } else {
            log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
        }
    } else {
        authResult = output.AuthenticationResult
    }
    return authResult, err
}

// ForgotPassword starts a password recovery flow for a user. This flow typically
// sends a confirmation code
// to the user's configured notification destination, such as email.

```

```

func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
    userName string) (*types.CodeDeliveryDetailsType, error) {
    output, err := actor.CognitoClient.ForgotPassword(ctx,
        &cognitoidentityprovider.ForgotPasswordInput{
            ClientId: aws.String(clientId),
            Username: aws.String(userName),
        })
    if err != nil {
        log.Printf("Couldn't start password reset for user '%v'. Here's why: %v\n",
            userName, err)
    }
    return output.CodeDeliveryDetails, err
}

// ConfirmForgotPassword confirms a user with a confirmation code and a new
// password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
    string, code string, userName string, password string) error {
    _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
        &cognitoidentityprovider.ConfirmForgotPasswordInput{
            ClientId:      aws.String(clientId),
            ConfirmationCode: aws.String(code),
            Password:     aws.String(password),
            Username:     aws.String(userName),
        })
    if err != nil {
        var invalidPassword *types.InvalidPasswordException
        if errors.As(err, &invalidPassword) {
            log.Println(*invalidPassword.Message)
        } else {
            log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
        }
    }
    return err
}

// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken string)
    error {

```

```

_, err := actor.CognitoClient.DeleteUser(ctx,
&cognitoidentityprovider.DeleteUserInput{
    AccessToken: aws.String(userAccessToken),
})
if err != nil {
    log.Printf("Couldn't delete user. Here's why: %v\n", err)
}
return err
}

// AdminCreateUser uses administrator credentials to add a user to a user pool. This
// method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId string,
    userName string, userEmail string) error {
    _, err := actor.CognitoClient.AdminCreateUser(ctx,
    &cognitoidentityprovider.AdminCreateUserInput{
        UserPoolId:      aws.String(userPoolId),
        Username:        aws.String(userName),
        MessageAction:    types.MessageActionTypeSuppress,
        UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
aws.String(userEmail)}}},
    })
    if err != nil {
        var userExists *types.UsernameExistsException
        if errors.As(err, &userExists) {
            log.Printf("User %v already exists in the user pool.", userName)
            err = nil
        } else {
            log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
        }
    }
    return err
}

// AdminSetUserPassword uses administrator credentials to set a password for a user
// without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
    string, userName string, password string) error {

```

```

_, err := actor.CognitoClient.AdminSetUserPassword(ctx,
&cognitoidentityprovider.AdminSetUserPasswordInput{
    Password:    aws.String(password),
    UserPoolId:  aws.String(userPoolId),
    Username:    aws.String(userName),
    Permanent:   true,
})
if err != nil {
    var invalidPassword *types.InvalidPasswordException
    if errors.As(err, &invalidPassword) {
        log.Println(*invalidPassword.Message)
    } else {
        log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName, err)
    }
}
return err
}

```

Cree una estructura que ajuste las acciones de DynamoDB.

```

import (
    "context"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
// actions
// used in the examples.
type DynamoActions struct {
    DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {

```



```

    UserName string
    userEmail string
    LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
    UserPoolId string
    ClientId    string
    Time       string
}

// UserList defines a list of users.
type UserList struct {
    Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of strings.
func (users *UserList) UserNameList() []string {
    names := make([]string, len(users.Users))
    for i := 0; i < len(users.Users); i++ {
        names[i] = users.Users[i].UserName
    }
    return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
    error {
    var err error
    var item map[string]types.AttributeValue
    var writeReqs []types.WriteRequest
    for i := 1; i < 4; i++ {
        item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_%v",
        i), userEmail: fmt.Sprintf("test_email_%v@example.com", i)})
        if err != nil {
            log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n", err)
            return err
        }
        writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
        &types.PutRequest{Item: item}})
    }
    _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
        RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},

```

```

    })
    if err != nil {
        log.Printf("Couldn't populate table %v with users. Here's why: %v\n", tableName,
            err)
    }
    return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
    error) {
    var userList UserList
    output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
        TableName: aws.String(tableName),
    })
    if err != nil {
        log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName, err)
    } else {
        err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
        if err != nil {
            log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
        }
    }
    return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user User)
    error {
    userItem, err := attributevalue.MarshalMap(user)
    if err != nil {
        log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
    }
    _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
        Item:      userItem,
        TableName: aws.String(tableName),
    })
    if err != nil {
        log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
    }
    return err
}

```

Crea una estructura que agrupe las acciones de CloudWatch Logs.

```
import (
    "context"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
    "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

type CloudWatchLogsActions struct {
    CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
    functionName string) (types.LogStream, error) {
    var logStream types.LogStream
    logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
    output, err := actor.CwlClient.DescribeLogStreams(ctx,
        &cloudwatchlogs.DescribeLogStreamsInput{
            Descending:    aws.Bool(true),
            Limit:         aws.Int32(1),
            LogGroupName: aws.String(logGroupName),
            OrderBy:      types.OrderByLastEventTime,
        })
    if err != nil {
        log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
            logGroupName, err)
    } else {
        logStream = output.LogStreams[0]
    }
    return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
// stream.
```

```
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
    string, logStreamName string, eventCount int32) (
    []types.OutputLogEvent, error) {
    var events []types.OutputLogEvent
    logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
    output, err := actor.CwlClient.GetLogEvents(ctx, &cloudwatchlogs.GetLogEventsInput{
        LogStreamName: aws.String(logStreamName),
        Limit:          aws.Int32(eventCount),
        LogGroupName:   aws.String(logGroupName),
    })
    if err != nil {
        log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
            logStreamName, err)
    } else {
        events = output.Events
    }
    return events, err
}
```

Crea una estructura que agrupe las acciones. CloudFormation

```
import (
    "context"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
    CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
// structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName string)
    StackOutputs {
```

```

output, err := actor.CfnClient.DescribeStacks(ctx,
&cloudformation.DescribeStacksInput{
    StackName: aws.String(stackName),
})
if err != nil || len(output.Stacks) == 0 {
    log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
stackName, err)
}
stackOutputs := StackOutputs{}
for _, out := range output.Stacks[0].Outputs {
    stackOutputs[*out.OutputKey] = *out.OutputValue
}
return stackOutputs
}

```

Eliminación de recursos.

```

import (
    "context"
    "log"
    "user_pools_and_lambda_triggers/actions"

    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
    userPoolId      string
    userAccessTokens []string
    triggers        []actions.Trigger

    cognitoActor *actions.CognitoActions
    questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
demotools.IQuestioner) {
    resources.userAccessTokens = []string{}
    resources.triggers = []actions.Trigger{}
}

```

```

resources.cognitoActor = cognitoActor
resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
    defer func() {
        if r := recover(); r != nil {
            log.Printf("Something went wrong during cleanup.\n%v\n", r)
            log.Println("Use the AWS Management Console to remove any remaining resources \n"
+
            "that were created for this scenario.")
        }
    }()

    wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
resources that were created "+
    "during this demo (y/n)?", "y")
    if wantDelete {
        for _, accessToken := range resources.userAccessTokens {
            err := resources.cognitoActor.DeleteUser(ctx, accessToken)
            if err != nil {
                log.Println("Couldn't delete user during cleanup.")
                panic(err)
            }
            log.Println("Deleted user.")
        }
        triggerList := make([]actions.TriggerInfo, len(resources.triggers))
        for i := 0; i < len(resources.triggers); i++ {
            triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i], HandlerArn:
nil}
        }
        err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
triggerList...)
        if err != nil {
            log.Println("Couldn't update Cognito triggers during cleanup.")
            panic(err)
        }
        log.Println("Removed Cognito triggers from user pool.")
    } else {
        log.Println("Be sure to remove resources when you're done with them to avoid
unexpected charges!")
    }
}

```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Go .
 - [ConfirmForgotPassword](#)
 - [DeleteUser](#)
 - [ForgotPassword](#)
 - [InitiateAuth](#)
 - [SignUp](#)
 - [UpdateUserPool](#)

Escriba datos de actividad personalizados con una función de Lambda tras la autenticación de usuario de Amazon Cognito

En el siguiente ejemplo de código, se muestra cómo escribir datos de actividad personalizados con una función de Lambda tras la autenticación de usuarios de Amazon Cognito.

- Utilice las funciones de administrador para añadir un usuario a un grupo de usuarios.
- Configure un grupo de usuarios para que llame a una función de Lambda para el desencadenador `PostAuthentication`.
- Inicie sesión con el nuevo usuario en Amazon Cognito.
- La función Lambda escribe información personalizada en los CloudWatch registros y en una tabla de DynamoDB.
- Obtenga y exhiba los datos personalizados de la tabla de DynamoDB y, a continuación, elimine los recursos.

SDK para Go V2

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](#).

Ejecutar un escenario interactivo en un símbolo del sistema.

```

import (
    "context"
    "errors"
    "log"
    "strings"
    "user_pools_and_lambda_triggers/actions"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// ActivityLog separates the steps of this scenario into individual functions so
// that
// they are simpler to read and understand.
type ActivityLog struct {
    helper      IScenarioHelper
    questioner  demotools.IQuestioner
    resources   Resources
    cognitoActor *actions.CognitoActions
}

// NewActivityLog constructs a new activity log runner.
func NewActivityLog(sdkConfig aws.Config, questioner demotools.IQuestioner, helper
    IScenarioHelper) ActivityLog {
    scenario := ActivityLog{
        helper:      helper,
        questioner:  questioner,
        resources:   Resources{},
        cognitoActor: &actions.CognitoActions{CognitoClient:
            cognitoidentityprovider.NewFromConfig(sdkConfig)},
    }
    scenario.resources.init(scenario.cognitoActor, questioner)
    return scenario
}

// AddUserToPool selects a user from the known users table and uses administrator
// credentials to add the user to the user pool.
func (runner *ActivityLog) AddUserToPool(ctx context.Context, userPoolId string,
    tableName string) (string, string) {

```



```

log.Println("To facilitate this example, let's add a user to the user pool using
administrator privileges.")
users, err := runner.helper.GetKnownUsers(ctx, tableName)
if err != nil {
    panic(err)
}
user := users.Users[0]
log.Printf("Adding known user %v to the user pool.\n", user.UserName)
err = runner.cognitoActor.AdminCreateUser(ctx, userPoolId, user.UserName,
user.UserEmail)
if err != nil {
    panic(err)
}
pwSet := false
password := runner.questioner.AskPassword("\nEnter a password that has at least
eight characters, uppercase, lowercase, numbers and symbols.\n"+
"(the password will not display as you type):", 8)
for !pwSet {
    log.Printf("\nSetting password for user '%v'.\n", user.UserName)
    err = runner.cognitoActor.AdminSetUserPassword(ctx, userPoolId, user.UserName,
password)
    if err != nil {
        var invalidPassword *types.InvalidPasswordException
        if errors.As(err, &invalidPassword) {
            password = runner.questioner.AskPassword("\nEnter another password:", 8)
        } else {
            panic(err)
        }
    } else {
        pwSet = true
    }
}

log.Println(strings.Repeat("-", 88))

return user.UserName, password
}

// AddActivityLogTrigger adds a Lambda handler as an invocation target for the
PostAuthentication trigger.
func (runner *ActivityLog) AddActivityLogTrigger(ctx context.Context, userPoolId
string, activityLogArn string) {
    log.Println("Let's add a Lambda function to handle the PostAuthentication trigger
from Cognito.\n" +

```

```

    "This trigger happens after a user is authenticated, and lets your function take
    action, such as logging\n" +
    "the outcome.")
err := runner.cognitoActor.UpdateTriggers(
    ctx, userPoolId,
    actions.TriggerInfo{Trigger: actions.PostAuthentication, HandlerArn:
aws.String(activityLogArn)})
if err != nil {
    panic(err)
}
runner.resources.triggers = append(runner.resources.triggers,
actions.PostAuthentication)
log.Printf("Lambda function %v added to user pool %v to handle PostAuthentication
Cognito trigger.\n",
    activityLogArn, userPoolId)

log.Println(strings.Repeat("-", 88))
}

// SignInUser signs in as the specified user.
func (runner *ActivityLog) SignInUser(ctx context.Context, clientId string, userName
string, password string) {
    log.Printf("Now we'll sign in user %v and check the results in the logs and the
DynamoDB table.", userName)
    runner.questioner.Ask("Press Enter when you're ready.")
    authResult, err := runner.cognitoActor.SignIn(ctx, clientId, userName, password)
    if err != nil {
        panic(err)
    }
    log.Println("Sign in successful.",
        "The PostAuthentication Lambda handler writes custom information to CloudWatch
Logs.")

    runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
        *authResult.AccessToken)
}

// GetKnownUserLastLogin gets the login info for a user from the Amazon DynamoDB
table and displays it.
func (runner *ActivityLog) GetKnownUserLastLogin(ctx context.Context, tableName
string, userName string) {
    log.Println("The PostAuthentication handler also writes login data to the DynamoDB
table.")
    runner.questioner.Ask("Press Enter when you're ready to continue.")

```

```

users, err := runner.helper.GetKnownUsers(ctx, tableName)
if err != nil {
    panic(err)
}
for _, user := range users.Users {
    if user.UserName == userName {
        log.Println("The last login info for the user in the known users table is:")
        log.Printf("\t%+v", *user.LastLogin)
    }
}
log.Println(strings.Repeat("-", 88))
}

// Run runs the scenario.
func (runner *ActivityLog) Run(ctx context.Context, stackName string) {
    defer func() {
        if r := recover(); r != nil {
            log.Println("Something went wrong with the demo.")
            runner.resources.Cleanup(ctx)
        }
    }()

    log.Println(strings.Repeat("-", 88))
    log.Printf("Welcome\n")

    log.Println(strings.Repeat("-", 88))

    stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
    if err != nil {
        panic(err)
    }
    runner.resources.userPoolId = stackOutputs["UserPoolId"]
    runner.helper.PopulateUserTable(ctx, stackOutputs["TableName"])
    userName, password := runner.AddUserToPool(ctx, stackOutputs["UserPoolId"],
        stackOutputs["TableName"])

    runner.AddActivityLogTrigger(ctx, stackOutputs["UserPoolId"],
        stackOutputs["ActivityLogFunctionArn"])
    runner.SignInUser(ctx, stackOutputs["UserPoolClientId"], userName, password)
    runner.helper.ListRecentLogEvents(ctx, stackOutputs["ActivityLogFunction"])
    runner.GetKnownUserLastLogin(ctx, stackOutputs["TableName"], userName)

    runner.resources.Cleanup(ctx)
}

```

```

log.Println(strings.Repeat("-", 88))
log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}

```

Controle el desencadenador PostAuthentication con una función de Lambda.

```

import (
    "context"
    "fmt"
    "log"
    "os"
    "time"

    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
    dynamodbtypes "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

const TABLE_NAME = "TABLE_NAME"

// LoginInfo defines structured login data that can be marshalled to a DynamoDB
// format.
type LoginInfo struct {
    UserPoolId string `dynamodbav:"UserPoolId"`
    ClientId   string `dynamodbav:"ClientId"`
    Time      string `dynamodbav:"Time"`
}

// UserInfo defines structured user data that can be marshalled to a DynamoDB
// format.
type UserInfo struct {
    UserName   string `dynamodbav:"UserName"`
    UserEmail  string `dynamodbav:"UserEmail"`
    LastLogin  LoginInfo `dynamodbav:"LastLogin"`
}

```

```
// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
    userEmail, err := attributevalue.Marshal(user.UserEmail)
    if err != nil {
        panic(err)
    }
    return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
    dynamoClient *dynamodb.Client
}

// HandleRequest handles the PostAuthentication event by writing custom data to the
// logs and
// to an Amazon DynamoDB table.
func (h *handler) HandleRequest(ctx context.Context,
    event events.CognitoEventUserPoolsPostAuthentication)
(event.CognitoEventUserPoolsPostAuthentication, error) {
    log.Printf("Received post authentication trigger from %v for user '%v'",
        event.TriggerSource, event.UserName)
    tableName := os.Getenv(TABLE_NAME)
    user := UserInfo{
        UserName: event.UserName,
        UserEmail: event.Request.UserAttributes["email"],
        LastLogin: LoginInfo{
            UserPoolId: event.UserPoolID,
            ClientId: event.CallerContext.ClientID,
            Time: time.Now().Format(time.UnixDate),
        },
    }
    // Write to CloudWatch Logs.
    fmt.Printf("%#v", user)

    // Also write to an external system. This examples uses DynamoDB to demonstrate.
    userMap, err := attributevalue.MarshalMap(user)
    if err != nil {
        log.Printf("Couldn't marshal to DynamoDB map. Here's why: %v\n", err)
    } else if len(userMap) == 0 {
        log.Printf("User info marshaled to an empty map.")
    } else {
        _, err := h.dynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
            Item: userMap,
        })
    }
}
```

```

    TableName: aws.String(tableName),
  })
  if err != nil {
    log.Printf("Couldn't write to DynamoDB. Here's why: %v\n", err)
  } else {
    log.Printf("Wrote user info to DynamoDB table %v.\n", tableName)
  }
}

return event, nil
}

func main() {
  ctx := context.Background()
  sdkConfig, err := config.LoadDefaultConfig(ctx)
  if err != nil {
    log.Panicln(err)
  }
  h := handler{
    dynamoClient: dynamodb.NewFromConfig(sdkConfig),
  }
  lambda.Start(h.HandleRequest)
}

```

Cree una estructura que lleve a cabo las tareas habituales.

```

import (
  "context"
  "log"
  "strings"
  "time"
  "user_pools_and_lambda_triggers/actions"

  "github.com/aws/aws-sdk-go-v2/aws"
  "github.com/aws/aws-sdk-go-v2/service/cloudformation"
  "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
  "github.com/aws/aws-sdk-go-v2/service/dynamodb"
  "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

```

```
// IScenarioHelper defines common functions used by the workflows in this example.
type IScenarioHelper interface {
    Pause(secs int)
    GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
        error)
    PopulateUserTable(ctx context.Context, tableName string)
    GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
    AddKnownUser(ctx context.Context, tableName string, user actions.User)
    ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
// example.
type ScenarioHelper struct {
    questioner demotools.IQuestioner
    dynamoActor *actions.DynamoActions
    cfnActor     *actions.CloudFormationActions
    cwlActor     *actions.CloudWatchLogsActions
    isTestRun    bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
    ScenarioHelper {
    scenario := ScenarioHelper{
        questioner: questioner,
        dynamoActor: &actions.DynamoActions{DynamoClient:
            dynamodb.NewFromConfig(sdkConfig)},
        cfnActor:     &actions.CloudFormationActions{CfnClient:
            cloudformation.NewFromConfig(sdkConfig)},
        cwlActor:     &actions.CloudWatchLogsActions{CwlClient:
            cloudwatchlogs.NewFromConfig(sdkConfig)},
    }
    return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
    if !helper.isTestRun {
        time.Sleep(time.Duration(secs) * time.Second)
    }
}
```

```
// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
// structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName string)
(actions.StackOutputs, error) {
    return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
string) {
    log.Printf("First, let's add some users to the DynamoDB %v table we'll use for this
example.\n", tableName)
    err := helper.dynamoActor.PopulateTable(ctx, tableName)
    if err != nil {
        panic(err)
    }
}

// GetKnownUsers gets the users from the known users table in a structured format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
(actions.UserList, error) {
    knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
    if err != nil {
        log.Printf("Couldn't get known users from table %v. Here's why: %v\n", tableName,
err)
    }
    return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
user actions.User) {
    log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users table...
\n",
        user.UserName, user.UserEmail)
    err := helper.dynamoActor.AddUser(ctx, tableName, user)
    if err != nil {
        panic(err)
    }
}

// ListRecentLogEvents gets the most recent log stream and events for the specified
// Lambda function and displays them.
```



```

func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context, functionName
string) {
    log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
    helper.Pause(10)
    log.Println("Okay, let's check the logs to find what's happened recently with your
Lambda function.")
    logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
    if err != nil {
        panic(err)
    }
    log.Printf("Getting some recent events from log stream %v\n",
*logStream.LogStreamName)
    events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
*logStream.LogStreamName, 10)
    if err != nil {
        panic(err)
    }
    for _, event := range events {
        log.Printf("\t\t%v", *event.Message)
    }
    log.Println(strings.Repeat("-", 88))
}

```

Cree una estructura que ajuste las acciones de Amazon Cognito.

```

import (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
    "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
    CognitoClient *cognitoidentityprovider.Client
}

```

```
// Trigger and TriggerInfo define typed data for updating an Amazon Cognito trigger.
type Trigger int

const (
    PreSignUp Trigger = iota
    UserMigration
    PostAuthentication
)

type TriggerInfo struct {
    Trigger    Trigger
    HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger is
// specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId string,
    triggers ...TriggerInfo) error {
    output, err := actor.CognitoClient.DescribeUserPool(ctx,
        &cognitoidentityprovider.DescribeUserPoolInput{
            UserPoolId: aws.String(userPoolId),
        })
    if err != nil {
        log.Printf("Couldn't get info about user pool %v. Here's why: %v\n", userPoolId,
            err)
        return err
    }
    lambdaConfig := output.UserPool.LambdaConfig
    for _, trigger := range triggers {
        switch trigger.Trigger {
        case PreSignUp:
            lambdaConfig.PreSignUp = trigger.HandlerArn
        case UserMigration:
            lambdaConfig.UserMigration = trigger.HandlerArn
        case PostAuthentication:
            lambdaConfig.PostAuthentication = trigger.HandlerArn
        }
    }
    _, err = actor.CognitoClient.UpdateUserPool(ctx,
        &cognitoidentityprovider.UpdateUserPoolInput{
            UserPoolId:    aws.String(userPoolId),
            LambdaConfig: lambdaConfig,
        })
}
```

```

    })
    if err != nil {
        log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
    }
    return err
}

// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
string, password string, userEmail string) (bool, error) {
    confirmed := false
    output, err := actor.CognitoClient.SignUp(ctx,
    &cognitoidentityprovider.SignUpInput{
        ClientId: aws.String(clientId),
        Password: aws.String(password),
        Username: aws.String(userName),
        UserAttributes: []types.AttributeType{
            {Name: aws.String("email"), Value: aws.String(userEmail)},
        },
    })
    if err != nil {
        var invalidPassword *types.InvalidPasswordException
        if errors.As(err, &invalidPassword) {
            log.Println(*invalidPassword.Message)
        } else {
            log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
        }
    } else {
        confirmed = output.UserConfirmed
    }
    return confirmed, err
}

// SignIn signs in a user to Amazon Cognito using a username and password
authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
string, password string) (*types.AuthenticationResultType, error) {
    var authResult *types.AuthenticationResultType
    output, err := actor.CognitoClient.InitiateAuth(ctx,
    &cognitoidentityprovider.InitiateAuthInput{

```

```

    AuthFlow:      "USER_PASSWORD_AUTH",
    ClientId:      aws.String(clientId),
    AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
  })
  if err != nil {
    var resetRequired *types.PasswordResetRequiredException
    if errors.As(err, &resetRequired) {
      log.Println(*resetRequired.Message)
    } else {
      log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
    }
  } else {
    authResult = output.AuthenticationResult
  }
  return authResult, err
}

```

// ForgotPassword starts a password recovery flow for a user. This flow typically sends a confirmation code

// to the user's configured notification destination, such as email.

```

func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
  userName string) (*types.CodeDeliveryDetailsType, error) {
  output, err := actor.CognitoClient.ForgotPassword(ctx,
    &cognitoidentityprovider.ForgotPasswordInput{
      ClientId: aws.String(clientId),
      Username: aws.String(userName),
    })
  if err != nil {
    log.Printf("Couldn't start password reset for user '%v'. Here's why: %v\n",
      userName, err)
  }
  return output.CodeDeliveryDetails, err
}

```

// ConfirmForgotPassword confirms a user with a confirmation code and a new password.

```

func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
  string, code string, userName string, password string) error {
  _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
    &cognitoidentityprovider.ConfirmForgotPasswordInput{

```

```

    ClientId:      aws.String(clientId),
    ConfirmationCode: aws.String(code),
    Password:      aws.String(password),
    Username:      aws.String(userName),
  })
  if err != nil {
    var invalidPassword *types.InvalidPasswordException
    if errors.As(err, &invalidPassword) {
      log.Println(*invalidPassword.Message)
    } else {
      log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
    }
  }
  return err
}

// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken string)
  error {
  _, err := actor.CognitoClient.DeleteUser(ctx,
    &cognitoidentityprovider.DeleteUserInput{
      AccessToken: aws.String(userAccessToken),
    })
  if err != nil {
    log.Printf("Couldn't delete user. Here's why: %v\n", err)
  }
  return err
}

// AdminCreateUser uses administrator credentials to add a user to a user pool. This
  method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId string,
  userName string, userEmail string) error {
  _, err := actor.CognitoClient.AdminCreateUser(ctx,
    &cognitoidentityprovider.AdminCreateUserInput{
      UserPoolId:      aws.String(userPoolId),
      Username:        aws.String(userName),
      MessageAction:   types.MessageActionTypeSuppress,
    })
  if err != nil {
    log.Printf("Couldn't create user. Here's why: %v\n", err)
  }
  return err
}

```

```

    UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
aws.String(userEmail)}}},
})
if err != nil {
    var userExists *types.UsernameExistsException
    if errors.As(err, &userExists) {
        log.Printf("User %v already exists in the user pool.", userName)
        err = nil
    } else {
        log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
    }
}
return err
}

// AdminSetUserPassword uses administrator credentials to set a password for a user
// without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
string, userName string, password string) error {
    _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
    &cognitoidentityprovider.AdminSetUserPasswordInput{
        Password:    aws.String(password),
        UserPoolId:  aws.String(userPoolId),
        Username:    aws.String(userName),
        Permanent:   true,
    })
    if err != nil {
        var invalidPassword *types.InvalidPasswordException
        if errors.As(err, &invalidPassword) {
            log.Println(*invalidPassword.Message)
        } else {
            log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName, err)
        }
    }
    return err
}

```

Cree una estructura que ajuste las acciones de DynamoDB.

```
import (  
    "context"  
    "fmt"  
    "log"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"  
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"  
    "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"  
)  
  
// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)  
// actions  
// used in the examples.  
type DynamoActions struct {  
    DynamoClient *dynamodb.Client  
}  
  
// User defines structured user data.  
type User struct {  
    Username  string  
    UserEmail string  
    LastLogin *LoginInfo `dynamodbav:",omitempty"`  
}  
  
// LoginInfo defines structured custom login data.  
type LoginInfo struct {  
    UserPoolId string  
    ClientId   string  
    Time       string  
}  
  
// UserList defines a list of users.  
type UserList struct {  
    Users []User  
}  
  
// UserNameList returns the usernames contained in a UserList as a list of strings.  
func (users *UserList) UserNameList() []string {  
    names := make([]string, len(users.Users))  
    for i := 0; i < len(users.Users); i++ {  
        names[i] = users.Users[i].Username  
    }  
}
```

```

}
return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
error {
    var err error
    var item map[string]types.AttributeValue
    var writeReqs []types.WriteRequest
    for i := 1; i < 4; i++ {
        item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_%v",
i), userEmail: fmt.Sprintf("test_email_%v@example.com", i)})
        if err != nil {
            log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n", err)
            return err
        }
        writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
&types.PutRequest{Item: item}})
    }
    _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
})
    if err != nil {
        log.Printf("Couldn't populate table %v with users. Here's why: %v\n", tableName,
err)
    }
    return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
error) {
    var userList UserList
    output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
TableName: aws.String(tableName),
})
    if err != nil {
        log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName, err)
    } else {
        err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
        if err != nil {
            log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
        }
    }
}

```



```

}
return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user User)
error {
    userItem, err := attributevalue.MarshalMap(user)
    if err != nil {
        log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
    }
    _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
        Item:      userItem,
        TableName: aws.String(tableName),
    })
    if err != nil {
        log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
    }
    return err
}

```

Crea una estructura que agrupe las acciones de CloudWatch Logs.

```

import (
    "context"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
    "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

type CloudWatchLogsActions struct {
    CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
    functionName string) (types.LogStream, error) {

```

```

var logStream types.LogStream
logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
output, err := actor.CwlClient.DescribeLogStreams(ctx,
&cloudwatchlogs.DescribeLogStreamsInput{
    Descending:    aws.Bool(true),
    Limit:         aws.Int32(1),
    LogGroupName:  aws.String(logGroupName),
    OrderBy:       types.OrderByLastEventTime,
})
if err != nil {
    log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
logGroupName, err)
} else {
    logStream = output.LogStreams[0]
}
return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
string, logStreamName string, eventCount int32) (
[]types.OutputLogEvent, error) {
var events []types.OutputLogEvent
logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
output, err := actor.CwlClient.GetLogEvents(ctx, &cloudwatchlogs.GetLogEventsInput{
    LogStreamName: aws.String(logStreamName),
    Limit:         aws.Int32(eventCount),
    LogGroupName:  aws.String(logGroupName),
})
if err != nil {
    log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
logStreamName, err)
} else {
    events = output.Events
}
return events, err
}

```

Crea una estructura que agrupe las acciones. CloudFormation

```

import (
    "context"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
    CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
// structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName string)
    StackOutputs {
    output, err := actor.CfnClient.DescribeStacks(ctx,
        &cloudformation.DescribeStacksInput{
            StackName: aws.String(stackName),
        })
    if err != nil || len(output.Stacks) == 0 {
        log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
            stackName, err)
    }
    stackOutputs := StackOutputs{}
    for _, out := range output.Stacks[0].Outputs {
        stackOutputs[*out.OutputKey] = *out.OutputValue
    }
    return stackOutputs
}

```

Eliminación de recursos.

```

import (
    "context"
    "log"

```

```

"user_pools_and_lambda_triggers/actions"

"github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
    userPoolId      string
    userAccessTokens []string
    triggers        []actions.Trigger

    cognitoActor *actions.CognitoActions
    questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
demotools.IQuestioner) {
    resources.userAccessTokens = []string{}
    resources.triggers = []actions.Trigger{}
    resources.cognitoActor = cognitoActor
    resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
    defer func() {
        if r := recover(); r != nil {
            log.Printf("Something went wrong during cleanup.\n%v\n", r)
            log.Println("Use the AWS Management Console to remove any remaining resources \n"
+
            "that were created for this scenario.")
        }
    }()

    wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
resources that were created "+
    "during this demo (y/n)?", "y")
    if wantDelete {
        for _, accessToken := range resources.userAccessTokens {
            err := resources.cognitoActor.DeleteUser(ctx, accessToken)
            if err != nil {
                log.Println("Couldn't delete user during cleanup.")
                panic(err)
            }
        }
    }
}

```

```
    }
    log.Println("Deleted user.")
}
triggerList := make([]actions.TriggerInfo, len(resources.triggers))
for i := 0; i < len(resources.triggers); i++ {
    triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i], HandlerArn:
nil}
}
err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
triggerList...)
if err != nil {
    log.Println("Couldn't update Cognito triggers during cleanup.")
    panic(err)
}
log.Println("Removed Cognito triggers from user pool.")
} else {
    log.Println("Be sure to remove resources when you're done with them to avoid
unexpected charges!")
}
}
```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Go .
 - [AdminCreateUser](#)
 - [AdminSetUserPassword](#)
 - [DeleteUser](#)
 - [InitiateAuth](#)
 - [UpdateUserPool](#)

Ejemplos de tecnología sin servidor

Conexión a una base de datos de Amazon RDS en una función de Lambda

En el siguiente ejemplo de código, se muestra cómo implementar una función de Lambda que se conecta a una base de datos de RDS. La función realiza una solicitud sencilla a la base de datos y devuelve el resultado.

SDK para Go V2

 Note

Hay más en marcha. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Conexión a una base de datos de Amazon RDS en una función de Lambda mediante Go.

```
/*
Golang v2 code here.
*/

package main

import (
    "context"
    "database/sql"
    "encoding/json"
    "fmt"
    "os"

    "github.com/aws/aws-lambda-go/lambda"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/feature/rds/auth"
    _ "github.com/go-sql-driver/mysql"
)

type MyEvent struct {
    Name string `json:"name"`
}

func HandleRequest(event *MyEvent) (map[string]interface{}, error) {

    var dbName string = os.Getenv("DatabaseName")
    var dbUser string = os.Getenv("DatabaseUser")
    var dbHost string = os.Getenv("DBHost") // Add hostname without https
    var dbPort int = os.Getenv("Port")      // Add port number
    var dbEndpoint string = fmt.Sprintf("%s:%d", dbHost, dbPort)
    var region string = os.Getenv("AWS_REGION")

    cfg, err := config.LoadDefaultConfig(context.TODO())
```

```

if err != nil {
    panic("configuration error: " + err.Error())
}

authenticationToken, err := auth.BuildAuthToken(
    context.TODO(), dbEndpoint, region, dbUser, cfg.Credentials)
if err != nil {
    panic("failed to create authentication token: " + err.Error())
}

dsn := fmt.Sprintf("%s:%s@tcp(%s)/%s?tls=true&allowCleartextPasswords=true",
    dbUser, authenticationToken, dbEndpoint, dbName,
)

db, err := sql.Open("mysql", dsn)
if err != nil {
    panic(err)
}

defer db.Close()

var sum int
err = db.QueryRow("SELECT ?+? AS sum", 3, 2).Scan(&sum)
if err != nil {
    panic(err)
}
s := fmt.Sprint(sum)
message := fmt.Sprintf("The selected sum is: %s", s)

messageBytes, err := json.Marshal(message)
if err != nil {
    return nil, err
}

messageString := string(messageBytes)
return map[string]interface{}{
    "statusCode": 200,
    "headers":    map[string]string{"Content-Type": "application/json"},
    "body":       messageString,
}, nil
}

func main() {
    lambda.Start(HandleRequest)
}

```

```
}
```

Invocar una función de Lambda desde un desencadenador de Kinesis

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento activado al recibir registros de un flujo de Kinesis. La función recupera la carga útil de Kinesis, la decodifica desde Base64 y registra el contenido del registro.

SDK para Go V2

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Uso de un evento de Kinesis con Lambda mediante Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
    "context"
    "log"

    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, kinesisEvent events.KinesisEvent) error {
    if len(kinesisEvent.Records) == 0 {
        log.Printf("empty Kinesis event received")
        return nil
    }

    for _, record := range kinesisEvent.Records {
        log.Printf("processed Kinesis event with EventId: %v", record.EventID)
        recordDataBytes := record.Kinesis.Data
        recordDataText := string(recordDataBytes)
    }
}
```



```
    log.Printf("record data: %v", recordDataText)
    // TODO: Do interesting work based on the new data
}
log.Printf("successfully processed %v records", len(kinesisEvent.Records))
return nil
}

func main() {
    lambda.Start(handler)
}
```

Invocación de una función de Lambda desde un desencadenador de DynamoDB

El siguiente ejemplo de código muestra cómo implementar una función de Lambda que recibe un evento desencadenado al recibir registros de una secuencia de DynamoDB. La función recupera la carga útil de DynamoDB y registra el contenido del registro.

SDK para Go V2

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Consumo de un evento de DynamoDB con Lambda mediante Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
    "context"
    "github.com/aws/aws-lambda-go/lambda"
    "github.com/aws/aws-lambda-go/events"
    "fmt"
)

func HandleRequest(ctx context.Context, event events.DynamoDBEvent) (*string, error) {
    {
```

```
if len(event.Records) == 0 {
    return nil, fmt.Errorf("received empty event")
}

for _, record := range event.Records {
    LogDynamoDBRecord(record)
}

message := fmt.Sprintf("Records processed: %d", len(event.Records))
return &message, nil
}

func main() {
    lambda.Start(HandleRequest)
}

func LogDynamoDBRecord(record events.DynamoDBEventRecord){
    fmt.Println(record.EventID)
    fmt.Println(record.EventName)
    fmt.Printf("%+v\n", record.Change)
}
```

Invocación de una función de Lambda desde un desencadenador de Amazon DocumentDB

El siguiente ejemplo de código muestra cómo implementar una función de Lambda que recibe un evento desencadenado al recibir registros de una secuencia de cambios de DocumentDB. La función recupera la carga útil de DocumentDB y registra el contenido del registro.

SDK para Go V2

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Consumir un evento de Amazon DocumentDB con Lambda utilizando Go.

```
package main
```

```
import (  
    "context"  
    "encoding/json"  
    "fmt"  
  
    "github.com/aws/aws-lambda-go/lambda"  
)  
  
type Event struct {  
    Events []Record `json:"events"`  
}  
  
type Record struct {  
    Event struct {  
        OperationType string `json:"operationType"`  
        NS              struct {  
            DB      string `json:"db"`  
            Coll string `json:"coll"`  
        } `json:"ns"`  
        FullDocument interface{} `json:"fullDocument"`  
    } `json:"event"`  
}  
  
func main() {  
    lambda.Start(handler)  
}  
  
func handler(ctx context.Context, event Event) (string, error) {  
    fmt.Println("Loading function")  
    for _, record := range event.Events {  
        logDocumentDBEvent(record)  
    }  
  
    return "OK", nil  
}  
  
func logDocumentDBEvent(record Record) {  
    fmt.Printf("Operation type: %s\n", record.Event.OperationType)  
    fmt.Printf("db: %s\n", record.Event.NS.DB)  
    fmt.Printf("collection: %s\n", record.Event.NS.Coll)  
    docBytes, _ := json.MarshalIndent(record.Event.FullDocument, "", "  ")  
    fmt.Printf("Full document: %s\n", string(docBytes))  
}
```

Invocación de una función de Lambda desde un desencadenador de Amazon MSK

El siguientes ejemplo de código muestra cómo implementar una función de Lambda que recibe un evento desencadenado al recibir registros de un clúster de Amazon MSK. La función recupera la carga útil de MSK y registra el contenido del registro.

SDK para Go V2

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Consumo de un evento de Amazon MSK con Lambda mediante Go.

```
package main

import (
    "encoding/base64"
    "fmt"

    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
)

func handler(event events.KafkaEvent) {
    for key, records := range event.Records {
        fmt.Println("Key:", key)

        for _, record := range records {
            fmt.Println("Record:", record)

            decodedValue, _ := base64.StdEncoding.DecodeString(record.Value)
            message := string(decodedValue)
            fmt.Println("Message:", message)
        }
    }
}
```

```
}

func main() {
    lambda.Start(handler)
}
```

Invocación de una función de Lambda desde un desencadenador de Amazon S3

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento activado al cargar un objeto en un bucket de S3. La función recupera el nombre del bucket de S3 y la clave del objeto del parámetro de evento y llama a la API de Amazon S3 para recuperar y registrar el tipo de contenido del objeto.

SDK para Go V2

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos sin servidor](#).

Uso de un evento de S3 con Lambda mediante Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
    "context"
    "log"

    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"
)

func handler(ctx context.Context, s3Event events.S3Event) error {
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        log.Printf("failed to load default config: %s", err)
    }
}
```

```
    return err
}
s3Client := s3.NewFromConfig(sdkConfig)

for _, record := range s3Event.Records {
    bucket := record.S3.Bucket.Name
    key := record.S3.Object.URLDecodedKey
    headOutput, err := s3Client.HeadObject(ctx, &s3.HeadObjectInput{
        Bucket: &bucket,
        Key:     &key,
    })
    if err != nil {
        log.Printf("error getting head of object %s/%s: %s", bucket, key, err)
        return err
    }
    log.Printf("successfully retrieved %s/%s of type %s", bucket, key,
        *headOutput.ContentType)
}

return nil
}

func main() {
    lambda.Start(handler)
}
```

Invocación de una función de Lambda desde un desencadenador de Amazon SNS

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento activado al recibir mensajes de un tema de SNS. La función recupera los mensajes del parámetro de eventos y registra el contenido de cada mensaje.

SDK para Go V2

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Uso de un evento de SNS con Lambda mediante Go

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
    "context"
    "fmt"

    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, snsEvent events.SNSEvent) {
    for _, record := range snsEvent.Records {
        processMessage(record)
    }
    fmt.Println("done")
}

func processMessage(record events.SNSEventRecord) {
    message := record.SNS.Message
    fmt.Printf("Processed message: %s\n", message)
    // TODO: Process your record here
}

func main() {
    lambda.Start(handler)
}
```

Invocar una función de Lambda desde un desencadenador de Amazon SQS

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento activado al recibir mensajes de una cola de SQS. La función recupera los mensajes del parámetro de eventos y registra el contenido de cada mensaje.

SDK para Go V2

 Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Uso de un evento de SQS con Lambda mediante Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package integration_sqs_to_lambda

import (
    "fmt"
    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
)

func handler(event events.SQSEvent) error {
    for _, record := range event.Records {
        err := processMessage(record)
        if err != nil {
            return err
        }
    }
    fmt.Println("done")
    return nil
}

func processMessage(record events.SQSMessage) error {
    fmt.Printf("Processed message %s\n", record.Body)
    // TODO: Do interesting work based on the new message
    return nil
}

func main() {
    lambda.Start(handler)
}
```


Notificación de los errores de los elementos del lote de las funciones de Lambda mediante un desencadenador de Kinesis

En el siguiente ejemplo de código se muestra cómo implementar una respuesta por lotes parcial para funciones de Lambda que reciben eventos de un flujo de Kinesis. La función informa los errores de los elementos del lote en la respuesta y le indica a Lambda que vuelva a intentar esos mensajes más adelante.

SDK para Go V2

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Notificación de los errores de los elementos del lote de Kinesis con Lambda mediante Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
    "context"
    "fmt"
    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, kinesisEvent events.KinesisEvent)
(map[string]interface{}, error) {
    batchItemFailures := []map[string]interface{}{}

    for _, record := range kinesisEvent.Records {
        curRecordSequenceNumber := ""

        // Process your record
        if /* Your record processing condition here */ {
            curRecordSequenceNumber = record.Kinesis.SequenceNumber
        }

        // Add a condition to check if the record processing failed
    }
}
```

```

    if curRecordSequenceNumber != "" {
        batchItemFailures = append(batchItemFailures, map[string]interface{}{
            "itemIdentifier": curRecordSequenceNumber})
        }
    }

    kinesisBatchResponse := map[string]interface{}{
        "batchItemFailures": batchItemFailures,
    }
    return kinesisBatchResponse, nil
}

func main() {
    lambda.Start(handler)
}

```

Notificación de los errores de los elementos del lote de las funciones de Lambda con un desencadenador de DynamoDB

El siguiente ejemplo de código muestra cómo implementar una respuesta parcial por lotes para las funciones de Lambda que reciben eventos de una secuencia de DynamoDB. La función informa los errores de los elementos del lote en la respuesta y le indica a Lambda que vuelva a intentar esos mensajes más adelante.

SDK para Go V2

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Notificación de los errores de los elementos del lote de DynamoDB con Lambda mediante Go.

```

// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
    "context"

```

```
"github.com/aws/aws-lambda-go/events"
"github.com/aws/aws-lambda-go/lambda"
)

type BatchItemFailure struct {
    ItemIdentifier string `json:"ItemIdentifier"`
}

type BatchResult struct {
    BatchItemFailures []BatchItemFailure `json:"BatchItemFailures"`
}

func HandleRequest(ctx context.Context, event events.DynamoDBEvent) (*BatchResult,
error) {
    var batchItemFailures []BatchItemFailure
    curRecordSequenceNumber := ""

    for _, record := range event.Records {
        // Process your record
        curRecordSequenceNumber = record.Change.SequenceNumber
    }

    if curRecordSequenceNumber != "" {
        batchItemFailures = append(batchItemFailures, BatchItemFailure{ItemIdentifier:
curRecordSequenceNumber})
    }

    batchResult := BatchResult{
        BatchItemFailures: batchItemFailures,
    }

    return &batchResult, nil
}

func main() {
    lambda.Start(HandleRequest)
}
```

Notificación de los errores de los elementos del lote de las funciones de Lambda mediante un desencadenador de Amazon SQS.

En el siguiente ejemplo de código se muestra cómo implementar una respuesta por lotes parcial para funciones de Lambda que reciben eventos de una cola de SQS. La función informa los errores de los elementos del lote en la respuesta y le indica a Lambda que vuelva a intentar esos mensajes más adelante.

SDK para Go V2

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Notificación de los errores de los elementos del lote de SQS con Lambda mediante Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
    "context"
    "fmt"
    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, sqsEvent events.SQSEvent) (map[string]interface{},
    error) {
    batchItemFailures := []map[string]interface{}{}

    for _, message := range sqsEvent.Records {
        if len(message.Body) > 0 {
            // Your message processing condition here
            fmt.Printf("Successfully processed message: %s\n", message.Body)
        } else {
            // Message processing failed
            fmt.Printf("Failed to process message %s\n", message.MessageId)
            batchItemFailures = append(batchItemFailures, map[string]interface{}{
                "itemIdentifier": message.MessageId})
        }
    }

    return batchItemFailures, nil
}
```

```
    }  
  }  
  
  sqsBatchResponse := map[string]interface{}{  
    "batchItemFailures": batchItemFailures,  
  }  
  return sqsBatchResponse, nil  
}  
  
func main() {  
  lambda.Start(handler)  
}
```

AWS contribuciones de la comunidad

Cómo crear y probar una aplicación sin servidor

El siguiente ejemplo de código muestra cómo crear y probar una aplicación sin servidor mediante API Gateway con Lambda y DynamoDB.

SDK para Go V2

Muestra cómo crear y probar una aplicación sin servidor que consta de una API Gateway con Lambda y DynamoDB mediante el SDK de Go.

Para obtener el código fuente completo y las instrucciones sobre cómo configurarlo y ejecutarlo, consulte el ejemplo completo en [GitHub](#).

Servicios utilizados en este ejemplo

- API Gateway
- DynamoDB
- Lambda

Ejemplos de Amazon MSK con SDK para Go V2

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar situaciones comunes mediante el uso de la AWS SDK para Go V2 con Amazon MSK.

En cada ejemplo se incluye un enlace al código de origen completo, con instrucciones de configuración y ejecución del código en el contexto.

Temas

- [Ejemplos de tecnología sin servidor](#)

Ejemplos de tecnología sin servidor

Invocación de una función de Lambda desde un desencadenador de Amazon MSK

El siguientes ejemplo de código muestra cómo implementar una función de Lambda que recibe un evento desencadenado al recibir registros de un clúster de Amazon MSK. La función recupera la carga útil de MSK y registra el contenido del registro.

SDK para Go V2

Note

Hay más información al respecto. GitHub Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Consumo de un evento de Amazon MSK con Lambda mediante Go.

```
package main

import (
    "encoding/base64"
    "fmt"

    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
)

func handler(event events.KafkaEvent) {
    for key, records := range event.Records {
        fmt.Println("Key:", key)

        for _, record := range records {
```

```
    fmt.Println("Record:", record)

    decodedValue, _ := base64.StdEncoding.DecodeString(record.Value)
    message := string(decodedValue)
    fmt.Println("Message:", message)
}
}
}
```

```
func main() {
    lambda.Start(handler)
}
```

Ejemplos de Centro de socios con SDK para Go V2

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso de la AWS SDK para Go V2 con Partner Central.

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

GetOpportunity

En el siguiente ejemplo de código, se muestra cómo utilizar GetOpportunity.

SDK para Go V2

Obtener una oportunidad.

```

package main

import (
    "context"
    "encoding/json"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/partnercentral-selling"
)

func main() {
    config, err := config.LoadDefaultConfig(context.TODO())

    if err != nil {
        log.Fatal(err)
    }

    config.Region = "us-east-1"

    client := partnercentral-selling.NewFromConfig(config)

    output, err := client.GetOpportunity(context.TODO(),
        &partnercentral-selling.GetOpportunityInput{
            Identifier: aws.String("01111111"),
            Catalog:   aws.String("AWS"),
        })

    if err != nil {
        log.Fatal(err)
    }
    log.Println("printing opportunity...\n")

    jsonOutput, err := json.MarshalIndent(output, "", "    ")

    fmt.Println(string(jsonOutput))
}

```

- Para obtener más información sobre la API, consulte [GetOpportunity](#) la referencia AWS SDK para Go de la API.

ListOpportunities

En el siguiente ejemplo de código, se muestra cómo utilizar `ListOpportunities`.

SDK para Go V2

Mostrar las oportunidades.

```
package main

import (
    "context"
    "encoding/json"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/partnercentral-selling"
)

func main() {
    config, err := config.LoadDefaultConfig(context.TODO())

    if err != nil {
        log.Fatal(err)
    }

    config.Region = "us-east-1"

    client := partnercentral-selling.NewFromConfig(config)

    output, err := client.ListOpportunities(context.TODO(),
        &partnercentral-selling.ListOpportunitiesInput{
            MaxResults: aws.Int32(2),
            Catalog:   aws.String("AWS"),
        })

    if err != nil {
        log.Fatal(err)
    }

    jsonOutput, err := json.MarshalIndent(output, "", "    ")
}
```

```
fmt.Println(string(jsonOutput))
}
```

- Para obtener más información sobre la API, consulte [ListOpportunities](#) la Referencia AWS SDK para Go de la API.

Ejemplos de Amazon RDS usando SDK para Go V2

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso de la AWS SDK para Go V2 con 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.

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](#)
- [Ejemplos de tecnología sin servidor](#)

Introducción

Introducción a Amazon RDS

El siguiente ejemplo de código muestra cómo empezar a utilizar Amazon RDS.

SDK para Go V2

 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](#).

```
package main

import (
    "context"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/rds"
)

// main uses the AWS SDK for Go V2 to create an Amazon Relational Database Service
// (Amazon RDS)
// client and list up to 20 DB instances in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
        fmt.Println(err)
        return
    }
    rdsClient := rds.NewFromConfig(sdkConfig)
    const maxInstances = 20
    fmt.Printf("Let's list up to %v DB instances.\n", maxInstances)
    output, err := rdsClient.DescribeDBInstances(ctx,
        &rds.DescribeDBInstancesInput{MaxRecords: aws.Int32(maxInstances)})
    if err != nil {
        fmt.Printf("Couldn't list DB instances: %v\n", err)
        return
    }
}
```

```
}
if len(output.DBInstances) == 0 {
    fmt.Println("No DB instances found.")
} else {
    for _, instance := range output.DBInstances {
        fmt.Printf("DB instance %v has database %v.\n", *instance.DBInstanceIdentifier,
            *instance.DBName)
    }
}
}
```

- Para obtener más información sobre la API, consulta la [sección Describir DBInstances](#) en la referencia de la AWS SDK para Go 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 Go V2

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](#).

Ejecutar un escenario interactivo en un símbolo del sistema.

```

import (
    "context"
    "fmt"
    "log"
    "sort"
    "strconv"
    "strings"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/rds"
    "github.com/aws/aws-sdk-go-v2/service/rds/types"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/rds/actions"
    "github.com/google/uuid"
)

// GetStartedInstances is an interactive example that shows you how to use the AWS
// SDK for Go
// with Amazon Relation Database Service (Amazon RDS) to do the following:
//
// 1. Create a custom DB parameter group and set parameter values.
// 2. Create a DB instance that is configured to use the parameter group. The DB
//    instance
//    also contains a database.
// 3. Take a snapshot of the DB instance.
// 4. Delete the DB instance and parameter group.
type GetStartedInstances struct {
    sdkConfig  aws.Config
    instances  actions.DbInstances
    questioner demotools.IQuestioner
    helper     IScenarioHelper
    isTestRun  bool
}

// NewGetStartedInstances constructs a GetStartedInstances instance from a
// configuration.
// It uses the specified config to get an Amazon RDS
// client and create wrappers for the actions used in the scenario.
func NewGetStartedInstances(sdkConfig aws.Config, questioner demotools.IQuestioner,
    helper IScenarioHelper) GetStartedInstances {
    rdsClient := rds.NewFromConfig(sdkConfig)

```

```

return GetStartedInstances{
    sdkConfig:  sdkConfig,
    instances:  actions.DbInstances{RdsClient: rdsClient},
    questioner: questioner,
    helper:     helper,
}
}

// Run runs the interactive scenario.
func (scenario GetStartedInstances) Run(ctx context.Context, dbEngine string,
    parameterGroupName string,
    instanceName string, dbName string) {
    defer func() {
        if r := recover(); r != nil {
            log.Println("Something went wrong with the demo.")
        }
    }()

    log.Println(strings.Repeat("-", 88))
    log.Println("Welcome to the Amazon Relational Database Service (Amazon RDS) DB
    Instance demo.")
    log.Println(strings.Repeat("-", 88))

    parameterGroup := scenario.CreateParameterGroup(ctx, dbEngine, parameterGroupName)
    scenario.SetUserParameters(ctx, parameterGroupName)
    instance := scenario.CreateInstance(ctx, instanceName, dbEngine, dbName,
    parameterGroup)
    scenario.DisplayConnection(instance)
    scenario.CreateSnapshot(ctx, instance)
    scenario.Cleanup(ctx, instance, parameterGroup)

    log.Println(strings.Repeat("-", 88))
    log.Println("Thanks for watching!")
    log.Println(strings.Repeat("-", 88))
}

// CreateParameterGroup shows how to get available engine versions for a specified
// database engine and create a DB parameter group that is compatible with a
// selected engine family.
func (scenario GetStartedInstances) CreateParameterGroup(ctx context.Context,
    dbEngine string,
    parameterGroupName string) *types.DBParameterGroup {

    log.Printf("Checking for an existing DB parameter group named %v.\n",

```

```

    parameterGroupName)
parameterGroup, err := scenario.instances.GetParameterGroup(ctx,
parameterGroupName)
if err != nil {
    panic(err)
}
if parameterGroup == nil {
    log.Printf("Getting available database engine versions for %v.\n", dbEngine)
    engineVersions, err := scenario.instances.GetEngineVersions(ctx, dbEngine, "")
    if err != nil {
        panic(err)
    }

    familySet := map[string]struct{}{}
    for _, family := range engineVersions {
        familySet[*family.DBParameterGroupFamily] = struct{}{}
    }
    var families []string
    for family := range familySet {
        families = append(families, family)
    }
    sort.Strings(families)
    familyIndex := scenario.questioner.AskChoice("Which family do you want to use?\n",
families)
    log.Println("Creating a DB parameter group.")
    _, err = scenario.instances.CreateParameterGroup(
        ctx, parameterGroupName, families[familyIndex], "Example parameter group.")
    if err != nil {
        panic(err)
    }
    parameterGroup, err = scenario.instances.GetParameterGroup(ctx,
parameterGroupName)
    if err != nil {
        panic(err)
    }
}
log.Printf("Parameter group %v:\n", *parameterGroup.DBParameterGroupFamily)
log.Printf("\tName: %v\n", *parameterGroup.DBParameterGroupName)
log.Printf("\tARN: %v\n", *parameterGroup.DBParameterGroupArn)
log.Printf("\tFamily: %v\n", *parameterGroup.DBParameterGroupFamily)
log.Printf("\tDescription: %v\n", *parameterGroup.Description)
log.Println(strings.Repeat("-", 88))
return parameterGroup
}

```

```
// SetUserParameters shows how to get the parameters contained in a custom parameter
// group and update some of the parameter values in the group.
func (scenario GetStartedInstances) SetUserParameters(ctx context.Context,
    parameterGroupName string) {
    log.Println("Let's set some parameter values in your parameter group.")
    dbParameters, err := scenario.instances.GetParameters(ctx, parameterGroupName, "")
    if err != nil {
        panic(err)
    }
    var updateParams []types.Parameter
    for _, dbParam := range dbParameters {
        if strings.HasPrefix(*dbParam.ParameterName, "auto_increment") &&
            *dbParam.IsModifiable && *dbParam.DataType == "integer" {
            log.Printf("The %v parameter is described as:\n\t%v",
                *dbParam.ParameterName, *dbParam.Description)
            rangeSplit := strings.Split(*dbParam.AllowedValues, "-")
            lower, _ := strconv.Atoi(rangeSplit[0])
            upper, _ := strconv.Atoi(rangeSplit[1])
            newValue := scenario.questioner.AskInt(
                fmt.Sprintf("Enter a value between %v and %v:", lower, upper),
                demotools.InIntRange{Lower: lower, Upper: upper})
            dbParam.ParameterValue = aws.String(strconv.Itoa(newValue))
            updateParams = append(updateParams, dbParam)
        }
    }
    err = scenario.instances.UpdateParameters(ctx, parameterGroupName, updateParams)
    if err != nil {
        panic(err)
    }
    log.Println("To get a list of parameters that you set previously, specify a source
    of 'user'.")
    userParameters, err := scenario.instances.GetParameters(ctx, parameterGroupName,
    "user")
    if err != nil {
        panic(err)
    }
    log.Println("Here are the parameters you set:")
    for _, param := range userParameters {
        log.Printf("\t%v: %v\n", *param.ParameterName, *param.ParameterValue)
    }
    log.Println(strings.Repeat("-", 88))
}
```



```
// CreateInstance shows how to create a DB instance that contains a database of a
// specified type. The database is also configured to use a custom DB parameter
// group.
func (scenario GetStartedInstances) CreateInstance(ctx context.Context, instanceName
string, dbEngine string,
dbName string, parameterGroup *types.DBParameterGroup) *types.DBInstance {

    log.Println("Checking for an existing DB instance.")
    instance, err := scenario.instances.GetInstance(ctx, instanceName)
    if err != nil {
        panic(err)
    }
    if instance == nil {
        adminUsername := scenario.questioner.Ask(
            "Enter an administrator username for the database: ", demotools.NotEmpty{})
        adminPassword := scenario.questioner.AskPassword(
            "Enter a password for the administrator (at least 8 characters): ", 7)
        engineVersions, err := scenario.instances.GetEngineVersions(ctx, dbEngine,
            *parameterGroup.DBParameterGroupFamily)
        if err != nil {
            panic(err)
        }
        var engineChoices []string
        for _, engine := range engineVersions {
            engineChoices = append(engineChoices, *engine.EngineVersion)
        }
        engineIndex := scenario.questioner.AskChoice(
            "The available engines for your parameter group are:\n", engineChoices)
        engineSelection := engineVersions[engineIndex]
        instOpts, err := scenario.instances.GetOrderableInstances(ctx,
            *engineSelection.Engine,
            *engineSelection.EngineVersion)
        if err != nil {
            panic(err)
        }
        optSet := map[string]struct{}{}
        for _, opt := range instOpts {
            if strings.Contains(*opt.DBInstanceClass, "micro") {
                optSet[*opt.DBInstanceClass] = struct{}{}
            }
        }
        var optChoices []string
        for opt := range optSet {
            optChoices = append(optChoices, opt)
        }
    }
}
```

```

}
sort.Strings(optChoices)
optIndex := scenario.questioner.AskChoice(
    "The available micro DB instance classes for your database engine are:\n",
    optChoices)
storageType := "standard"
allocatedStorage := int32(5)
log.Printf("Creating a DB instance named %v and database %v.\n"+
    "The DB instance is configured to use your custom parameter group %v,\n"+
    "selected engine %v,\n"+
    "selected DB instance class %v,"+
    "and %v GiB of %v storage.\n"+
    "This typically takes several minutes.",
    instanceName, dbName, *parameterGroup.DBParameterGroupName,
    *engineSelection.EngineVersion,
    optChoices[optIndex], allocatedStorage, storageType)
instance, err = scenario.instances.CreateInstance(
    ctx, instanceName, dbName, *engineSelection.Engine,
    *engineSelection.EngineVersion,
    *parameterGroup.DBParameterGroupName, optChoices[optIndex], storageType,
    allocatedStorage, adminUsername, adminPassword)
if err != nil {
    panic(err)
}
for *instance.DBInstanceStatus != "available" {
    scenario.helper.Pause(30)
    instance, err = scenario.instances.GetInstance(ctx, instanceName)
    if err != nil {
        panic(err)
    }
}
log.Println("Instance created and available.")
}
log.Println("Instance data:")
log.Printf("\tDBInstanceIdentifier: %v\n", *instance.DBInstanceIdentifier)
log.Printf("\tARN: %v\n", *instance.DBInstanceArn)
log.Printf("\tStatus: %v\n", *instance.DBInstanceStatus)
log.Printf("\tEngine: %v\n", *instance.Engine)
log.Printf("\tEngine version: %v\n", *instance.EngineVersion)
log.Println(strings.Repeat("-", 88))
return instance
}

// DisplayConnection displays connection information about a DB instance and tips

```

```
// on how to connect to it.
func (scenario GetStartedInstances) DisplayConnection(instance *types.DBInstance) {
    log.Println(
        "You can now connect to your database by using your favorite MySQL client.\n" +
        "One way to connect is by using the 'mysql' shell on an Amazon EC2 instance\n" +
        "that is running in the same VPC as your DB instance. Pass the endpoint,\n" +
        "port, and administrator username to 'mysql'. Then, enter your password\n" +
        "when prompted:")
    log.Printf("\n\tmysql -h %v -P %v -u %v -p\n",
        *instance.Endpoint.Address, instance.Endpoint.Port, *instance.MasterUsername)
    log.Println("For more information, see the User Guide for RDS:\n" +
        "\thttps://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/
CHAP_GettingStarted.CreatingConnecting.MySQL.html#CHAP_GettingStarted.Connecting.MySQL")
    log.Println(strings.Repeat("-", 88))
}

// CreateSnapshot shows how to create a DB instance snapshot and wait until it's
// available.
func (scenario GetStartedInstances) CreateSnapshot(ctx context.Context, instance
    *types.DBInstance) {
    if scenario.questioner.AskBool(
        "Do you want to create a snapshot of your DB instance (y/n)? ", "y") {
        snapshotId := fmt.Sprintf("%v-%v", *instance.DBInstanceIdentifier,
            scenario.helper.UniqueId())
        log.Printf("Creating a snapshot named %v. This typically takes a few minutes.\n",
            snapshotId)
        snapshot, err := scenario.instances.CreateSnapshot(ctx,
            *instance.DBInstanceIdentifier, snapshotId)
        if err != nil {
            panic(err)
        }
        for *snapshot.Status != "available" {
            scenario.helper.Pause(30)
            snapshot, err = scenario.instances.GetSnapshot(ctx, snapshotId)
            if err != nil {
                panic(err)
            }
        }
        log.Println("Snapshot data:")
        log.Printf("\tDBSnapshotIdentifier: %v\n", *snapshot.DBSnapshotIdentifier)
        log.Printf("\tARN: %v\n", *snapshot.DBSnapshotArn)
        log.Printf("\tStatus: %v\n", *snapshot.Status)
        log.Printf("\tEngine: %v\n", *snapshot.Engine)
        log.Printf("\tEngine version: %v\n", *snapshot.EngineVersion)
    }
}
```

```

    log.Printf("\tDBInstanceIdentifier: %v\n", *snapshot.DBInstanceIdentifier)
    log.Printf("\tSnapshotCreateTime: %v\n", *snapshot.SnapshotCreateTime)
    log.Println(strings.Repeat("-", 88))
}
}

// Cleanup shows how to clean up a DB instance and DB parameter group.
// Before the DB parameter group can be deleted, all associated DB instances must
// first be deleted.
func (scenario GetStartedInstances) Cleanup(
    ctx context.Context, instance *types.DBInstance, parameterGroup
    *types.DBParameterGroup) {

    if scenario.questioner.AskBool(
        "\nDo you want to delete the database instance and parameter group (y/n)? ", "y")
    {
        log.Printf("Deleting database instance %v.\n", *instance.DBInstanceIdentifier)
        err := scenario.instances.DeleteInstance(ctx, *instance.DBInstanceIdentifier)
        if err != nil {
            panic(err)
        }
        log.Println(
            "Waiting for the DB instance to delete. This typically takes several minutes.")
        for instance != nil {
            scenario.helper.Pause(30)
            instance, err = scenario.instances.GetInstance(ctx,
                *instance.DBInstanceIdentifier)
            if err != nil {
                panic(err)
            }
        }
        log.Printf("Deleting parameter group %v.", *parameterGroup.DBParameterGroupName)
        err = scenario.instances.DeleteParameterGroup(ctx,
            *parameterGroup.DBParameterGroupName)
        if err != nil {
            panic(err)
        }
    }
}

// IScenarioHelper abstracts the function from a scenario so that it
// can be mocked for unit testing.
type IScenarioHelper interface {
    Pause(secs int)

```

```

    UniqueId() string
}
type ScenarioHelper struct{}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
    time.Sleep(time.Duration(secs) * time.Second)
}

// UniqueId returns a new UUID.
func (helper ScenarioHelper) UniqueId() string {
    return uuid.New().String()
}

```

Defina las funciones a las que llama el escenario para administrar las acciones de Amazon RDS.

```

import (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/rds"
    "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
    RdsClient *rds.Client
}

// GetParameterGroup gets a DB parameter group by name.
func (instances *DbInstances) GetParameterGroup(ctx context.Context,
    parameterGroupName string) (
    *types.DBParameterGroup, error) {
    output, err := instances.RdsClient.DescribeDBParameterGroups(
        ctx, &rds.DescribeDBParameterGroupsInput{
            DBParameterGroupName: aws.String(parameterGroupName),
        })
    if err != nil {

```

```

var notFoundError *types.DBParameterGroupNotFoundFault
if errors.As(err, &notFoundError) {
    log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
    err = nil
} else {
    log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
}
return nil, err
} else {
    return &output.DBParameterGroups[0], err
}
}

// CreateParameterGroup creates a DB parameter group that is based on the specified
// parameter group family.
func (instances *DbInstances) CreateParameterGroup(
    ctx context.Context, parameterGroupName string, parameterGroupFamily string,
    description string) (
    *types.DBParameterGroup, error) {

    output, err := instances.RdsClient.CreateDBParameterGroup(ctx,
        &rds.CreateDBParameterGroupInput{
            DBParameterGroupName:    aws.String(parameterGroupName),
            DBParameterGroupFamily: aws.String(parameterGroupFamily),
            Description:            aws.String(description),
        })
    if err != nil {
        log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
        return nil, err
    } else {
        return output.DBParameterGroup, err
    }
}

// DeleteParameterGroup deletes the named DB parameter group.
func (instances *DbInstances) DeleteParameterGroup(ctx context.Context,
    parameterGroupName string) error {
    _, err := instances.RdsClient.DeleteDBParameterGroup(ctx,
        &rds.DeleteDBParameterGroupInput{
            DBParameterGroupName: aws.String(parameterGroupName),
        })
    if err != nil {
        log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
        return err
    }
    return nil
}

```

```

    })
    if err != nil {
        log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
        return err
    } else {
        return nil
    }
}

// GetParameters gets the parameters that are contained in a DB parameter group.
func (instances *DbInstances) GetParameters(ctx context.Context, parameterGroupName
string, source string) (
[]types.Parameter, error) {

    var output *rds.DescribeDBParametersOutput
    var params []types.Parameter
    var err error
    parameterPaginator := rds.NewDescribeDBParametersPaginator(instances.RdsClient,
        &rds.DescribeDBParametersInput{
            DBParameterGroupName: aws.String(parameterGroupName),
            Source:                aws.String(source),
        })
    for parameterPaginator.HasMorePages() {
        output, err = parameterPaginator.NextPage(ctx)
        if err != nil {
            log.Printf("Couldn't get parameters for %v: %v\n", parameterGroupName, err)
            break
        } else {
            params = append(params, output.Parameters...)
        }
    }
    return params, err
}

// UpdateParameters updates parameters in a named DB parameter group.
func (instances *DbInstances) UpdateParameters(ctx context.Context,
parameterGroupName string, params []types.Parameter) error {
    _, err := instances.RdsClient.ModifyDBParameterGroup(ctx,
        &rds.ModifyDBParameterGroupInput{
            DBParameterGroupName: aws.String(parameterGroupName),

```

```

    Parameters:          params,
  })
  if err != nil {
    log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
    return err
  } else {
    return nil
  }
}

// CreateSnapshot creates a snapshot of a DB instance.
func (instances *DbInstances) CreateSnapshot(ctx context.Context, instanceName
string, snapshotName string) (
  *types.DBSnapshot, error) {
  output, err := instances.RdsClient.CreateDBSnapshot(ctx,
    &rds.CreateDBSnapshotInput{
      DBInstanceIdentifier: aws.String(instanceName),
      DBSnapshotIdentifier: aws.String(snapshotName),
    })
  if err != nil {
    log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
    return nil, err
  } else {
    return output.DBSnapshot, nil
  }
}

// GetSnapshot gets a DB instance snapshot.
func (instances *DbInstances) GetSnapshot(ctx context.Context, snapshotName string)
(*types.DBSnapshot, error) {
  output, err := instances.RdsClient.DescribeDBSnapshots(ctx,
    &rds.DescribeDBSnapshotsInput{
      DBSnapshotIdentifier: aws.String(snapshotName),
    })
  if err != nil {
    log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
    return nil, err
  } else {
    return &output.DBSnapshots[0], nil
  }
}

```



```

}

// CreateInstance creates a DB instance.
func (instances *DbInstances) CreateInstance(ctx context.Context, instanceName
    string, dbName string,
    dbEngine string, dbEngineVersion string, parameterGroupName string, dbInstanceClass
    string,
    storageType string, allocatedStorage int32, adminName string, adminPassword string)
    (
    *types.DBInstance, error) {
    output, err := instances.RdsClient.CreateDBInstance(ctx,
    &rds.CreateDBInstanceInput{
        DBInstanceIdentifier: aws.String(instanceName),
        DBName:                aws.String(dbName),
        DBParameterGroupName: aws.String(parameterGroupName),
        Engine:                aws.String(dbEngine),
        EngineVersion:         aws.String(dbEngineVersion),
        DBInstanceClass:       aws.String(dbInstanceClass),
        StorageType:           aws.String(storageType),
        AllocatedStorage:      aws.Int32(allocatedStorage),
        MasterUsername:        aws.String(adminName),
        MasterUserPassword:    aws.String(adminPassword),
    })
    if err != nil {
        log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
        return nil, err
    } else {
        return output.DBInstance, nil
    }
}

// GetInstance gets data about a DB instance.
func (instances *DbInstances) GetInstance(ctx context.Context, instanceName string)
    (
    *types.DBInstance, error) {
    output, err := instances.RdsClient.DescribeDBInstances(ctx,
    &rds.DescribeDBInstancesInput{
        DBInstanceIdentifier: aws.String(instanceName),
    })
    if err != nil {

```

```

var notFoundError *types.DBInstanceNotFoundFault
if errors.As(err, &notFoundError) {
    log.Printf("DB instance %v does not exist.\n", instanceName)
    err = nil
} else {
    log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
}
return nil, err
} else {
    return &output.DBInstances[0], nil
}
}

// DeleteInstance deletes a DB instance.
func (instances *DbInstances) DeleteInstance(ctx context.Context, instanceName
string) error {
    _, err := instances.RdsClient.DeleteDBInstance(ctx, &rds.DeleteDBInstanceInput{
        DBInstanceIdentifier:    aws.String(instanceName),
        SkipFinalSnapshot:      aws.Bool(true),
        DeleteAutomatedBackups: aws.Bool(true),
    })
    if err != nil {
        log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
        return err
    } else {
        return nil
    }
}

// GetEngineVersions gets database engine versions that are available for the
// specified engine
// and parameter group family.
func (instances *DbInstances) GetEngineVersions(ctx context.Context, engine string,
parameterGroupFamily string) (
[]types.DBEngineVersion, error) {
    output, err := instances.RdsClient.DescribeDBEngineVersions(ctx,
        &rds.DescribeDBEngineVersionsInput{
            Engine:                aws.String(engine),
            DBParameterGroupFamily: aws.String(parameterGroupFamily),
        })
}

```

```

if err != nil {
    log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
    return nil, err
} else {
    return output.DBEngineVersions, nil
}
}

// GetOrderableInstances uses a paginator to get DB instance options that can be
// used to create DB instances that are
// compatible with a set of specifications.
func (instances *DbInstances) GetOrderableInstances(ctx context.Context, engine
string, engineVersion string) (
[]types.OrderableDBInstanceOption, error) {

    var output *rds.DescribeOrderableDBInstanceOptionsOutput
    var instanceOptions []types.OrderableDBInstanceOption
    var err error
    orderablePaginator :=
rds.NewDescribeOrderableDBInstanceOptionsPaginator(instances.RdsClient,
    &rds.DescribeOrderableDBInstanceOptionsInput{
        Engine:      aws.String(engine),
        EngineVersion: aws.String(engineVersion),
    })
    for orderablePaginator.HasMorePages() {
        output, err = orderablePaginator.NextPage(ctx)
        if err != nil {
            log.Printf("Couldn't get orderable DB instance options: %v\n", err)
            break
        } else {
            instanceOptions = append(instanceOptions, output.OrderableDBInstanceOptions...)
        }
    }
    return instanceOptions, err
}

```

- Para obtener información sobre la API, consulte los siguientes temas en la referencia de la API de AWS SDK para Go .

- [CrearDBInstance](#)

- [Crear DBParameter grupo](#)
- [CrearDBSnapshot](#)
- [DeleteDBInstance](#)
- [Eliminar DBParameter grupo](#)
- [Describe DBEngine 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 Go V2

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](#).

```
import (  
    "context"  
    "errors"  
    "log"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/rds"  
    "github.com/aws/aws-sdk-go-v2/service/rds/types"  
)
```

```

type DbInstances struct {
    RdsClient *rds.Client
}

// CreateInstance creates a DB instance.
func (instances *DbInstances) CreateInstance(ctx context.Context, instanceName
    string, dbName string,
    dbEngine string, dbEngineVersion string, parameterGroupName string, dbInstanceClass
    string,
    storageType string, allocatedStorage int32, adminName string, adminPassword string)
    (
    *types.DBInstance, error) {
    output, err := instances.RdsClient.CreateDBInstance(ctx,
    &rds.CreateDBInstanceInput{
        DBInstanceIdentifier: aws.String(instanceName),
        DBName:               aws.String(dbName),
        DBParameterGroupName: aws.String(parameterGroupName),
        Engine:               aws.String(dbEngine),
        EngineVersion:        aws.String(dbEngineVersion),
        DBInstanceClass:      aws.String(dbInstanceClass),
        StorageType:          aws.String(storageType),
        AllocatedStorage:      aws.Int32(allocatedStorage),
        MasterUsername:        aws.String(adminName),
        MasterUserPassword:    aws.String(adminPassword),
    })
    if err != nil {
        log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
        return nil, err
    } else {
        return output.DBInstance, nil
    }
}

```

- Para obtener más información sobre la API, consulta la [sección Crear DBInstance](#) en la referencia de la AWS SDK para Go API.

CreateDBParameterGroup

En el siguiente ejemplo de código, se muestra cómo utilizar CreateDBParameterGroup.

SDK para Go V2

 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 (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/rds"
    "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
    RdsClient *rds.Client
}

// CreateParameterGroup creates a DB parameter group that is based on the specified
// parameter group family.
func (instances *DbInstances) CreateParameterGroup(
    ctx context.Context, parameterGroupName string, parameterGroupFamily string,
    description string) (
    *types.DBParameterGroup, error) {

    output, err := instances.RdsClient.CreateDBParameterGroup(ctx,
        &rds.CreateDBParameterGroupInput{
            DBParameterGroupName:  aws.String(parameterGroupName),
            DBParameterGroupFamily: aws.String(parameterGroupFamily),
            Description:          aws.String(description),
        })
    if err != nil {
        log.Printf("Couldn't create parameter group %v: %v\n", parameterGroupName, err)
        return nil, err
    } else {
```

```
    return output.DBParameterGroup, err
  }
}
```

- Para obtener más información sobre la API, consulta [Crear un DBParameter grupo](#) en la referencia AWS SDK para Go de la API.

CreateDBSnapshot

En el siguiente ejemplo de código, se muestra cómo utilizar CreateDBSnapshot.

SDK para Go V2

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 (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/rds"
    "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
    RdsClient *rds.Client
}

// CreateSnapshot creates a snapshot of a DB instance.
func (instances *DbInstances) CreateSnapshot(ctx context.Context, instanceName
    string, snapshotName string) (
```

```

*types.DBSnapshot, error) {
output, err := instances.RdsClient.CreateDBSnapshot(ctx,
&rds.CreateDBSnapshotInput{
    DBInstanceIdentifier: aws.String(instanceName),
    DBSnapshotIdentifier: aws.String(snapshotName),
})
if err != nil {
    log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
    return nil, err
} else {
    return output.DBSnapshot, nil
}
}

```

- Para obtener más información sobre la API, consulta la [sección Crear DBSnapshot](#) en la referencia de la AWS SDK para Go API.

DeleteDBInstance

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteDBInstance.

SDK para Go V2

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 (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/rds"
    "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

```



```
type DbInstances struct {
    RdsClient *rds.Client
}

// DeleteInstance deletes a DB instance.
func (instances *DbInstances) DeleteInstance(ctx context.Context, instanceName
    string) error {
    _, err := instances.RdsClient.DeleteDBInstance(ctx, &rds.DeleteDBInstanceInput{
        DBInstanceIdentifier:    aws.String(instanceName),
        SkipFinalSnapshot:       aws.Bool(true),
        DeleteAutomatedBackups:  aws.Bool(true),
    })
    if err != nil {
        log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
        return err
    } else {
        return nil
    }
}
```

- Para obtener más información sobre la API, consulta [Eliminar DBInstance](#) en la referencia AWS SDK para Go de la API.

DeleteDBParameterGroup

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteDBParameterGroup.

SDK para Go V2

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 (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/rds"
    "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
    RdsClient *rds.Client
}

// DeleteParameterGroup deletes the named DB parameter group.
func (instances *DbInstances) DeleteParameterGroup(ctx context.Context,
    parameterGroupName string) error {
    _, err := instances.RdsClient.DeleteDBParameterGroup(ctx,
        &rds.DeleteDBParameterGroupInput{
            DBParameterGroupName: aws.String(parameterGroupName),
        })
    if err != nil {
        log.Printf("Couldn't delete parameter group %v: %v\n", parameterGroupName, err)
        return err
    } else {
        return nil
    }
}
```

- Para obtener más información sobre la API, consulta [Eliminar DBParameter grupo](#) en la referencia AWS SDK para Go de la API.

DescribeDBEngineVersions

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeDBEngineVersions.

SDK para Go V2

 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 (  
    "context"  
    "errors"  
    "log"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/rds"  
    "github.com/aws/aws-sdk-go-v2/service/rds/types"  
)  
  
type DbInstances struct {  
    RdsClient *rds.Client  
}  
  
// GetEngineVersions gets database engine versions that are available for the  
// specified engine  
// and parameter group family.  
func (instances *DbInstances) GetEngineVersions(ctx context.Context, engine string,  
    parameterGroupFamily string) (  
    []types.DBEngineVersion, error) {  
    output, err := instances.RdsClient.DescribeDBEngineVersions(ctx,  
        &rds.DescribeDBEngineVersionsInput{  
            Engine:          aws.String(engine),  
            DBParameterGroupFamily: aws.String(parameterGroupFamily),  
        })  
    if err != nil {  
        log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)  
        return nil, err  
    } else {  
        return output.DBEngineVersions, nil  
    }  
}
```

```
}
```

- Para obtener más información sobre la API, consulta la [sección Describir DBEngine las versiones](#) en la referencia de la AWS SDK para Go API.

DescribeDBInstances

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeDBInstances.

SDK para Go V2

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 (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/rds"
    "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
    RdsClient *rds.Client
}

// GetInstance gets data about a DB instance.
func (instances *DbInstances) GetInstance(ctx context.Context, instanceName string) (
    *types.DBInstance, error) {
    output, err := instances.RdsClient.DescribeDBInstances(ctx,
```

```
&rds.DescribeDBInstancesInput{
    DBInstanceIdentifier: aws.String(instanceName),
})
if err != nil {
    var notFoundError *types.DBInstanceNotFoundFault
    if errors.As(err, &notFoundError) {
        log.Printf("DB instance %v does not exist.\n", instanceName)
        err = nil
    } else {
        log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
    }
    return nil, err
} else {
    return &output.DBInstances[0], nil
}
}
```

- Para obtener más información sobre la API, consulta la [sección Describir DBInstances](#) en la referencia de la AWS SDK para Go API.

DescribeDBParameterGroups

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeDBParameterGroups.

SDK para Go V2

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 (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
```

```

"github.com/aws/aws-sdk-go-v2/service/rds"
"github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
    RdsClient *rds.Client
}

// GetParameterGroup gets a DB parameter group by name.
func (instances *DbInstances) GetParameterGroup(ctx context.Context,
    parameterGroupName string) (
    *types.DBParameterGroup, error) {
    output, err := instances.RdsClient.DescribeDBParameterGroups(
        ctx, &rds.DescribeDBParameterGroupsInput{
            DBParameterGroupName: aws.String(parameterGroupName),
        })
    if err != nil {
        var notFoundError *types.DBParameterGroupNotFoundFault
        if errors.As(err, &notFoundError) {
            log.Printf("Parameter group %v does not exist.\n", parameterGroupName)
            err = nil
        } else {
            log.Printf("Error getting parameter group %v: %v\n", parameterGroupName, err)
        }
        return nil, err
    } else {
        return &output.DBParameterGroups[0], err
    }
}

```

- Para obtener más información sobre la API, consulta la [sección Describir DBParameter los grupos](#) en la referencia de la AWS SDK para Go API.

DescribeDBParameters

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeDBParameters.

SDK para Go V2

 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 (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/rds"
    "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
    RdsClient *rds.Client
}

// GetParameters gets the parameters that are contained in a DB parameter group.
func (instances *DbInstances) GetParameters(ctx context.Context, parameterGroupName
string, source string) (
    []types.Parameter, error) {

    var output *rds.DescribeDBParametersOutput
    var params []types.Parameter
    var err error
    parameterPaginator := rds.NewDescribeDBParametersPaginator(instances.RdsClient,
        &rds.DescribeDBParametersInput{
            DBParameterGroupName: aws.String(parameterGroupName),
            Source:                aws.String(source),
        })
    for parameterPaginator.HasMorePages() {
        output, err = parameterPaginator.NextPage(ctx)
        if err != nil {
            log.Printf("Couldn't get parameters for %v: %v\n", parameterGroupName, err)
        }
    }
}
```

```
    break
  } else {
    params = append(params, output.Parameters...)
  }
}
return params, err
}
```

- Para obtener más información sobre la API, consulta la [sección Describir DBParameters](#) en la referencia de la AWS SDK para Go API.

DescribeDBSnapshots

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeDBSnapshots.

SDK para Go V2

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 (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/rds"
    "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
    RdsClient *rds.Client
}
```



```
// GetSnapshot gets a DB instance snapshot.
func (instances *DbInstances) GetSnapshot(ctx context.Context, snapshotName string)
(*types.DBSnapshot, error) {
    output, err := instances.RdsClient.DescribeDBSnapshots(ctx,
        &rds.DescribeDBSnapshotsInput{
            DBSnapshotIdentifier: aws.String(snapshotName),
        })
    if err != nil {
        log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
        return nil, err
    } else {
        return &output.DBSnapshots[0], nil
    }
}
```

- Para obtener más información sobre la API, consulta la [sección Describir DBSnapshots](#) en la referencia de la AWS SDK para Go API.

DescribeOrderableDBInstanceOptions

En el siguiente ejemplo de código, se muestra cómo utilizar `DescribeOrderableDBInstanceOptions`.

SDK para Go V2

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 (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
```

```

"github.com/aws/aws-sdk-go-v2/service/rds"
"github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
  RdsClient *rds.Client
}

// GetOrderableInstances uses a paginator to get DB instance options that can be
// used to create DB instances that are
// compatible with a set of specifications.
func (instances *DbInstances) GetOrderableInstances(ctx context.Context, engine
string, engineVersion string) (
[]types.OrderableDBInstanceOption, error) {

var output *rds.DescribeOrderableDBInstanceOptionsOutput
var instanceOptions []types.OrderableDBInstanceOption
var err error
orderablePaginator :=
rds.NewDescribeOrderableDBInstanceOptionsPaginator(instances.RdsClient,
&rds.DescribeOrderableDBInstanceOptionsInput{
  Engine:      aws.String(engine),
  EngineVersion: aws.String(engineVersion),
})
for orderablePaginator.HasMorePages() {
  output, err = orderablePaginator.NextPage(ctx)
  if err != nil {
    log.Printf("Couldn't get orderable DB instance options: %v\n", err)
    break
  } else {
    instanceOptions = append(instanceOptions, output.OrderableDBInstanceOptions...)
  }
}
return instanceOptions, err
}

```

- Para obtener más información sobre la API, consulta [DescribeOrderableDBInstanceOptions](#) en la referencia AWS SDK para Go de la API.

ModifyDBParameterGroup

En el siguiente ejemplo de código, se muestra cómo utilizar `ModifyDBParameterGroup`.

SDK para Go V2

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 (
    "context"
    "errors"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/rds"
    "github.com/aws/aws-sdk-go-v2/service/rds/types"
)

type DbInstances struct {
    RdsClient *rds.Client
}

// UpdateParameters updates parameters in a named DB parameter group.
func (instances *DbInstances) UpdateParameters(ctx context.Context,
    parameterGroupName string, params []types.Parameter) error {
    _, err := instances.RdsClient.ModifyDBParameterGroup(ctx,
        &rds.ModifyDBParameterGroupInput{
            DBParameterGroupName: aws.String(parameterGroupName),
            Parameters:           params,
        })
    if err != nil {
        log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
        return err
    } else {
        return nil
    }
}
```

```
}
```

- Para obtener más información sobre la API, consulta [Modificar el DBParameter grupo](#) en la referencia de la AWS SDK para Go API.

Ejemplos de tecnología sin servidor

Conexión a una base de datos de Amazon RDS en una función de Lambda

En el siguiente ejemplo de código, se muestra cómo implementar una función de Lambda que se conecta a una base de datos de RDS. La función realiza una solicitud sencilla a la base de datos y devuelve el resultado.

SDK para Go V2

Note

Hay más información al respecto GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Conexión a una base de datos de Amazon RDS en una función de Lambda mediante Go.

```
/*
Golang v2 code here.
*/

package main

import (
    "context"
    "database/sql"
    "encoding/json"
    "fmt"
    "os"

    "github.com/aws/aws-lambda-go/lambda"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/feature/rds/auth"

```

```

_ "github.com/go-sql-driver/mysql"
)

type MyEvent struct {
    Name string `json:"name"`
}

func HandleRequest(event *MyEvent) (map[string]interface{}, error) {

    var dbName string = os.Getenv("DatabaseName")
    var dbUser string = os.Getenv("DatabaseUser")
    var dbHost string = os.Getenv("DBHost") // Add hostname without https
    var dbPort int = os.Getenv("Port")      // Add port number
    var dbEndpoint string = fmt.Sprintf("%s:%d", dbHost, dbPort)
    var region string = os.Getenv("AWS_REGION")

    cfg, err := config.LoadDefaultConfig(context.TODO())
    if err != nil {
        panic("configuration error: " + err.Error())
    }

    authenticationToken, err := auth.BuildAuthToken(
        context.TODO(), dbEndpoint, region, dbUser, cfg.Credentials)
    if err != nil {
        panic("failed to create authentication token: " + err.Error())
    }

    dsn := fmt.Sprintf("%s:%s@tcp(%s)/%s?tls=true&allowCleartextPasswords=true",
        dbUser, authenticationToken, dbEndpoint, dbName,
    )

    db, err := sql.Open("mysql", dsn)
    if err != nil {
        panic(err)
    }

    defer db.Close()

    var sum int
    err = db.QueryRow("SELECT ?+? AS sum", 3, 2).Scan(&sum)
    if err != nil {
        panic(err)
    }

    s := fmt.Sprint(sum)

```

```
message := fmt.Sprintf("The selected sum is: %s", s)

messageBytes, err := json.Marshal(message)
if err != nil {
    return nil, err
}

messageString := string(messageBytes)
return map[string]interface{}{
    "statusCode": 200,
    "headers":    map[string]string{"Content-Type": "application/json"},
    "body":       messageString,
}, nil
}

func main() {
    lambda.Start(HandleRequest)
}
```

Ejemplos de Amazon Redshift con SDK para Go V2

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso de la AWS SDK para Go V2 con Amazon Redshift.

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

Hello Amazon Redshift

En el siguiente ejemplo de código se muestra cómo empezar a utilizar Amazon Redshift.

SDK para Go V2

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](#).

```
package main

import (
    "context"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/redshift"
)

// main uses the AWS SDK for Go V2 to create a Redshift client
// and list up to 10 clusters in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
        fmt.Println(err)
        return
    }
    redshiftClient := redshift.NewFromConfig(sdkConfig)
    count := 20
    fmt.Printf("Let's list up to %v clusters for your account.\n", count)
```

```
result, err := redshiftClient.DescribeClusters(ctx,
&redshift.DescribeClustersInput{
    MaxRecords: aws.Int32(int32(count)),
})
if err != nil {
    fmt.Printf("Couldn't list clusters for your account. Here's why: %v\n", err)
    return
}
if len(result.Clusters) == 0 {
    fmt.Println("You don't have any clusters!")
    return
}
for _, cluster := range result.Clusters {
    fmt.Printf("\t\t%v : %v\n", *cluster.ClusterIdentifier, *cluster.ClusterStatus)
}
}
```

- Para obtener más información sobre la API, consulta [DescribeClusters](#) la Referencia AWS SDK para Go de la API.

Conceptos básicos

Conceptos básicos

En el siguiente ejemplo de código, se muestra cómo:

- Crear un clúster de Redshift.
- Enumerar las bases de datos del clúster.
- Crear una tabla denominada "Movies".
- Rellenar la tabla "Movies".
- Consultar la tabla "Movies" por año.
- Modificar el clúster de Redshift.
- Eliminar el clúster de Amazon Redshift.

SDK para Go V2

 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](#).

```
package scenarios

import (
    "context"
    "encoding/json"
    "errors"
    "fmt"
    "log"
    "math/rand"
    "strings"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    redshift_types "github.com/aws/aws-sdk-go-v2/service/redshift/types"
    redshiftdata_types "github.com/aws/aws-sdk-go-v2/service/redshiftdata/types"
    "github.com/aws/aws-sdk-go-v2/service/secretsmanager"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/redshift/actions"

    "github.com/aws/aws-sdk-go-v2/service/redshift"
    "github.com/aws/aws-sdk-go-v2/service/redshiftdata"
)

// IScenarioHelper abstracts input and wait functions from a scenario so that they
// can be mocked for unit testing.
type IScenarioHelper interface {
    GetName() string
}

const rMax = 100000

type ScenarioHelper struct {
    Prefix string
}
```

```

    Random *rand.Rand
}

// GetName returns a unique name formed of a prefix and a random number.
func (helper ScenarioHelper) GetName() string {
    return fmt.Sprintf("%v%v", helper.Prefix, helper.Random.Intn(rMax))
}

// RedshiftBasicsScenario separates the steps of this scenario into individual
// functions so that
// they are simpler to read and understand.
type RedshiftBasicsScenario struct {
    sdkConfig      aws.Config
    helper          IScenarioHelper
    questioner      demotools.IQuestioner
    pauser          demotools.IPausable
    filesystem      demotools.IFileSystem
    redshiftActor   *actions.RedshiftActions
    redshiftDataActor *actions.RedshiftDataActions
    secretsmanager  *SecretsManager
}

// SecretsManager is used to retrieve username and password information from a
// secure service.
type SecretsManager struct {
    SecretsManagerClient *secretsmanager.Client
}

// RedshiftBasics constructs a new Redshift Basics runner.
func RedshiftBasics(sdkConfig aws.Config, questioner demotools.IQuestioner, pauser
demotools.IPausable, filesystem demotools.IFileSystem, helper IScenarioHelper)
RedshiftBasicsScenario {
    scenario := RedshiftBasicsScenario{
        sdkConfig:      sdkConfig,
        helper:          helper,
        questioner:      questioner,
        pauser:          pauser,
        filesystem:      filesystem,
        secretsmanager:  &SecretsManager{SecretsManagerClient:
secretsmanager.NewFromConfig(sdkConfig)},
        redshiftActor:   &actions.RedshiftActions{RedshiftClient:
redshift.NewFromConfig(sdkConfig)},
        redshiftDataActor: &actions.RedshiftDataActions{RedshiftDataClient:
redshiftdata.NewFromConfig(sdkConfig)},
    }
}

```

```

}
return scenario
}

// Movie makes it easier to use Movie objects given in json format.
type Movie struct {
    ID    int    `json:"id"`
    Title string `json:"title"`
    Year  int    `json:"year"`
}

// User makes it easier to get the User data back from SecretsManager and use it
// later.
type User struct {
    Username string `json:"userName"`
    Password string `json:"userPassword"`
}

// Run runs the RedshiftBasics interactive example that shows you how to use Amazon
// Redshift and how to interact with its common endpoints.
//
// 0. Retrieve username and password information to access Redshift.
// 1. Create a cluster.
// 2. Wait for the cluster to become available.
// 3. List the available databases in the region.
// 4. Create a table named "Movies" in the "dev" database.
// 5. Populate the movies table from the "movies.json" file.
// 6. Query the movies table by year.
// 7. Modify the cluster's maintenance window.
// 8. Optionally clean up all resources created during this demo.
//
// This example creates an Amazon Redshift service client from the specified
// sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
// example.
// This package can be found in the ../..demotools folder of this repo.
func (runner *RedshiftBasicsScenario) Run(ctx context.Context) {

    user := User{}
    secretId := "s3express/basics/secrets"

```

```

clusterId := "demo-cluster-1"
maintenanceWindow := "wed:07:30-wed:08:00"
databaseName := "dev"
tableName := "Movies"
fileName := "Movies.json"
nodeType := "ra3.xlplus"
clusterType := "single-node"

defer func() {
    if r := recover(); r != nil {
        log.Println("Something went wrong with the demo.")
        _, isMock := runner.questioner.(*demotools.MockQuestioner)
        if isMock || runner.questioner.AskBool("Do you want to see the full error message (y/n)?", "y") {
            log.Println(r)
        }
        runner.cleanUpResources(ctx, clusterId, databaseName, tableName, user.Username, runner.questioner)
    }
}()

// Retrieve the userName and userPassword from SecretsManager
output, err := runner.secretsmanager.SecretsManagerClient.GetSecretValue(ctx,
    &secretsmanager.GetSecretValueInput{
        SecretId: aws.String(secretId),
    })
if err != nil {
    log.Printf("There was a problem getting the secret value: %s", err)
    log.Printf("Please make sure to create a secret named 's3express/basics/secrets' with keys of 'userName' and 'userPassword'.")
    panic(err)
}

err = json.Unmarshal([]byte(*output.SecretString), &user)
if err != nil {
    log.Printf("There was a problem parsing the secret value from JSON: %s", err)
    panic(err)
}

// Create the Redshift cluster
_, err = runner.redshiftActor.CreateCluster(ctx, clusterId, user.Username, user.Password, nodeType, clusterType, true)
if err != nil {
    var clusterAlreadyExistsFault *redshift_types.ClusterAlreadyExistsFault

```

```

    if errors.As(err, &clusterAlreadyExistsFault) {
        log.Println("Cluster already exists. Continuing.")
    } else {
        log.Println("Error creating cluster.")
        panic(err)
    }
}

// Wait for the cluster to become available
waiter := redshift.NewClusterAvailableWaiter(runner.redshiftActor.RedshiftClient)
err = waiter.Wait(ctx, &redshift.DescribeClustersInput{
    ClusterIdentifier: aws.String(clusterId),
}, 5*time.Minute)
if err != nil {
    log.Println("An error occurred waiting for the cluster.")
    panic(err)
}

// Get some info about the cluster
describeOutput, err := runner.redshiftActor.DescribeClusters(ctx, clusterId)
if err != nil {
    log.Println("Something went wrong trying to get information about the cluster.")
    panic(err)
}
log.Println("Here's some information about the cluster.")
log.Printf("The cluster's status is %s", *describeOutput.Clusters[0].ClusterStatus)
log.Printf("The cluster was created at %s",
*describeOutput.Clusters[0].ClusterCreateTime)

// List databases
log.Println("List databases in", clusterId)
runner.questioner.Ask("Press Enter to continue...")
err = runner.redshiftDataActor.ListDatabases(ctx, clusterId, databaseName,
user.Username)
if err != nil {
    log.Printf("Failed to list databases: %v\n", err)
    panic(err)
}

// Create the "Movies" table
log.Println("Now you will create a table named " + tableName + ".")
runner.questioner.Ask("Press Enter to continue...")
err = nil

```

```

result, err := runner.redshiftDataActor.CreateTable(ctx, clusterId, databaseName,
tableName, user.Username, runner.pauser, []string{"title VARCHAR(256)", "year
INT"})
if err != nil {
    log.Printf("Failed to create table: %v\n", err)
    panic(err)
}

describeInput := redshiftdata.DescribeStatementInput{
    Id: result.Id,
}
query := actions.RedshiftQuery{
    Context: ctx,
    Input:    describeInput,
    Result:   result,
}
err = runner.redshiftDataActor.WaitForQueryStatus(query, runner.pauser, true)
if err != nil {
    log.Printf("Failed to execute query: %v\n", err)
    panic(err)
}
log.Printf("Successfully executed query\n")

// Populate the "Movies" table
runner.PopulateMoviesTable(ctx, clusterId, databaseName, tableName, user.Username,
fileName)

// Query the "Movies" table by year
log.Println("Query the Movies table by year.")
year := runner.questioner.AskInt(
    fmt.Sprintf("Enter a value between %v and %v:", 2012, 2014),
    demotools.InIntRange{Lower: 2012, Upper: 2014})
runner.QueryMoviesByYear(ctx, clusterId, databaseName, tableName, user.Username,
year)

// Modify the cluster's maintenance window
runner.redshiftActor.ModifyCluster(ctx, clusterId, maintenanceWindow)

// Delete the Redshift cluster if confirmed
runner.cleanUpResources(ctx, clusterId, databaseName, tableName, user.Username,
runner.questioner)

log.Println("Thanks for watching!")
}

```

```

// cleanUpResources asks the user if they would like to delete each resource created
// during the scenario, from most
// impactful to least impactful. If any choice to delete is made, further deletion
// attempts are skipped.
func (runner *RedshiftBasicsScenario) cleanUpResources(ctx context.Context,
clusterId string, databaseName string, tableName string, userName string,
questioner demotools.IQuestioner) {
    deleted := false
    var err error = nil
    if questioner.AskBool("Do you want to delete the entire cluster? This will clean up
all resources. (y/n)", "y") {
        deleted, err = runner.redshiftActor.DeleteCluster(ctx, clusterId)
        if err != nil {
            log.Printf("Error deleting cluster: %v", err)
        }
    }
    if !deleted && questioner.AskBool("Do you want to delete the dev table? This will
clean up all inserted records but keep your cluster intact. (y/n)", "y") {
        deleted, err = runner.redshiftDataActor.DeleteTable(ctx, clusterId, databaseName,
tableName, userName)
        if err != nil {
            log.Printf("Error deleting movies table: %v", err)
        }
    }
    if !deleted && questioner.AskBool("Do you want to delete all rows in the Movies
table? This will clean up all inserted records but keep your cluster and table
intact. (y/n)", "y") {
        deleted, err = runner.redshiftDataActor.DeleteDataRows(ctx, clusterId,
databaseName, tableName, userName, runner.pauser)
        if err != nil {
            log.Printf("Error deleting data rows: %v", err)
        }
    }
    if !deleted {
        log.Print("Please manually delete any unwanted resources.")
    }
}

// loadMoviesFromJSON takes the <fileName> file and populates a slice of Movie
// objects.
func (runner *RedshiftBasicsScenario) loadMoviesFromJSON(fileName string, filesystem
demotools.IFileSystem) ([]Movie, error) {

```

```

file, err := filesystem.OpenFile("../resources/sample_files/" + fileName)
if err != nil {
    return nil, err
}
defer filesystem.CloseFile(file)

var movies []Movie
err = json.NewDecoder(file).Decode(&movies)
if err != nil {
    return nil, err
}

return movies, nil
}

// PopulateMoviesTable reads data from the <fileName> file and inserts records into
the "Movies" table.
func (runner *RedshiftBasicsScenario) PopulateMoviesTable(ctx context.Context,
clusterId string, databaseName string, tableName string, userName string, fileName
string) {
    log.Println("Populate the " + tableName + " table using the " + fileName + "
file.")
    numRecords := runner.questioner.AskInt(
        fmt.Sprintf("Enter a value between %v and %v:", 10, 100),
        demotools.InIntRange{Lower: 10, Upper: 100})

    movies, err := runner.loadMoviesFromJSON(fileName, runner.filesystem)
    if err != nil {
        log.Printf("Failed to load movies from JSON: %v\n", err)
        panic(err)
    }

    var sqlStatements []string

    for i, movie := range movies {
        if i >= numRecords {
            break
        }

        sqlStatement := fmt.Sprintf(`INSERT INTO %s (title, year) VALUES ('%s', %d);`,
            tableName,

```



```

    strings.Replace(movie.Title, "'", "''", -1), // Double any single quotes to
    escape them
    movie.Year)

    sqlStatements = append(sqlStatements, sqlStatement)
}

input := &redshiftdata.BatchExecuteStatementInput{
    ClusterIdentifier: aws.String(clusterId),
    Database:          aws.String(databaseName),
    DbUser:            aws.String(userName),
    Sqls:              sqlStatements,
}

result, err := runner.redshiftDataActor.ExecuteBatchStatement(ctx, *input)
if err != nil {
    log.Printf("Failed to execute batch statement: %v\n", err)
    panic(err)
}

describeInput := redshiftdata.DescribeStatementInput{
    Id: result.Id,
}

query := actions.RedshiftQuery{
    Context: ctx,
    Result:  result,
    Input:   describeInput,
}

err = runner.redshiftDataActor.WaitForQueryStatus(query, runner.pauser, true)
if err != nil {
    log.Printf("Failed to execute batch insert query: %v\n", err)
    return
}

log.Printf("Successfully executed batch statement\n")

log.Printf("%d records were added to the Movies table.\n", numRecords)
}

// QueryMoviesByYear retrieves only movies from the "Movies" table which match the
given year.
```

```

func (runner *RedshiftBasicsScenario) QueryMoviesByYear(ctx context.Context,
    clusterId string, databaseName string, tableName string, userName string, year int)
{

    sqlStatement := fmt.Sprintf(`SELECT title FROM %s WHERE year = %d;`, tableName,
    year)

    input := &redshiftdata.ExecuteStatementInput{
        ClusterIdentifier: aws.String(clusterId),
        Database:          aws.String(databaseName),
        DbUser:            aws.String(userName),
        Sql:               aws.String(sqlStatement),
    }

    result, err := runner.redshiftDataActor.ExecuteStatement(ctx, *input)
    if err != nil {
        log.Printf("Failed to query movies: %v\n", err)
        panic(err)
    }

    log.Println("The identifier of the statement is ", *result.Id)

    describeInput := redshiftdata.DescribeStatementInput{
        Id: result.Id,
    }

    query := actions.RedshiftQuery{
        Context: ctx,
        Input:   describeInput,
        Result:  result,
    }

    err = runner.redshiftDataActor.WaitForQueryStatus(query, runner.pauser, true)
    if err != nil {
        log.Printf("Failed to execute query: %v\n", err)
        panic(err)
    }
    log.Printf("Successfully executed query\n")

    getResultOutput, err := runner.redshiftDataActor.GetStatementResult(ctx,
    *result.Id)
    if err != nil {
        log.Printf("Failed to query movies: %v\n", err)
        panic(err)
    }
}

```

```
for _, row := range getResultOutput.Records {
    for _, col := range row {
        title, ok := col.(*redshiftdata_types.FieldMemberStringValue)
        if !ok {
            log.Println("Failed to parse the field")
        } else {
            log.Printf("The Movie title field is %s\n", title.Value)
        }
    }
}
}
```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Go .
 - [CreateCluster](#)
 - [DescribeClusters](#)
 - [DescribeStatement](#)
 - [ExecuteStatement](#)
 - [GetStatementResult](#)
 - [ListDatabasesPaginator](#)
 - [ModifyCluster](#)

Acciones

CreateCluster

En el siguiente ejemplo de código, se muestra cómo utilizar CreateCluster.

SDK para Go V2

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
import (
    "context"
    "errors"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/redshift"
    "github.com/aws/aws-sdk-go-v2/service/redshift/types"
)

// RedshiftActions wraps Redshift service actions.
type RedshiftActions struct {
    RedshiftClient *redshift.Client
}

// CreateCluster sends a request to create a cluster with the given clusterId using
// the provided credentials.
func (actor RedshiftActions) CreateCluster(ctx context.Context, clusterId string,
    userName string, userPassword string, nodeType string, clusterType string,
    publiclyAccessible bool) (*redshift.CreateClusterOutput, error) {
    // Create a new Redshift cluster
    input := &redshift.CreateClusterInput{
        ClusterIdentifier: aws.String(clusterId),
        MasterUserPassword: aws.String(userPassword),
        MasterUsername:     aws.String(userName),
        NodeType:           aws.String(nodeType),
        ClusterType:         aws.String(clusterType),
        PubliclyAccessible: aws.Bool(publiclyAccessible),
    }
    var opErr *types.ClusterAlreadyExistsFault
    output, err := actor.RedshiftClient.CreateCluster(ctx, input)
    if err != nil && errors.As(err, &opErr) {
        log.Println("Cluster already exists")
        return nil, nil
    } else if err != nil {
        log.Printf("Failed to create Redshift cluster: %v\n", err)
        return nil, err
    }
}
```

```
}

log.Printf("Created cluster %s\n", *output.Cluster.ClusterIdentifier)
return output, nil
}
```

- Para obtener más información sobre la API, consulta [CreateCluster](#) la Referencia AWS SDK para Go de la API.

DeleteCluster

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteCluster.

SDK para Go V2

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 (
    "context"
    "errors"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/redshift"
    "github.com/aws/aws-sdk-go-v2/service/redshift/types"
)

// RedshiftActions wraps Redshift service actions.
type RedshiftActions struct {
    RedshiftClient *redshift.Client
}
```

```
// DeleteCluster deletes the given cluster.
func (actor RedshiftActions) DeleteCluster(ctx context.Context, clusterId string)
(bool, error) {
    input := redshift.DeleteClusterInput{
        ClusterIdentifier:      aws.String(clusterId),
        SkipFinalClusterSnapshot: aws.Bool(true),
    }
    _, err := actor.RedshiftClient.DeleteCluster(ctx, &input)
    var opErr *types.ClusterNotFoundFault
    if err != nil && errors.As(err, &opErr) {
        log.Println("Cluster was not found. Where could it be?")
        return false, err
    } else if err != nil {
        log.Printf("Failed to delete Redshift cluster: %v\n", err)
        return false, err
    }
    waiter := redshift.NewClusterDeletedWaiter(actor.RedshiftClient)
    err = waiter.Wait(ctx, &redshift.DescribeClustersInput{
        ClusterIdentifier: aws.String(clusterId),
    }, 5*time.Minute)
    if err != nil {
        log.Printf("Wait time exceeded for deleting cluster, continuing: %v\n", err)
    }
    log.Printf("The cluster %s was deleted\n", clusterId)
    return true, nil
}
```

- Para obtener más información sobre la API, consulta [DeleteCluster](#) la Referencia AWS SDK para Go de la API.

DescribeClusters

En el siguiente ejemplo de código, se muestra cómo utilizar DescribeClusters.

SDK para Go V2

 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 (  
    "context"  
    "errors"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/redshift"  
    "github.com/aws/aws-sdk-go-v2/service/redshift/types"  
)  
  
// RedshiftActions wraps Redshift service actions.  
type RedshiftActions struct {  
    RedshiftClient *redshift.Client  
}  
  
// DescribeClusters returns information about the given cluster.  
func (actor RedshiftActions) DescribeClusters(ctx context.Context, clusterId string)  
    (*redshift.DescribeClustersOutput, error) {  
    input, err := actor.RedshiftClient.DescribeClusters(ctx,  
        &redshift.DescribeClustersInput{  
            ClusterIdentifier: aws.String(clusterId),  
        })  
    var opErr *types.AccessToClusterDeniedFault  
    if errors.As(err, &opErr) {  
        println("Access to cluster denied.")  
        panic(err)  
    } else if err != nil {  
        println("Failed to describe Redshift clusters.")  
    }
```

```
    return nil, err
}
return input, nil
}
```

- Para obtener más información sobre la API, consulta [DescribeClusters](#) la Referencia AWS SDK para Go de la API.

ModifyCluster

En el siguiente ejemplo de código, se muestra cómo utilizar `ModifyCluster`.

SDK para Go V2

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 (
    "context"
    "errors"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/redshift"
    "github.com/aws/aws-sdk-go-v2/service/redshift/types"
)

// RedshiftActions wraps Redshift service actions.
type RedshiftActions struct {
    RedshiftClient *redshift.Client
}
```



```
// ModifyCluster sets the preferred maintenance window for the given cluster.
func (actor RedshiftActions) ModifyCluster(ctx context.Context, clusterId string,
    maintenanceWindow string) *redshift.ModifyClusterOutput {
    // Modify the cluster's maintenance window
    input := &redshift.ModifyClusterInput{
        ClusterIdentifier:      aws.String(clusterId),
        PreferredMaintenanceWindow: aws.String(maintenanceWindow),
    }

    var opErr *types.InvalidClusterStateFault
    output, err := actor.RedshiftClient.ModifyCluster(ctx, input)
    if err != nil && errors.As(err, &opErr) {
        log.Println("Cluster is in an invalid state.")
        panic(err)
    } else if err != nil {
        log.Printf("Failed to modify Redshift cluster: %v\n", err)
        panic(err)
    }

    log.Printf("The cluster was successfully modified and now has %s as the maintenance window\n", *output.Cluster.PreferredMaintenanceWindow)
    return output
}
```

- Para obtener más información sobre la API, consulta [ModifyCluster](#) la Referencia AWS SDK para Go de la API.

Ejemplos de Amazon S3 usando SDK para Go V2

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar situaciones comunes mediante el uso de la AWS SDK para Go V2 con 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](#)
- [Ejemplos de tecnología sin servidor](#)

Introducción

Introducción a Amazon S3

El siguiente ejemplo de código muestra cómo empezar a utilizar Amazon S3.

SDK para Go V2

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](#).

```
package main

import (
    "context"
    "errors"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/smithy-go"
)
```

```
// main uses the AWS SDK for Go V2 to create an Amazon Simple Storage Service
// (Amazon S3) client and list up to 10 buckets in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
        fmt.Println(err)
        return
    }
    s3Client := s3.NewFromConfig(sdkConfig)
    count := 10
    fmt.Printf("Let's list up to %v buckets for your account.\n", count)
    result, err := s3Client.ListBuckets(ctx, &s3.ListBucketsInput{})
    if err != nil {
        var ae smithy.APIError
        if errors.As(err, &ae) && ae.ErrorCode() == "AccessDenied" {
            fmt.Println("You don't have permission to list buckets for this account.")
        } else {
            fmt.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
        }
        return
    }
    if len(result.Buckets) == 0 {
        fmt.Println("You don't have any buckets!")
    } else {
        if count > len(result.Buckets) {
            count = len(result.Buckets)
        }
        for _, bucket := range result.Buckets[:count] {
            fmt.Printf("\t\t%v\n", *bucket.Name)
        }
    }
}
```

- Para obtener más información sobre la API, consulta [ListBuckets](#) la Referencia AWS SDK para Go 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 Go V2

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](#).

Defina una estructura que envuelva las acciones de bucket y objeto utilizadas por el escenario.

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)
```

```
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// ListBuckets lists the buckets in the current account.
func (basics BucketBasics) ListBuckets(ctx context.Context) ([]types.Bucket, error) {
    {
        var err error
        var output *s3.ListBucketsOutput
        var buckets []types.Bucket
        bucketPaginator := s3.NewListBucketsPaginator(basics.S3Client,
            &s3.ListBucketsInput{})
        for bucketPaginator.HasMorePages() {
            output, err = bucketPaginator.NextPage(ctx)
            if err != nil {
                var apiErr smithy.APIError
                if errors.As(err, &apiErr) && apiErr.ErrorCode() == "AccessDenied" {
                    fmt.Println("You don't have permission to list buckets for this account.")
                    err = apiErr
                } else {
                    log.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
                }
                break
            } else {
                buckets = append(buckets, output.Buckets...)
            }
        }
        return buckets, err
    }
}

// BucketExists checks whether a bucket exists in the current account.
func (basics BucketBasics) BucketExists(ctx context.Context, bucketName string) (bool, error) {
    _, err := basics.S3Client.HeadBucket(ctx, &s3.HeadBucketInput{
        Bucket: aws.String(bucketName),
    })
}
```

```

exists := true
if err != nil {
    var apiError smithy.APIError
    if errors.As(err, &apiError) {
        switch apiError.(type) {
            case *types.NotFound:
                log.Printf("Bucket %v is available.\n", bucketName)
                exists = false
                err = nil
            default:
                log.Printf("Either you don't have access to bucket %v or another error occurred.
"+
                    "Here's what happened: %v\n", bucketName, err)
        }
    }
} else {
    log.Printf("Bucket %v exists and you already own it.", bucketName)
}

return exists, err
}

// CreateBucket creates a bucket with the specified name in the specified Region.
func (basics BucketBasics) CreateBucket(ctx context.Context, name string, region
string) error {
    _, err := basics.S3Client.CreateBucket(ctx, &s3.CreateBucketInput{
        Bucket: aws.String(name),
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            LocationConstraint: types.BucketLocationConstraint(region),
        },
    })
    if err != nil {
        var owned *types.BucketAlreadyOwnedByYou
        var exists *types.BucketAlreadyExists
        if errors.As(err, &owned) {
            log.Printf("You already own bucket %s.\n", name)
            err = owned
        } else if errors.As(err, &exists) {
            log.Printf("Bucket %s already exists.\n", name)
            err = exists
        }
    } else {

```

```

err = s3.NewBucketExistsWaiter(basics.S3Client).Wait(
    ctx, &s3.HeadBucketInput{Bucket: aws.String(name)}, time.Minute)
if err != nil {
    log.Printf("Failed attempt to wait for bucket %s to exist.\n", name)
}
}
return err
}

// UploadFile reads from a file and puts the data into an object in a bucket.
func (basics BucketBasics) UploadFile(ctx context.Context, bucketName string,
    objectKey string, fileName string) error {
    file, err := os.Open(fileName)
    if err != nil {
        log.Printf("Couldn't open file %v to upload. Here's why: %v\n", fileName, err)
    } else {
        defer file.Close()
        _, err = basics.S3Client.PutObject(ctx, &s3.PutObjectInput{
            Bucket: aws.String(bucketName),
            Key:     aws.String(objectKey),
            Body:    file,
        })
        if err != nil {
            var apiErr smithy.APIError
            if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
                log.Printf("Error while uploading object to %s. The object is too large.\n"+
                    "To upload objects larger than 5GB, use the S3 console (160GB max)\n"+
                    "or the multipart upload API (5TB max).", bucketName)
            } else {
                log.Printf("Couldn't upload file %v to %v:%v. Here's why: %v\n",
                    fileName, bucketName, objectKey, err)
            }
        } else {
            err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
                ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
aws.String(objectKey)}, time.Minute)
            if err != nil {
                log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
            }
        }
    }
    return err
}

```

```

}

// UploadLargeObject uses an upload manager to upload data to an object in a bucket.
// The upload manager breaks large data into parts and uploads the parts
// concurrently.
func (basics BucketBasics) UploadLargeObject(ctx context.Context, bucketName string,
    objectKey string, largeObject []byte) error {
    largeBuffer := bytes.NewReader(largeObject)
    var partMiBs int64 = 10
    uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
        u.PartSize = partMiBs * 1024 * 1024
    })
    _, err := uploader.Upload(ctx, &s3.PutObjectInput{
        Bucket: aws.String(bucketName),
        Key:     aws.String(objectKey),
        Body:    largeBuffer,
    })
    if err != nil {
        var apiErr smithy.APIError
        if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
            log.Printf("Error while uploading object to %s. The object is too large.\n"+
                "The maximum size for a multipart upload is 5TB.", bucketName)
        } else {
            log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
                bucketName, objectKey, err)
        }
    } else {
        err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
                aws.String(objectKey)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
        }
    }

    return err
}

// DownloadFile gets an object from a bucket and stores it in a local file.

```



```

func (basics BucketBasics) DownloadFile(ctx context.Context, bucketName string,
    objectKey string, fileName string) error {
    result, err := basics.S3Client.GetObject(ctx, &s3.GetObjectInput{
        Bucket: aws.String(bucketName),
        Key:     aws.String(objectKey),
    })
    if err != nil {
        var noKey *types.NoSuchKey
        if errors.As(err, &noKey) {
            log.Printf("Can't get object %s from bucket %s. No such key exists.\n",
                objectKey, bucketName)
            err = noKey
        } else {
            log.Printf("Couldn't get object %v:%v. Here's why: %v\n", bucketName, objectKey,
                err)
        }
        return err
    }
    defer result.Body.Close()
    file, err := os.Create(fileName)
    if err != nil {
        log.Printf("Couldn't create file %v. Here's why: %v\n", fileName, err)
        return err
    }
    defer file.Close()
    body, err := io.ReadAll(result.Body)
    if err != nil {
        log.Printf("Couldn't read object body from %v. Here's why: %v\n", objectKey, err)
    }
    _, err = file.Write(body)
    return err
}

```

// DownloadLargeObject uses a download manager to download an object from a bucket.
 // The download manager gets the data in parts and writes them to a buffer until all
 of

// the data has been downloaded.

```

func (basics BucketBasics) DownloadLargeObject(ctx context.Context, bucketName
    string, objectKey string) ([]byte, error) {
    var partMiBs int64 = 10
    downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader) {
        d.PartSize = partMiBs * 1024 * 1024
    })

```

```

    })
    buffer := manager.NewWriteAtBuffer([]byte{})
    _, err := downloader.Download(ctx, buffer, &s3.GetObjectInput{
        Bucket: aws.String(bucketName),
        Key:     aws.String(objectKey),
    })
    if err != nil {
        log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
            bucketName, objectKey, err)
    }
    return buffer.Bytes(), err
}

// CopyToFolder copies an object in a bucket to a subfolder in the same bucket.
func (basics BucketBasics) CopyToFolder(ctx context.Context, bucketName string,
    objectKey string, folderName string) error {
    objectDest := fmt.Sprintf("%v/%v", folderName, objectKey)
    _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
        Bucket:      aws.String(bucketName),
        CopySource:   aws.String(fmt.Sprintf("%v/%v", bucketName, objectKey)),
        Key:          aws.String(objectDest),
    })
    if err != nil {
        var notActive *types.ObjectNotInActiveTierError
        if errors.As(err, &notActive) {
            log.Printf("Couldn't copy object %s from %s because the object isn't in the
active tier.\n",
                objectKey, bucketName)
            err = notActive
        }
    } else {
        err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
aws.String(objectDest)}), time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist.\n", objectDest)
        }
    }
    return err
}

```

```
// CopyToBucket copies an object in a bucket to another bucket.
func (basics BucketBasics) CopyToBucket(ctx context.Context, sourceBucket string,
    destinationBucket string, objectKey string) error {
    _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
        Bucket:      aws.String(destinationBucket),
        CopySource:  aws.String(fmt.Sprintf("%v/%v", sourceBucket, objectKey)),
        Key:         aws.String(objectKey),
    })
    if err != nil {
        var notActive *types.ObjectNotInActiveTierError
        if errors.As(err, &notActive) {
            log.Printf("Couldn't copy object %s from %s because the object isn't in the
active tier.\n",
                objectKey, sourceBucket)
            err = notActive
        }
    } else {
        err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(destinationBucket), Key:
aws.String(objectKey)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
        }
    }
    return err
}

// ListObjects lists the objects in a bucket.
func (basics BucketBasics) ListObjects(ctx context.Context, bucketName string)
([]types.Object, error) {
    var err error
    var output *s3.ListObjectsV2Output
    input := &s3.ListObjectsV2Input{
        Bucket: aws.String(bucketName),
    }
    var objects []types.Object
    objectPaginator := s3.NewListObjectsV2Paginator(basics.S3Client, input)
    for objectPaginator.HasMorePages() {
        output, err = objectPaginator.NextPage(ctx)
        if err != nil {
            var noBucket *types.NoSuchBucket

```

```

    if errors.As(err, &noBucket) {
        log.Printf("Bucket %s does not exist.\n", bucketName)
        err = noBucket
    }
    break
} else {
    objects = append(objects, output.Contents...)
}
}
return objects, err
}

// DeleteObjects deletes a list of objects from a bucket.
func (basics BucketBasics) DeleteObjects(ctx context.Context, bucketName string,
    objectKeys []string) error {
    var objectIds []types.ObjectIdentifier
    for _, key := range objectKeys {
        objectIds = append(objectIds, types.ObjectIdentifier{Key: aws.String(key)})
    }
    output, err := basics.S3Client.DeleteObjects(ctx, &s3.DeleteObjectsInput{
        Bucket: aws.String(bucketName),
        Delete: &types.Delete{Objects: objectIds, Quiet: aws.Bool(true)},
    })
    if err != nil || len(output.Errors) > 0 {
        log.Printf("Error deleting objects from bucket %s.\n", bucketName)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucketName)
                err = noBucket
            }
        }
        if len(output.Errors) > 0 {
            for _, outErr := range output.Errors {
                log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
            }
            err = fmt.Errorf("%s", *output.Errors[0].Message)
        }
    } else {
        for _, delObjs := range output.Deleted {
            err = s3.NewObjectNotExistsWaiter(basics.S3Client).Wait(
                ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key: delObjs.Key},
                time.Minute)
        }
    }
}

```

```

    if err != nil {
        log.Printf("Failed attempt to wait for object %s to be deleted.\n",
*del0bjs.Key)
    } else {
        log.Printf("Deleted %s.\n", *del0bjs.Key)
    }
}
}
return err
}

// DeleteBucket deletes a bucket. The bucket must be empty or an error is returned.
func (basics BucketBasics) DeleteBucket(ctx context.Context, bucketName string)
error {
_, err := basics.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
    Bucket: aws.String(bucketName)})
if err != nil {
    var noBucket *types.NoSuchBucket
    if errors.As(err, &noBucket) {
        log.Printf("Bucket %s does not exist.\n", bucketName)
        err = noBucket
    } else {
        log.Printf("Couldn't delete bucket %v. Here's why: %v\n", bucketName, err)
    }
} else {
    err = s3.NewBucketNotExistsWaiter(basics.S3Client).Wait(
        ctx, &s3.HeadBucketInput{Bucket: aws.String(bucketName)}, time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for bucket %s to be deleted.\n", bucketName)
    } else {
        log.Printf("Deleted %s.\n", bucketName)
    }
}
return err
}

```

Ejecutar un escenario interactivo que muestra cómo trabajar con buckets y objetos de S3.

```

import (
    "context"
    "fmt"
    "log"
    "os"
    "strings"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
)

// RunGetStartedScenario is an interactive example that shows you how to use Amazon
// Simple Storage Service (Amazon S3) to create an S3 bucket and use it to store
// objects.
//
// 1. Create a bucket.
// 2. Upload a local file to the bucket.
// 3. Download an object to a local file.
// 4. Copy an object to a different folder in the bucket.
// 5. List objects in the bucket.
// 6. Delete all objects in the bucket.
// 7. Delete the bucket.
//
// This example creates an Amazon S3 service client from the specified sdkConfig so
// that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
// example.
// This package can be found in the `..\..\demotools` folder of this repo.
func RunGetStartedScenario(ctx context.Context, sdkConfig aws.Config, questioner
demotools.IQuestioner) {
    defer func() {
        if r := recover(); r != nil {
            log.Println("Something went wrong with the demo.")
            _, isMock := questioner.(*demotools.MockQuestioner)
            if isMock || questioner.AskBool("Do you want to see the full error message (y/
n)?", "y") {
                log.Println(r)
            }
        }
    }()
}

```

```
log.Println(strings.Repeat("-", 88))
log.Println("Welcome to the Amazon S3 getting started demo.")
log.Println(strings.Repeat("-", 88))

s3Client := s3.NewFromConfig(sdkConfig)
bucketBasics := actions.BucketBasics{S3Client: s3Client}

count := 10
log.Printf("Let's list up to %v buckets for your account:", count)
buckets, err := bucketBasics.ListBuckets(ctx)
if err != nil {
    panic(err)
}
if len(buckets) == 0 {
    log.Println("You don't have any buckets!")
} else {
    if count > len(buckets) {
        count = len(buckets)
    }
    for _, bucket := range buckets[:count] {
        log.Printf("\t\t%v\n", *bucket.Name)
    }
}

bucketName := questioner.Ask("Let's create a bucket. Enter a name for your
bucket:",
    demotools.NotEmpty{})
bucketExists, err := bucketBasics.BucketExists(ctx, bucketName)
if err != nil {
    panic(err)
}
if !bucketExists {
    err = bucketBasics.CreateBucket(ctx, bucketName, sdkConfig.Region)
    if err != nil {
        panic(err)
    } else {
        log.Println("Bucket created.")
    }
}
log.Println(strings.Repeat("-", 88))

fmt.Println("Let's upload a file to your bucket.")
smallFile := questioner.Ask("Enter the path to a file you want to upload:",
```

```

    demotools.NotEmpty{ })
    const smallKey = "doc-example-key"
    err = bucketBasics.UploadFile(ctx, bucketName, smallKey, smallFile)
    if err != nil {
        panic(err)
    }
    log.Printf("Uploaded %v as %v.\n", smallFile, smallKey)
    log.Println(strings.Repeat("-", 88))

    log.Printf("Let's download %v to a file.", smallKey)
    downloadFileName := questioner.Ask("Enter a name for the downloaded file:",
    demotools.NotEmpty{ })
    err = bucketBasics.DownloadFile(ctx, bucketName, smallKey, downloadFileName)
    if err != nil {
        panic(err)
    }
    log.Printf("File %v downloaded.", downloadFileName)
    log.Println(strings.Repeat("-", 88))

    log.Printf("Let's copy %v to a folder in the same bucket.", smallKey)
    folderName := questioner.Ask("Enter a folder name: ", demotools.NotEmpty{ })
    err = bucketBasics.CopyToFolder(ctx, bucketName, smallKey, folderName)
    if err != nil {
        panic(err)
    }
    log.Printf("Copied %v to %v/%v.\n", smallKey, folderName, smallKey)
    log.Println(strings.Repeat("-", 88))

    log.Println("Let's list the objects in your bucket.")
    questioner.Ask("Press Enter when you're ready.")
    objects, err := bucketBasics.ListObjects(ctx, bucketName)
    if err != nil {
        panic(err)
    }
    log.Printf("Found %v objects.\n", len(objects))
    var objKeys []string
    for _, object := range objects {
        objKeys = append(objKeys, *object.Key)
        log.Printf("\t%v\n", *object.Key)
    }
    log.Println(strings.Repeat("-", 88))

    if questioner.AskBool("Do you want to delete your bucket and all of its "+
    "contents? (y/n)", "y") {

```



```
log.Println("Deleting objects.")
err = bucketBasics.DeleteObjects(ctx, bucketName, objKeys)
if err != nil {
    panic(err)
}
log.Println("Deleting bucket.")
err = bucketBasics.DeleteBucket(ctx, bucketName)
if err != nil {
    panic(err)
}
log.Printf("Deleting downloaded file %v.\n", downloadFileName)
err = os.Remove(downloadFileName)
if err != nil {
    panic(err)
}
} else {
    log.Println("Okay. Don't forget to delete objects from your bucket to avoid charges.")
}
log.Println(strings.Repeat("-", 88))

log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}
```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Go .
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

Acciones

CopyObject

En el siguiente ejemplo de código, se muestra cómo utilizar CopyObject.

SDK para Go V2

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions  
// used in the examples.  
// It contains S3Client, an Amazon S3 service client that is used to perform bucket  
// and object actions.  
type BucketBasics struct {  
    S3Client *s3.Client  
}
```

```
// CopyToBucket copies an object in a bucket to another bucket.
func (basics BucketBasics) CopyToBucket(ctx context.Context, sourceBucket string,
    destinationBucket string, objectKey string) error {
    _, err := basics.S3Client.CopyObject(ctx, &s3.CopyObjectInput{
        Bucket:      aws.String(destinationBucket),
        CopySource:  aws.String(fmt.Sprintf("%v/%v", sourceBucket, objectKey)),
        Key:         aws.String(objectKey),
    })
    if err != nil {
        var notActive *types.ObjectNotInActiveTierError
        if errors.As(err, &notActive) {
            log.Printf("Couldn't copy object %s from %s because the object isn't in the
active tier.\n",
                objectKey, sourceBucket)
            err = notActive
        }
    } else {
        err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(destinationBucket), Key:
aws.String(objectKey)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
        }
    }
    return err
}
```

- Para obtener más información sobre la API, consulta [CopyObject](#) la Referencia AWS SDK para Go de la API.

CreateBucket

En el siguiente ejemplo de código, se muestra cómo utilizar CreateBucket.

SDK para Go V2

 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](#).

Cree un bucket con la configuración predeterminada.

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "io"
    "log"
    "os"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// CreateBucket creates a bucket with the specified name in the specified Region.
func (basics BucketBasics) CreateBucket(ctx context.Context, name string, region
string) error {
    _, err := basics.S3Client.CreateBucket(ctx, &s3.CreateBucketInput{
        Bucket: aws.String(name),
```

```

    CreateBucketConfiguration: &types.CreateBucketConfiguration{
        LocationConstraint: types.BucketLocationConstraint(region),
    },
})
if err != nil {
    var owned *types.BucketAlreadyOwnedByYou
    var exists *types.BucketAlreadyExists
    if errors.As(err, &owned) {
        log.Printf("You already own bucket %s.\n", name)
        err = owned
    } else if errors.As(err, &exists) {
        log.Printf("Bucket %s already exists.\n", name)
        err = exists
    }
} else {
    err = s3.NewBucketExistsWaiter(basics.S3Client).Wait(
        ctx, &s3.HeadBucketInput{Bucket: aws.String(name)}, time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for bucket %s to exist.\n", name)
    }
}
return err
}

```

Cree un bucket con el bloqueo de objetos y espere a que aparezca.

```

import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

```

```

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// CreateBucketWithLock creates a new S3 bucket with optional object locking enabled
// and waits for the bucket to exist before returning.
func (actor S3Actions) CreateBucketWithLock(ctx context.Context, bucket string,
    region string, enableObjectLock bool) (string, error) {
    input := &s3.CreateBucketInput{
        Bucket: aws.String(bucket),
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            LocationConstraint: types.BucketLocationConstraint(region),
        },
    }

    if enableObjectLock {
        input.ObjectLockEnabledForBucket = aws.Bool(true)
    }

    _, err := actor.S3Client.CreateBucket(ctx, input)
    if err != nil {
        var owned *types.BucketAlreadyOwnedByYou
        var exists *types.BucketAlreadyExists
        if errors.As(err, &owned) {
            log.Printf("You already own bucket %s.\n", bucket)
            err = owned
        } else if errors.As(err, &exists) {
            log.Printf("Bucket %s already exists.\n", bucket)
            err = exists
        }
    } else {
        err = s3.NewBucketExistsWaiter(actor.S3Client).Wait(
            ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for bucket %s to exist.\n", bucket)
        }
    }

    return bucket, err
}

```

```
}
```

- Para obtener más información sobre la API, consulta [CreateBucket](#) la Referencia AWS SDK para Go de la API.

DeleteBucket

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteBucket.

SDK para Go V2

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 (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions  
// used in the examples.  
// It contains S3Client, an Amazon S3 service client that is used to perform bucket  
// and object actions.
```

```

type BucketBasics struct {
    S3Client *s3.Client
}

// DeleteBucket deletes a bucket. The bucket must be empty or an error is returned.
func (basics BucketBasics) DeleteBucket(ctx context.Context, bucketName string)
    error {
    _, err := basics.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
        Bucket: aws.String(bucketName)})
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucketName)
            err = noBucket
        } else {
            log.Printf("Couldn't delete bucket %v. Here's why: %v\n", bucketName, err)
        }
    } else {
        err = s3.NewBucketNotExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadBucketInput{Bucket: aws.String(bucketName)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for bucket %s to be deleted.\n", bucketName)
        } else {
            log.Printf("Deleted %s.\n", bucketName)
        }
    }
    return err
}

```

- Para obtener más información sobre la API, consulta [DeleteBucket](#) la Referencia AWS SDK para Go de la API.

DeleteObject

En el siguiente ejemplo de código, se muestra cómo utilizar `DeleteObject`.

SDK para Go V2

 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 (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// S3Actions wraps S3 service actions.  
type S3Actions struct {  
    S3Client    *s3.Client  
    S3Manager   *manager.Uploader  
}  
  
// DeleteObject deletes an object from a bucket.  
func (actor S3Actions) DeleteObject(ctx context.Context, bucket string, key string,  
    versionId string, bypassGovernance bool) (bool, error) {  
    deleted := false  
    input := &s3.DeleteObjectInput{  
        Bucket: aws.String(bucket),  
        Key:     aws.String(key),  
    }  
    if versionId != "" {  
        input.VersionId = aws.String(versionId)
```

```

}
if bypassGovernance {
    input.BypassGovernanceRetention = aws.Bool(true)
}
_, err := actor.S3Client.DeleteObject(ctx, input)
if err != nil {
    var noKey *types.NoSuchKey
    var apiErr *smithy.GenericAPIError
    if errors.As(err, &noKey) {
        log.Printf("Object %s does not exist in %s.\n", key, bucket)
        err = noKey
    } else if errors.As(err, &apiErr) {
        switch apiErr.ErrorCode() {
        case "AccessDenied":
            log.Printf("Access denied: cannot delete object %s from %s.\n", key, bucket)
            err = nil
        case "InvalidArgument":
            if bypassGovernance {
                log.Printf("You cannot specify bypass governance on a bucket without lock
enabled.")
                err = nil
            }
        }
    }
} else {
    err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
        ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: aws.String(key)},
        time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for object %s in bucket %s to be deleted.\n",
key, bucket)
    } else {
        deleted = true
    }
}
return deleted, err
}

```

- Para obtener más información sobre la API, consulta [DeleteObject](#) la Referencia AWS SDK para Go de la API.

DeleteObjects

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteObjects.

SDK para Go V2

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 (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// DeleteObjects deletes a list of objects from a bucket.
func (actor S3Actions) DeleteObjects(ctx context.Context, bucket string, objects
[]types.ObjectIdentifier, bypassGovernance bool) error {
    if len(objects) == 0 {
        return nil
    }
}
```

```

input := s3.DeleteObjectsInput{
    Bucket: aws.String(bucket),
    Delete: &types.Delete{
        Objects: objects,
        Quiet:   aws.Bool(true),
    },
}
if bypassGovernance {
    input.BypassGovernanceRetention = aws.Bool(true)
}
delOut, err := actor.S3Client.DeleteObjects(ctx, &input)
if err != nil || len(delOut.Errors) > 0 {
    log.Printf("Error deleting objects from bucket %s.\n", bucket)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
    } else if len(delOut.Errors) > 0 {
        for _, outErr := range delOut.Errors {
            log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
        }
        err = fmt.Errorf("%s", *delOut.Errors[0].Message)
    }
} else {
    for _, delObjs := range delOut.Deleted {
        err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: delObjs.Key},
            time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to be deleted.\n",
                *delObjs.Key)
        } else {
            log.Printf("Deleted %s.\n", *delObjs.Key)
        }
    }
}
return err
}

```

- Para obtener más información sobre la API, consulta [DeleteObjects](#) la Referencia AWS SDK para Go de la API.

GetObject

En el siguiente ejemplo de código, se muestra cómo utilizar `GetObject`.

SDK para Go V2

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 (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions  
// used in the examples.  
// It contains S3Client, an Amazon S3 service client that is used to perform bucket  
// and object actions.  
type BucketBasics struct {  
    S3Client *s3.Client  
}
```

```
// DownloadFile gets an object from a bucket and stores it in a local file.
func (basics BucketBasics) DownloadFile(ctx context.Context, bucketName string,
    objectKey string, fileName string) error {
    result, err := basics.S3Client.GetObject(ctx, &s3.GetObjectInput{
        Bucket: aws.String(bucketName),
        Key:    aws.String(objectKey),
    })
    if err != nil {
        var noKey *types.NoSuchKey
        if errors.As(err, &noKey) {
            log.Printf("Can't get object %s from bucket %s. No such key exists.\n",
                objectKey, bucketName)
            err = noKey
        } else {
            log.Printf("Couldn't get object %v:%v. Here's why: %v\n", bucketName, objectKey,
                err)
        }
        return err
    }
    defer result.Body.Close()
    file, err := os.Create(fileName)
    if err != nil {
        log.Printf("Couldn't create file %v. Here's why: %v\n", fileName, err)
        return err
    }
    defer file.Close()
    body, err := io.ReadAll(result.Body)
    if err != nil {
        log.Printf("Couldn't read object body from %v. Here's why: %v\n", objectKey, err)
    }
    _, err = file.Write(body)
    return err
}
```

- Para obtener más información sobre la API, consulta [GetObject](#) la Referencia AWS SDK para Go de la API.

GetObjectLegalHold

En el siguiente ejemplo de código, se muestra cómo utilizar `GetObjectLegalHold`.

SDK para Go V2

 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 (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// S3Actions wraps S3 service actions.  
type S3Actions struct {  
    S3Client    *s3.Client  
    S3Manager   *manager.Uploader  
}  
  
// GetObjectLegalHold retrieves the legal hold status for an S3 object.  
func (actor S3Actions) GetObjectLegalHold(ctx context.Context, bucket string, key  
    string, versionId string) (*types.ObjectLockLegalHoldStatus, error) {  
    var status *types.ObjectLockLegalHoldStatus  
    input := &s3.GetObjectLegalHoldInput{  
        Bucket:    aws.String(bucket),  
        Key:       aws.String(key),  
        VersionId: aws.String(versionId),  
    }  
}
```

```
output, err := actor.S3Client.GetObjectLegalHold(ctx, input)
if err != nil {
    var noSuchKeyErr *types.NoSuchKey
    var apiErr *smithy.GenericAPIError
    if errors.As(err, &noSuchKeyErr) {
        log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
        err = noSuchKeyErr
    } else if errors.As(err, &apiErr) {
        switch apiErr.ErrorCode() {
        case "NoSuchObjectLockConfiguration":
            log.Printf("Object %s does not have an object lock configuration.\n", key)
            err = nil
        case "InvalidRequest":
            log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
            err = nil
        }
    }
} else {
    status = &output.LegalHold.Status
}

return status, err
}
```

- Para obtener más información sobre la API, consulta [GetObjectLegalHold](#) la Referencia AWS SDK para Go de la API.

GetObjectLockConfiguration

En el siguiente ejemplo de código, se muestra cómo utilizar `GetObjectLockConfiguration`.

SDK para Go V2

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 (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// GetObjectLockConfiguration retrieves the object lock configuration for an S3
// bucket.
func (actor S3Actions) GetObjectLockConfiguration(ctx context.Context, bucket
    string) (*types.ObjectLockConfiguration, error) {
    var lockConfig *types.ObjectLockConfiguration
    input := &s3.GetObjectLockConfigurationInput{
        Bucket: aws.String(bucket),
    }

    output, err := actor.S3Client.GetObjectLockConfiguration(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        } else if errors.As(err, &apiErr) && apiErr.ErrorCode() ==
            "ObjectLockConfigurationNotFoundError" {
            log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)

```

```
    err = nil
  }
} else {
  lockConfig = output.ObjectLockConfiguration
}

return lockConfig, err
}
```

- Para obtener más información sobre la API, consulta [GetObjectLockConfiguration](#) la Referencia AWS SDK para Go de la API.

GetObjectRetention

En el siguiente ejemplo de código, se muestra cómo utilizar `GetObjectRetention`.

SDK para Go V2

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 (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)
```

```
// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// GetObjectRetention retrieves the object retention configuration for an S3 object.
func (actor S3Actions) GetObjectRetention(ctx context.Context, bucket string, key
    string) (*types.ObjectLockRetention, error) {
    var retention *types.ObjectLockRetention
    input := &s3.GetObjectRetentionInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
    }

    output, err := actor.S3Client.GetObjectRetention(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noKey
        } else if errors.As(err, &apiErr) {
            switch apiErr.ErrorCode() {
            case "NoSuchObjectLockConfiguration":
                err = nil
            case "InvalidRequest":
                log.Printf("Bucket %s does not have locking enabled.", bucket)
                err = nil
            }
        }
    } else {
        retention = output.Retention
    }

    return retention, err
}
```

- Para obtener más información sobre la API, consulta [GetObjectRetention](#) la Referencia AWS SDK para Go de la API.

HeadBucket

En el siguiente ejemplo de código, se muestra cómo utilizar HeadBucket.

SDK para Go V2

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 (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions  
// used in the examples.  
// It contains S3Client, an Amazon S3 service client that is used to perform bucket  
// and object actions.  
type BucketBasics struct {  
    S3Client *s3.Client  
}
```

```
// BucketExists checks whether a bucket exists in the current account.
func (basics BucketBasics) BucketExists(ctx context.Context, bucketName string)
(bool, error) {
    _, err := basics.S3Client.HeadBucket(ctx, &s3.HeadBucketInput{
        Bucket: aws.String(bucketName),
    })
    exists := true
    if err != nil {
        var apiError smithy.APIError
        if errors.As(err, &apiError) {
            switch apiError.(type) {
            case *types.NotFound:
                log.Printf("Bucket %v is available.\n", bucketName)
                exists = false
                err = nil
            default:
                log.Printf("Either you don't have access to bucket %v or another error occurred.
"+
                    "Here's what happened: %v\n", bucketName, err)
            }
        }
    } else {
        log.Printf("Bucket %v exists and you already own it.", bucketName)
    }

    return exists, err
}
```

- Para obtener más información sobre la API, consulta [HeadBucket](#) la Referencia AWS SDK para Go de la API.

ListBuckets

En el siguiente ejemplo de código, se muestra cómo utilizar `ListBuckets`.

SDK para Go V2

 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 (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions  
// used in the examples.  
// It contains S3Client, an Amazon S3 service client that is used to perform bucket  
// and object actions.  
type BucketBasics struct {  
    S3Client *s3.Client  
}  
  
// ListBuckets lists the buckets in the current account.  
func (basics BucketBasics) ListBuckets(ctx context.Context) ([]types.Bucket, error)  
{  
    var err error  
    var output *s3.ListBucketsOutput  
    var buckets []types.Bucket
```

```

bucketPaginator := s3.NewListBucketsPaginator(basics.S3Client,
&s3.ListBucketsInput{})
for bucketPaginator.HasMorePages() {
    output, err = bucketPaginator.NextPage(ctx)
    if err != nil {
        var apiErr smithy.APIError
        if errors.As(err, &apiErr) && apiErr.ErrorCode() == "AccessDenied" {
            fmt.Println("You don't have permission to list buckets for this account.")
            err = apiErr
        } else {
            log.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
        }
        break
    } else {
        buckets = append(buckets, output.Buckets...)
    }
}
return buckets, err
}

```

- Para obtener más información sobre la API, consulta [ListBuckets](#) la Referencia AWS SDK para Go de la API.

ListObjectVersions

En el siguiente ejemplo de código, se muestra cómo utilizar ListObjectVersions.

SDK para Go V2

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 (
    "bytes"
    "context"

```

```

"errors"
"fmt"
"log"
"time"

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
  S3Client    *s3.Client
  S3Manager   *manager.Uploader
}

// ListObjectVersions lists all versions of all objects in a bucket.
func (actor S3Actions) ListObjectVersions(ctx context.Context, bucket string)
  ([]types.ObjectVersion, error) {
  var err error
  var output *s3.ListObjectVersionsOutput
  var versions []types.ObjectVersion
  input := &s3.ListObjectVersionsInput{Bucket: aws.String(bucket)}
  versionPaginator := s3.NewListObjectVersionsPaginator(actor.S3Client, input)
  for versionPaginator.HasMorePages() {
    output, err = versionPaginator.NextPage(ctx)
    if err != nil {
      var noBucket *types.NoSuchBucket
      if errors.As(err, &noBucket) {
        log.Printf("Bucket %s does not exist.\n", bucket)
        err = noBucket
      }
      break
    } else {
      versions = append(versions, output.Versions...)
    }
  }
  return versions, err
}

```


- Para obtener más información sobre la API, consulta [ListObjectVersions](#) la Referencia AWS SDK para Go de la API.

ListObjectsV2

En el siguiente ejemplo de código, se muestra cómo utilizar ListObjectsV2.

SDK para Go V2

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 (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions  
// used in the examples.  
// It contains S3Client, an Amazon S3 service client that is used to perform bucket  
// and object actions.  
type BucketBasics struct {  
    S3Client *s3.Client
```

```

}

// ListObjects lists the objects in a bucket.
func (basics BucketBasics) ListObjects(ctx context.Context, bucketName string)
([]types.Object, error) {
    var err error
    var output *s3.ListObjectsV2Output
    input := &s3.ListObjectsV2Input{
        Bucket: aws.String(bucketName),
    }
    var objects []types.Object
    objectPaginator := s3.NewListObjectsV2Paginator(basics.S3Client, input)
    for objectPaginator.HasMorePages() {
        output, err = objectPaginator.NextPage(ctx)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucketName)
                err = noBucket
            }
            break
        } else {
            objects = append(objects, output.Contents...)
        }
    }
    return objects, err
}

```

- Para obtener más información sobre la API, consulta la [ListObjectsversión 2](#) en la referencia de la AWS SDK para Go API.

PutObject

En el siguiente ejemplo de código, se muestra cómo utilizar PutObject.

SDK para Go V2

 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](#).

Coloque un objeto en un bucket con la API de bajo nivel.

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "io"
    "log"
    "os"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// UploadFile reads from a file and puts the data into an object in a bucket.
func (basics BucketBasics) UploadFile(ctx context.Context, bucketName string,
    objectKey string, fileName string) error {
    file, err := os.Open(fileName)
    if err != nil {
```

```

    log.Printf("Couldn't open file %v to upload. Here's why: %v\n", fileName, err)
} else {
    defer file.Close()
    _, err = basics.S3Client.PutObject(ctx, &s3.PutObjectInput{
        Bucket: aws.String(bucketName),
        Key:     aws.String(objectKey),
        Body:    file,
    })
    if err != nil {
        var apiErr smithy.APIError
        if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
            log.Printf("Error while uploading object to %s. The object is too large.\n"+
                "To upload objects larger than 5GB, use the S3 console (160GB max)\n"+
                "or the multipart upload API (5TB max).", bucketName)
        } else {
            log.Printf("Couldn't upload file %v to %v:%v. Here's why: %v\n",
                fileName, bucketName, objectKey, err)
        }
    } else {
        err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
aws.String(objectKey)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
        }
    }
}
return err
}

```

Cargue un objeto en un bucket con un administrador de transferencias.

```

import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

```

```

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// UploadObject uses the S3 upload manager to upload an object to a bucket.
func (actor S3Actions) UploadObject(ctx context.Context, bucket string, key string,
    contents string) (string, error) {
    var outKey string
    input := &s3.PutObjectInput{
        Bucket:      aws.String(bucket),
        Key:         aws.String(key),
        Body:        bytes.NewReader([]byte(contents)),
        ChecksumAlgorithm: types.ChecksumAlgorithmSha256,
    }
    output, err := actor.S3Manager.Upload(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
    } else {
        err := s3.NewObjectExistsWaiter(actor.S3Client).Wait(ctx, &s3.HeadObjectInput{
            Bucket: aws.String(bucket),
            Key:   aws.String(key),
        }, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist in %s.\n", key, bucket)
        } else {
            outKey = *output.Key
        }
    }
    return outKey, err
}

```

```
}
```

- Para obtener más información sobre la API, consulta [PutObject](#) la Referencia AWS SDK para Go de la API.

PutObjectLegalHold

En el siguiente ejemplo de código, se muestra cómo utilizar PutObjectLegalHold.

SDK para Go V2

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 (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// S3Actions wraps S3 service actions.  
type S3Actions struct {  
    S3Client    *s3.Client  
    S3Manager   *manager.Uploader  
}
```

```
// PutObjectLegalHold sets the legal hold configuration for an S3 object.
func (actor S3Actions) PutObjectLegalHold(ctx context.Context, bucket string, key
    string, versionId string, legalHoldStatus types.ObjectLockLegalHoldStatus) error {
    input := &s3.PutObjectLegalHoldInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
        LegalHold: &types.ObjectLockLegalHold{
            Status: legalHoldStatus,
        },
    }
    if versionId != "" {
        input.VersionId = aws.String(versionId)
    }

    _, err := actor.S3Client.PutObjectLegalHold(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noKey
        }
    }

    return err
}
```

- Para obtener más información sobre la API, consulta [PutObjectLegalHold](#) la Referencia AWS SDK para Go de la API.

PutObjectLockConfiguration

En el siguiente ejemplo de código, se muestra cómo utilizar PutObjectLockConfiguration.

SDK para Go V2

 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](#).

Establecimiento de la configuración de bloqueo de objetos de un bucket

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "log"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
    "github.com/aws/aws-sdk-go-v2/service/s3/types"  
    "github.com/aws/smithy-go"  
)  
  
// S3Actions wraps S3 service actions.  
type S3Actions struct {  
    S3Client    *s3.Client  
    S3Manager *manager.Uploader  
}  
  
// EnableObjectLockOnBucket enables object locking on an existing bucket.  
func (actor S3Actions) EnableObjectLockOnBucket(ctx context.Context, bucket string)  
    error {  
    // Versioning must be enabled on the bucket before object locking is enabled.  
    verInput := &s3.PutBucketVersioningInput{  
        Bucket: aws.String(bucket),  
        VersioningConfiguration: &types.VersioningConfiguration{  
            MFADelete: types.MFADeleteDisabled,  
            Status:    types.BucketVersioningStatusEnabled,  
        },  
    },  
}
```



```

    },
}
_, err := actor.S3Client.PutBucketVersioning(ctx, verInput)
if err != nil {
    var noBucket *types.NoSuchBucket
    if errors.As(err, &noBucket) {
        log.Printf("Bucket %s does not exist.\n", bucket)
        err = noBucket
    }
    return err
}

input := &s3.PutObjectLockConfigurationInput{
    Bucket: aws.String(bucket),
    ObjectLockConfiguration: &types.ObjectLockConfiguration{
        ObjectLockEnabled: types.ObjectLockEnabledEnabled,
    },
}
_, err = actor.S3Client.PutObjectLockConfiguration(ctx, input)
if err != nil {
    var noBucket *types.NoSuchBucket
    if errors.As(err, &noBucket) {
        log.Printf("Bucket %s does not exist.\n", bucket)
        err = noBucket
    }
}

return err
}

```

Establecimiento del período de retención predeterminado de un bucket

```

import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

```

```

"github.com/aws/aws-sdk-go-v2/aws"
"github.com/aws/aws-sdk-go-v2/feature/s3/manager"
"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/aws/aws-sdk-go-v2/service/s3/types"
"github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// ModifyDefaultBucketRetention modifies the default retention period of an existing
// bucket.
func (actor S3Actions) ModifyDefaultBucketRetention(
    ctx context.Context, bucket string, lockMode types.ObjectLockEnabled,
    retentionPeriod int32, retentionMode types.ObjectLockRetentionMode) error {

    input := &s3.PutObjectLockConfigurationInput{
        Bucket: aws.String(bucket),
        ObjectLockConfiguration: &types.ObjectLockConfiguration{
            ObjectLockEnabled: lockMode,
            Rule: &types.ObjectLockRule{
                DefaultRetention: &types.DefaultRetention{
                    Days: aws.Int32(retentionPeriod),
                    Mode: retentionMode,
                },
            },
        },
    }

    _, err := actor.S3Client.PutObjectLockConfiguration(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
    }

    return err
}

```

- Para obtener más información sobre la API, consulta [PutObjectLockConfiguration](#) la Referencia AWS SDK para Go de la API.

PutObjectRetention

En el siguiente ejemplo de código, se muestra cómo utilizar PutObjectRetention.

SDK para Go V2

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 (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}
```

```
// PutObjectRetention sets the object retention configuration for an S3 object.
func (actor S3Actions) PutObjectRetention(ctx context.Context, bucket string, key
string, retentionMode types.ObjectLockRetentionMode, retentionPeriodDays int32)
error {
    input := &s3.PutObjectRetentionInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
        Retention: &types.ObjectLockRetention{
            Mode:          retentionMode,
            RetainUntilDate: aws.Time(time.Now().AddDate(0, 0, int(retentionPeriodDays))),
        },
        BypassGovernanceRetention: aws.Bool(true),
    }

    _, err := actor.S3Client.PutObjectRetention(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noKey
        }
    }

    return err
}
```

- Para obtener más información sobre la API, consulta [PutObjectRetention](#) la Referencia AWS SDK para Go 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 Go V2

 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](#).

Crear funciones que encapsulen acciones prefirma de S3.

```
import (
    "context"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    v4 "github.com/aws/aws-sdk-go-v2/aws/signer/v4"
    "github.com/aws/aws-sdk-go-v2/service/s3"
)

// Presigner encapsulates the Amazon Simple Storage Service (Amazon S3) presign
// actions
// used in the examples.
// It contains PresignClient, a client that is used to presign requests to Amazon
// S3.
// Presigned requests contain temporary credentials and can be made from any HTTP
// client.
type Presigner struct {
    PresignClient *s3.PresignClient
}

// GetObject makes a presigned request that can be used to get an object from a
// bucket.
// The presigned request is valid for the specified number of seconds.
func (presigner Presigner) GetObject(
    ctx context.Context, bucketName string, objectKey string, lifetimeSecs int64)
(*v4.PresignedHTTPRequest, error) {
    request, err := presigner.PresignClient.PresignGetObject(ctx, &s3.GetObjectInput{
        Bucket: aws.String(bucketName),
        Key:    aws.String(objectKey),
```

```

    }, func(opts *s3.PresignOptions) {
        opts.Expires = time.Duration(lifetimeSecs * int64(time.Second))
    })
    if err != nil {
        log.Printf("Couldn't get a presigned request to get %v:%v. Here's why: %v\n",
            bucketName, objectKey, err)
    }
    return request, err
}

// PutObject makes a presigned request that can be used to put an object in a
// bucket.
// The presigned request is valid for the specified number of seconds.
func (presigner Presigner) PutObject(
    ctx context.Context, bucketName string, objectKey string, lifetimeSecs int64)
(*v4.PresignedHTTPRequest, error) {
    request, err := presigner.PresignClient.PresignPutObject(ctx, &s3.PutObjectInput{
        Bucket: aws.String(bucketName),
        Key:    aws.String(objectKey),
    }, func(opts *s3.PresignOptions) {
        opts.Expires = time.Duration(lifetimeSecs * int64(time.Second))
    })
    if err != nil {
        log.Printf("Couldn't get a presigned request to put %v:%v. Here's why: %v\n",
            bucketName, objectKey, err)
    }
    return request, err
}

// DeleteObject makes a presigned request that can be used to delete an object from
// a bucket.
func (presigner Presigner) DeleteObject(ctx context.Context, bucketName string,
    objectKey string) (*v4.PresignedHTTPRequest, error) {
    request, err := presigner.PresignClient.PresignDeleteObject(ctx,
        &s3.DeleteObjectInput{
            Bucket: aws.String(bucketName),
            Key:    aws.String(objectKey),
        })
    if err != nil {

```

```

    log.Printf("Couldn't get a presigned request to delete object %v. Here's why: %v\n", objectKey, err)
}
return request, err
}

func (presigner Presigner) PresignPostObject(ctx context.Context, bucketName string,
objectKey string, lifetimeSecs int64) (*s3.PresignedPostRequest, error) {
    request, err := presigner.PresignClient.PresignPostObject(ctx, &s3.PutObjectInput{
        Bucket: aws.String(bucketName),
        Key:    aws.String(objectKey),
    }, func(options *s3.PresignPostOptions) {
        options.Expires = time.Duration(lifetimeSecs) * time.Second
    })
    if err != nil {
        log.Printf("Couldn't get a presigned post request to put %v:%v. Here's why: %v\n",
            bucketName, objectKey, err)
    }
    return request, nil
}

```

Ejecute un ejemplo interactivo que genere y utilice un objeto prefirado URLs para cargar, descargar y eliminar un objeto de S3.

```

import (
    "bytes"
    "context"
    "io"
    "log"
    "mime/multipart"
    "net/http"
    "os"
    "strings"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"

```

```

"github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
)

// RunPresigningScenario is an interactive example that shows you how to get
// presigned
// HTTP requests that you can use to move data into and out of Amazon Simple Storage
// Service (Amazon S3). The presigned requests contain temporary credentials and can
// be used by an HTTP client.
//
// 1. Get a presigned request to put an object in a bucket.
// 2. Use the net/http package to use the presigned request to upload a local file
//    to the bucket.
// 3. Get a presigned request to get an object from a bucket.
// 4. Use the net/http package to use the presigned request to download the object
//    to a local file.
// 5. Get a presigned request to delete an object from a bucket.
// 6. Use the net/http package to use the presigned request to delete the object.
//
// This example creates an Amazon S3 presign client from the specified sdkConfig so
// that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
// example.
// This package can be found in the ..\..\demotools folder of this repo.
//
// It uses an IHttpRequester interface to abstract HTTP requests so they can be
// mocked
// during testing.
func RunPresigningScenario(ctx context.Context, sdkConfig aws.Config, questioner
demotools.IQuestioner, httpRequester IHttpRequester) {
defer func() {
if r := recover(); r != nil {
log.Println("Something went wrong with the demo.")
_, isMock := questioner.(*demotools.MockQuestioner)
if isMock || questioner.AskBool("Do you want to see the full error message (y/
n)?", "y") {
log.Println(r)
}
}
}()
}

```



```

log.Println(strings.Repeat("-", 88))
log.Println("Welcome to the Amazon S3 presigning demo.")
log.Println(strings.Repeat("-", 88))

s3Client := s3.NewFromConfig(sdkConfig)
bucketBasics := actions.BucketBasics{S3Client: s3Client}
presignClient := s3.NewPresignClient(s3Client)
presigner := actions.Presigner{PresignClient: presignClient}

bucketName := questioner.Ask("We'll need a bucket. Enter a name for a bucket "+
    "you own or one you want to create:", demotools.NotEmpty{})
bucketExists, err := bucketBasics.BucketExists(ctx, bucketName)
if err != nil {
    panic(err)
}
if !bucketExists {
    err = bucketBasics.CreateBucket(ctx, bucketName, sdkConfig.Region)
    if err != nil {
        panic(err)
    } else {
        log.Println("Bucket created.")
    }
}
log.Println(strings.Repeat("-", 88))

log.Printf("Let's presign a request to upload a file to your bucket.")
uploadFilename := questioner.Ask("Enter the path to a file you want to upload:",
    demotools.NotEmpty{})
uploadKey := questioner.Ask("What would you like to name the uploaded object?",
    demotools.NotEmpty{})
uploadFile, err := os.Open(uploadFilename)
if err != nil {
    panic(err)
}
defer uploadFile.Close()
presignedPutRequest, err := presigner.PutObject(ctx, bucketName, uploadKey, 60)
if err != nil {
    panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
    presignedPutRequest.Method,
    presignedPutRequest.URL)
log.Println("Using net/http to send the request...")
info, err := uploadFile.Stat()

```

```
if err != nil {
    panic(err)
}
putResponse, err := httpRequester.Put(presignedPutRequest.URL, info.Size(),
uploadFile)
if err != nil {
    panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.",
presignedPutRequest.Method,
uploadKey, putResponse.StatusCode)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's presign a request to download the object.")
questioner.Ask("Press Enter when you're ready.")
presignedGetRequest, err := presigner.GetObject(ctx, bucketName, uploadKey, 60)
if err != nil {
    panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedGetRequest.Method,
presignedGetRequest.URL)
log.Println("Using net/http to send the request...")
getResponse, err := httpRequester.Get(presignedGetRequest.URL)
if err != nil {
    panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.",
presignedGetRequest.Method,
uploadKey, getResponse.StatusCode)
defer getResponse.Body.Close()
downloadBody, err := io.ReadAll(getResponse.Body)
if err != nil {
    panic(err)
}
log.Printf("Downloaded %v bytes. Here are the first 100 of them:\n",
len(downloadBody))
log.Println(strings.Repeat("-", 88))
log.Println(string(downloadBody[:100]))
log.Println(strings.Repeat("-", 88))

log.Println("Now we'll create a new request to put the same object using a
presigned post request")
questioner.Ask("Press Enter when you're ready.")
```

```

presignPostRequest, err := presigner.PresignPostObject(ctx, bucketName, uploadKey,
60)
if err != nil {
    panic(err)
}
log.Printf("Got a presigned post request to url %v with values %v\n",
presignPostRequest.URL, presignPostRequest.Values)
log.Println("Using net/http multipart to send the request...")
uploadFile, err = os.Open(uploadFilename)
if err != nil {
    panic(err)
}
defer uploadFile.Close()
multiPartResponse, err := sendMultipartRequest(presignPostRequest.URL,
presignPostRequest.Values, uploadFile, uploadKey, httpRequester)
if err != nil {
    panic(err)
}
log.Printf("Presign post object %v with presigned URL returned %v.", uploadKey,
multiPartResponse.StatusCode)

log.Println("Let's presign a request to delete the object.")
questioner.Ask("Press Enter when you're ready.")
presignedDelRequest, err := presigner.DeleteObject(ctx, bucketName, uploadKey)
if err != nil {
    panic(err)
}
log.Printf("Got a presigned %v request to URL:\n\t%v\n",
presignedDelRequest.Method,
presignedDelRequest.URL)
log.Println("Using net/http to send the request...")
delResponse, err := httpRequester.Delete(presignedDelRequest.URL)
if err != nil {
    panic(err)
}
log.Printf("%v object %v with presigned URL returned %v.\n",
presignedDelRequest.Method,
uploadKey, delResponse.StatusCode)
log.Println(strings.Repeat("-", 88))

log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}

```

Defina un contenedor de solicitudes HTTP utilizado en el ejemplo para realizar solicitudes HTTP.

```
// IHttpRequester abstracts HTTP requests into an interface so it can be mocked
// during
// unit testing.
type IHttpRequester interface {
    Get(url string) (resp *http.Response, err error)
    Post(url, contentType string, body io.Reader) (resp *http.Response, err error)
    Put(url string, contentLength int64, body io.Reader) (resp *http.Response, err
    error)
    Delete(url string) (resp *http.Response, err error)
}

// HttpRequester uses the net/http package to make HTTP requests during the
// scenario.
type HttpRequester struct{}

func (httpReq HttpRequester) Get(url string) (resp *http.Response, err error) {
    return http.Get(url)
}

func (httpReq HttpRequester) Post(url, contentType string, body io.Reader) (resp
    *http.Response, err error) {
    postRequest, err := http.NewRequest("POST", url, body)
    if err != nil {
        return nil, err
    }
    postRequest.Header.Set("Content-Type", contentType)
    return http.DefaultClient.Do(postRequest)
}

func (httpReq HttpRequester) Put(url string, contentLength int64, body io.Reader)
    (resp *http.Response, err error) {
    putRequest, err := http.NewRequest("PUT", url, body)
    if err != nil {
        return nil, err
    }
    putRequest.ContentLength = contentLength
    return http.DefaultClient.Do(putRequest)
}

func (httpReq HttpRequester) Delete(url string) (resp *http.Response, err error) {
```

```
delRequest, err := http.NewRequest("DELETE", url, nil)
if err != nil {
    return nil, err
}
return http.DefaultClient.Do(delRequest)
}
```

Bloqueo de objetos de Amazon S3

En el siguiente ejemplo de código, se muestra cómo trabajar con características de bloqueo de objetos de S3.

SDK para Go V2

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](#).

Ejecute un escenario interactivo en el que se demuestren las características de bloqueo de objetos de Amazon S3.

```
import (
    "context"
    "fmt"
    "log"
    "strings"

    "s3_object_lock/actions"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)
```

```
// ObjectLockScenario contains the steps to run the S3 Object Lock workflow.
type ObjectLockScenario struct {
    questioner demotools.IQuestioner
    resources  Resources
    s3Actions  *actions.S3Actions
    sdkConfig  aws.Config
}

// NewObjectLockScenario constructs a new ObjectLockScenario instance.
func NewObjectLockScenario(sdkConfig aws.Config, questioner demotools.IQuestioner)
    ObjectLockScenario {
    scenario := ObjectLockScenario{
        questioner: questioner,
        resources:  Resources{},
        s3Actions:  &actions.S3Actions{S3Client: s3.NewFromConfig(sdkConfig)},
        sdkConfig:  sdkConfig,
    }
    scenario.s3Actions.S3Manager = manager.NewUploader(scenario.s3Actions.S3Client)
    scenario.resources.init(scenario.s3Actions, questioner)
    return scenario
}

type nameLocked struct {
    name  string
    locked bool
}

var createInfo = []nameLocked{
    {"standard-bucket", false},
    {"lock-bucket", true},
    {"retention-bucket", false},
}

// CreateBuckets creates the S3 buckets required for the workflow.
func (scenario *ObjectLockScenario) CreateBuckets(ctx context.Context) {
    log.Println("Let's create some S3 buckets to use for this workflow.")
    success := false
    for !success {
        prefix := scenario.questioner.Ask(
            "This example creates three buckets. Enter a prefix to name your buckets\n(remember bucket names must be globally unique):")

        for _, info := range createInfo {
            log.Println(fmt.Sprintf("%s.%s", prefix, info.name))
        }
    }
}
```

```

    bucketName, err := scenario.s3Actions.CreateBucketWithLock(ctx, fmt.Sprintf("%s.
%s", prefix, info.name), scenario.sdkConfig.Region, info.locked)
    if err != nil {
        switch err.(type) {
            case *types.BucketAlreadyExists, *types.BucketAlreadyOwnedByYou:
                log.Printf("Couldn't create bucket %s.\n", bucketName)
            default:
                panic(err)
        }
        break
    }
    scenario.resources.demoBuckets[info.name] = &DemoBucket{
        name:      bucketName,
        objectKeys: []string{},
    }
    log.Printf("Created bucket %s.\n", bucketName)
}

if len(scenario.resources.demoBuckets) < len(createInfo) {
    scenario.resources.deleteBuckets(ctx)
} else {
    success = true
}
}

log.Println("S3 buckets created.")
log.Println(strings.Repeat("-", 88))
}

// EnableLockOnBucket enables object locking on an existing bucket.
func (scenario *ObjectLockScenario) EnableLockOnBucket(ctx context.Context) {
    log.Println("\nA bucket can be configured to use object locking.")
    scenario.questioner.Ask("Press Enter to continue.")

    var err error
    bucket := scenario.resources.demoBuckets["retention-bucket"]
    err = scenario.s3Actions.EnableObjectLockOnBucket(ctx, bucket.name)
    if err != nil {
        switch err.(type) {
            case *types.NoSuchBucket:
                log.Printf("Couldn't enable object locking on bucket %s.\n", bucket.name)
            default:
                panic(err)
        }
    }
}

```

```

    } else {
        log.Printf("Object locking enabled on bucket %s.", bucket.name)
    }

    log.Println(strings.Repeat("-", 88))
}

// SetDefaultRetentionPolicy sets a default retention governance policy on a bucket.
func (scenario *ObjectLockScenario) SetDefaultRetentionPolicy(ctx context.Context) {
    log.Println("\nA bucket can be configured to use object locking with a default
    retention period.")

    bucket := scenario.resources.demoBuckets["retention-bucket"]
    retentionPeriod := scenario.questioner.AskInt("Enter the default retention period
    in days: ")
    err := scenario.s3Actions.ModifyDefaultBucketRetention(ctx,
    bucket.name, types.ObjectLockEnabledEnabled, int32(retentionPeriod),
    types.ObjectLockRetentionModeGovernance)
    if err != nil {
        switch err.(type) {
        case *types.NoSuchBucket:
            log.Printf("Couldn't configure a default retention period on bucket %s.\n",
            bucket.name)
            default:
                panic(err)
        }
    } else {
        log.Printf("Default retention policy set on bucket %s with %d day retention
        period.", bucket.name, retentionPeriod)
        bucket.retentionEnabled = true
    }

    log.Println(strings.Repeat("-", 88))
}

// UploadTestObjects uploads test objects to the S3 buckets.
func (scenario *ObjectLockScenario) UploadTestObjects(ctx context.Context) {
    log.Println("Uploading test objects to S3 buckets.")

    for _, info := range createInfo {
        bucket := scenario.resources.demoBuckets[info.name]
        for i := 0; i < 2; i++ {
            key, err := scenario.s3Actions.UploadObject(ctx, bucket.name,
            fmt.Sprintf("example-%d", i),

```



```

    fmt.Sprintf("Example object content %#d in bucket %s.", i, bucket.name))
if err != nil {
    switch err.(type) {
    case *types.NoSuchBucket:
        log.Printf("Couldn't upload %s to bucket %s.\n", key, bucket.name)
    default:
        panic(err)
    }
} else {
    log.Printf("Uploaded %s to bucket %s.\n", key, bucket.name)
    bucket.objectKeys = append(bucket.objectKeys, key)
}
}
}

scenario.questioner.Ask("Test objects uploaded. Press Enter to continue.")
log.Println(strings.Repeat("-", 88))
}

// SetObjectLockConfigurations sets object lock configurations on the test objects.
func (scenario *ObjectLockScenario) SetObjectLockConfigurations(ctx context.Context)
{
    log.Println("Now let's set object lock configurations on individual objects.")

    buckets := []*DemoBucket{scenario.resources.demoBuckets["lock-bucket"],
    scenario.resources.demoBuckets["retention-bucket"]}
    for _, bucket := range buckets {
        for index, objKey := range bucket.objectKeys {
            switch index {
            case 0:
                if scenario.questioner.AskBool(fmt.Sprintf("\nDo you want to add a legal hold to
%s in %s (y/n)? ", objKey, bucket.name), "y") {
                    err := scenario.s3Actions.PutObjectLegalHold(ctx, bucket.name, objKey, "",
types.ObjectLockLegalHoldStatusOn)
                    if err != nil {
                        switch err.(type) {
                        case *types.NoSuchKey:
                            log.Printf("Couldn't set legal hold on %s.\n", objKey)
                        default:
                            panic(err)
                        }
                    }
                } else {
                    log.Printf("Legal hold set on %s.\n", objKey)
                }
            }
        }
    }
}

```

```

    }
    case 1:
        q := fmt.Sprintf("\nDo you want to add a 1 day Governance retention period to %s
in %s?\n"+
            "Reminder: Only a user with the s3:BypassGovernanceRetention permission is able
to delete this object\n"+
            "or its bucket until the retention period has expired. (y/n) ", objKey,
bucket.name)
        if scenario.questioner.AskBool(q, "y") {
            err := scenario.s3Actions.PutObjectRetention(ctx, bucket.name, objKey,
types.ObjectLockRetentionModeGovernance, 1)
            if err != nil {
                switch err.(type) {
                case *types.NoSuchKey:
                    log.Printf("Couldn't set retention period on %s in %s.\n", objKey,
bucket.name)
                default:
                    panic(err)
                }
            } else {
                log.Printf("Retention period set to 1 for %s.", objKey)
                bucket.retentionEnabled = true
            }
        }
    }
}
}
log.Println(strings.Repeat("-", 88))
}

const (
    ListAll = iota
    DeleteObject
    DeleteRetentionObject
    OverwriteObject
    ViewRetention
    ViewLegalHold
    Finish
)

// InteractWithObjects allows the user to interact with the objects and test the
object lock configurations.
func (scenario *ObjectLockScenario) InteractWithObjects(ctx context.Context) {

```

```

log.Println("Now you can interact with the objects to explore the object lock
configurations.")
interactiveChoices := []string{
    "List all objects and buckets.",
    "Attempt to delete an object.",
    "Attempt to delete an object with retention period bypass.",
    "Attempt to overwrite a file.",
    "View the retention settings for an object.",
    "View the legal hold settings for an object.",
    "Finish the workflow."}

choice := ListAll
for choice != Finish {
    objList := scenario.GetAllObjects(ctx)
    objChoices := scenario.makeObjectChoiceList(objList)
    choice = scenario.questioner.AskChoice("Choose an action from the menu:\n",
interactiveChoices)
    switch choice {
    case ListAll:
        log.Println("The current objects in the example buckets are:")
        for _, objChoice := range objChoices {
            log.Println("\t", objChoice)
        }
    case DeleteObject, DeleteRetentionObject:
        objChoice := scenario.questioner.AskChoice("Enter the number of the object to
delete:\n", objChoices)
        obj := objList[objChoice]
        deleted, err := scenario.s3Actions.DeleteObject(ctx, obj.bucket, obj.key,
obj.versionId, choice == DeleteRetentionObject)
        if err != nil {
            switch err.(type) {
            case *types.NoSuchKey:
                log.Println("Nothing to delete.")
            default:
                panic(err)
            }
        } else if deleted {
            log.Printf("Object %s deleted.\n", obj.key)
        }
    case OverwriteObject:
        objChoice := scenario.questioner.AskChoice("Enter the number of the object to
overwrite:\n", objChoices)
        obj := objList[objChoice]

```

```

_, err := scenario.s3Actions.UploadObject(ctx, obj.bucket, obj.key,
fmt.Sprintf("New content in object %s.", obj.key))
if err != nil {
    switch err.(type) {
    case *types.NoSuchBucket:
        log.Println("Couldn't upload to nonexistent bucket.")
    default:
        panic(err)
    }
} else {
    log.Printf("Uploaded new content to object %s.\n", obj.key)
}
case ViewRetention:
    objChoice := scenario.questioner.AskChoice("Enter the number of the object to
view:\n", objChoices)
    obj := objList[objChoice]
    retention, err := scenario.s3Actions.GetObjectRetention(ctx, obj.bucket, obj.key)
    if err != nil {
        switch err.(type) {
        case *types.NoSuchKey:
            log.Printf("Can't get retention configuration for %s.\n", obj.key)
        default:
            panic(err)
        }
    } else if retention != nil {
        log.Printf("Object %s has retention mode %s until %v.\n", obj.key,
retention.Mode, retention.RetainUntilDate)
    } else {
        log.Printf("Object %s does not have object retention configured.\n", obj.key)
    }
case ViewLegalHold:
    objChoice := scenario.questioner.AskChoice("Enter the number of the object to
view:\n", objChoices)
    obj := objList[objChoice]
    legalHold, err := scenario.s3Actions.GetObjectLegalHold(ctx, obj.bucket, obj.key,
obj.versionId)
    if err != nil {
        switch err.(type) {
        case *types.NoSuchKey:
            log.Printf("Can't get legal hold configuration for %s.\n", obj.key)
        default:
            panic(err)
        }
    } else if legalHold != nil {

```

```

    log.Printf("Object %s has legal hold %v.", obj.key, *legalHold)
} else {
    log.Printf("Object %s does not have legal hold configured.", obj.key)
}
case Finish:
    log.Println("Let's clean up.")
}
log.Println(strings.Repeat("-", 88))
}
}

type BucketKeyVersionId struct {
    bucket    string
    key       string
    versionId string
}

// GetAllObjects gets the object versions in the example S3 buckets and returns them
// in a flattened list.
func (scenario *ObjectLockScenario) GetAllObjects(ctx context.Context)
[]BucketKeyVersionId {
    var objectList []BucketKeyVersionId
    for _, info := range createInfo {
        bucket := scenario.resources.demoBuckets[info.name]
        versions, err := scenario.s3Actions.ListObjectVersions(ctx, bucket.name)
        if err != nil {
            switch err.(type) {
            case *types.NoSuchBucket:
                log.Printf("Couldn't get object versions for %s.\n", bucket.name)
            default:
                panic(err)
            }
        } else {
            for _, version := range versions {
                objectList = append(objectList,
                    BucketKeyVersionId{bucket: bucket.name, key: *version.Key, versionId:
                    *version.VersionId})
            }
        }
    }
    return objectList
}

```

```

// makeObjectChoiceList makes the object version list into a list of strings that
// are displayed
// as choices.
func (scenario *ObjectLockScenario) makeObjectChoiceList(bucketObjects
    []BucketKeyVersionId) []string {
    choices := make([]string, len(bucketObjects))
    for i := 0; i < len(bucketObjects); i++ {
        choices[i] = fmt.Sprintf("%s in %s with VersionId %s.",
            bucketObjects[i].key, bucketObjects[i].bucket, bucketObjects[i].versionId)
    }
    return choices
}

// Run runs the S3 Object Lock scenario.
func (scenario *ObjectLockScenario) Run(ctx context.Context) {
    defer func() {
        if r := recover(); r != nil {
            log.Println("Something went wrong with the demo.")
            _, isMock := scenario.questioner.(*demotools.MockQuestioner)
            if isMock || scenario.questioner.AskBool("Do you want to see the full error
message (y/n)?", "y") {
                log.Println(r)
            }
            scenario.resources.Cleanup(ctx)
        }
    }()

    log.Println(strings.Repeat("-", 88))
    log.Println("Welcome to the Amazon S3 Object Lock Feature Scenario.")
    log.Println(strings.Repeat("-", 88))

    scenario.CreateBuckets(ctx)
    scenario.EnableLockOnBucket(ctx)
    scenario.SetDefaultRetentionPolicy(ctx)
    scenario.UploadTestObjects(ctx)
    scenario.SetObjectLockConfigurations(ctx)
    scenario.InteractWithObjects(ctx)

    scenario.resources.Cleanup(ctx)

    log.Println(strings.Repeat("-", 88))
    log.Println("Thanks for watching!")
    log.Println(strings.Repeat("-", 88))
}

```

Defina una estructura que envuelva las acciones de S3 utilizadas en este ejemplo.

```
import (
    "bytes"
    "context"
    "errors"
    "fmt"
    "log"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// S3Actions wraps S3 service actions.
type S3Actions struct {
    S3Client    *s3.Client
    S3Manager   *manager.Uploader
}

// CreateBucketWithLock creates a new S3 bucket with optional object locking enabled
// and waits for the bucket to exist before returning.
func (actor S3Actions) CreateBucketWithLock(ctx context.Context, bucket string,
    region string, enableObjectLock bool) (string, error) {
    input := &s3.CreateBucketInput{
        Bucket: aws.String(bucket),
        CreateBucketConfiguration: &types.CreateBucketConfiguration{
            LocationConstraint: types.BucketLocationConstraint(region),
        },
    }

    if enableObjectLock {
        input.ObjectLockEnabledForBucket = aws.Bool(true)
    }
}
```

```

_, err := actor.S3Client.CreateBucket(ctx, input)
if err != nil {
    var owned *types.BucketAlreadyOwnedByYou
    var exists *types.BucketAlreadyExists
    if errors.As(err, &owned) {
        log.Printf("You already own bucket %s.\n", bucket)
        err = owned
    } else if errors.As(err, &exists) {
        log.Printf("Bucket %s already exists.\n", bucket)
        err = exists
    }
} else {
    err = s3.NewBucketExistsWaiter(actor.S3Client).Wait(
        ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket)}, time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for bucket %s to exist.\n", bucket)
    }
}

return bucket, err
}

// GetObjectLegalHold retrieves the legal hold status for an S3 object.
func (actor S3Actions) GetObjectLegalHold(ctx context.Context, bucket string, key
string, versionId string) (*types.ObjectLockLegalHoldStatus, error) {
    var status *types.ObjectLockLegalHoldStatus
    input := &s3.GetObjectLegalHoldInput{
        Bucket:    aws.String(bucket),
        Key:       aws.String(key),
        VersionId: aws.String(versionId),
    }

    output, err := actor.S3Client.GetObjectLegalHold(ctx, input)
    if err != nil {
        var noSuchKeyErr *types.NoSuchKey
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noSuchKeyErr) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noSuchKeyErr
        } else if errors.As(err, &apiErr) {
            switch apiErr.ErrorCode() {

```



```

    case "NoSuchObjectLockConfiguration":
        log.Printf("Object %s does not have an object lock configuration.\n", key)
        err = nil
    case "InvalidRequest":
        log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
        err = nil
    }
}
} else {
    status = &output.LegalHold.Status
}

return status, err
}

// GetObjectLockConfiguration retrieves the object lock configuration for an S3
// bucket.
func (actor S3Actions) GetObjectLockConfiguration(ctx context.Context, bucket
string) (*types.ObjectLockConfiguration, error) {
    var lockConfig *types.ObjectLockConfiguration
    input := &s3.GetObjectLockConfigurationInput{
        Bucket: aws.String(bucket),
    }

    output, err := actor.S3Client.GetObjectLockConfiguration(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        } else if errors.As(err, &apiErr) && apiErr.ErrorCode() ==
"ObjectLockConfigurationNotFoundError" {
            log.Printf("Bucket %s does not have an object lock configuration.\n", bucket)
            err = nil
        }
    } else {
        lockConfig = output.ObjectLockConfiguration
    }

    return lockConfig, err
}

```

```
// GetObjectRetention retrieves the object retention configuration for an S3 object.
func (actor S3Actions) GetObjectRetention(ctx context.Context, bucket string, key
string) (*types.ObjectLockRetention, error) {
    var retention *types.ObjectLockRetention
    input := &s3.GetObjectRetentionInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
    }

    output, err := actor.S3Client.GetObjectRetention(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noKey
        } else if errors.As(err, &apiErr) {
            switch apiErr.ErrorCode() {
            case "NoSuchObjectLockConfiguration":
                err = nil
            case "InvalidRequest":
                log.Printf("Bucket %s does not have locking enabled.", bucket)
                err = nil
            }
        }
    } else {
        retention = output.Retention
    }

    return retention, err
}

// PutObjectLegalHold sets the legal hold configuration for an S3 object.
func (actor S3Actions) PutObjectLegalHold(ctx context.Context, bucket string, key
string, versionId string, legalHoldStatus types.ObjectLockLegalHoldStatus) error {
    input := &s3.PutObjectLegalHoldInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
        LegalHold: &types.ObjectLockLegalHold{

```

```

    Status: legalHoldStatus,
  },
}
if versionId != "" {
    input.VersionId = aws.String(versionId)
}

_, err := actor.S3Client.PutObjectLegalHold(ctx, input)
if err != nil {
    var noKey *types.NoSuchKey
    if errors.As(err, &noKey) {
        log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
        err = noKey
    }
}

return err
}

// ModifyDefaultBucketRetention modifies the default retention period of an existing
// bucket.
func (actor S3Actions) ModifyDefaultBucketRetention(
    ctx context.Context, bucket string, lockMode types.ObjectLockEnabled,
    retentionPeriod int32, retentionMode types.ObjectLockRetentionMode) error {

    input := &s3.PutObjectLockConfigurationInput{
        Bucket: aws.String(bucket),
        ObjectLockConfiguration: &types.ObjectLockConfiguration{
            ObjectLockEnabled: lockMode,
            Rule: &types.ObjectLockRule{
                DefaultRetention: &types.DefaultRetention{
                    Days: aws.Int32(retentionPeriod),
                    Mode: retentionMode,
                },
            },
        },
    }

    _, err := actor.S3Client.PutObjectLockConfiguration(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)

```

```

    err = noBucket
}
}

return err
}

// EnableObjectLockOnBucket enables object locking on an existing bucket.
func (actor S3Actions) EnableObjectLockOnBucket(ctx context.Context, bucket string)
error {
    // Versioning must be enabled on the bucket before object locking is enabled.
    verInput := &s3.PutBucketVersioningInput{
        Bucket: aws.String(bucket),
        VersioningConfiguration: &types.VersioningConfiguration{
            MFADelete: types.MFADeleteDisabled,
            Status:    types.BucketVersioningStatusEnabled,
        },
    }
    _, err := actor.S3Client.PutBucketVersioning(ctx, verInput)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
        return err
    }

    input := &s3.PutObjectLockConfigurationInput{
        Bucket: aws.String(bucket),
        ObjectLockConfiguration: &types.ObjectLockConfiguration{
            ObjectLockEnabled: types.ObjectLockEnabledEnabled,
        },
    }
    _, err = actor.S3Client.PutObjectLockConfiguration(ctx, input)
    if err != nil {
        var noBucket *types.NoSuchBucket
        if errors.As(err, &noBucket) {
            log.Printf("Bucket %s does not exist.\n", bucket)
            err = noBucket
        }
    }
}

```

```

    return err
}

// PutObjectRetention sets the object retention configuration for an S3 object.
func (actor S3Actions) PutObjectRetention(ctx context.Context, bucket string, key
    string, retentionMode types.ObjectLockRetentionMode, retentionPeriodDays int32)
    error {
    input := &s3.PutObjectRetentionInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
        Retention: &types.ObjectLockRetention{
            Mode:          retentionMode,
            RetainUntilDate: aws.Time(time.Now().AddDate(0, 0, int(retentionPeriodDays))),
        },
        BypassGovernanceRetention: aws.Bool(true),
    }

    _, err := actor.S3Client.PutObjectRetention(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in bucket %s.\n", key, bucket)
            err = noKey
        }
    }

    return err
}

// UploadObject uses the S3 upload manager to upload an object to a bucket.
func (actor S3Actions) UploadObject(ctx context.Context, bucket string, key string,
    contents string) (string, error) {
    var outKey string
    input := &s3.PutObjectInput{
        Bucket:      aws.String(bucket),
        Key:          aws.String(key),
        Body:         bytes.NewReader([]byte(contents)),
        ChecksumAlgorithm: types.ChecksumAlgorithmSha256,
    }

```

```

output, err := actor.S3Manager.Upload(ctx, input)
if err != nil {
    var noBucket *types.NoSuchBucket
    if errors.As(err, &noBucket) {
        log.Printf("Bucket %s does not exist.\n", bucket)
        err = noBucket
    }
} else {
    err := s3.NewObjectExistsWaiter(actor.S3Client).Wait(ctx, &s3.HeadObjectInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
    }, time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for object %s to exist in %s.\n", key, bucket)
    } else {
        outKey = *output.Key
    }
}
return outKey, err
}

```

```

// ListObjectVersions lists all versions of all objects in a bucket.
func (actor S3Actions) ListObjectVersions(ctx context.Context, bucket string)
([]types.ObjectVersion, error) {
    var err error
    var output *s3.ListObjectVersionsOutput
    var versions []types.ObjectVersion
    input := &s3.ListObjectVersionsInput{Bucket: aws.String(bucket)}
    versionPaginator := s3.NewListObjectVersionsPaginator(actor.S3Client, input)
    for versionPaginator.HasMorePages() {
        output, err = versionPaginator.NextPage(ctx)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucket)
                err = noBucket
            }
            break
        } else {
            versions = append(versions, output.Versions...)
        }
    }
}

```

```

    return versions, err
}

// DeleteObject deletes an object from a bucket.
func (actor S3Actions) DeleteObject(ctx context.Context, bucket string, key string,
    versionId string, bypassGovernance bool) (bool, error) {
    deleted := false
    input := &s3.DeleteObjectInput{
        Bucket: aws.String(bucket),
        Key:     aws.String(key),
    }
    if versionId != "" {
        input.VersionId = aws.String(versionId)
    }
    if bypassGovernance {
        input.BypassGovernanceRetention = aws.Bool(true)
    }
    _, err := actor.S3Client.DeleteObject(ctx, input)
    if err != nil {
        var noKey *types.NoSuchKey
        var apiErr *smithy.GenericAPIError
        if errors.As(err, &noKey) {
            log.Printf("Object %s does not exist in %s.\n", key, bucket)
            err = noKey
        } else if errors.As(err, &apiErr) {
            switch apiErr.ErrorCode() {
            case "AccessDenied":
                log.Printf("Access denied: cannot delete object %s from %s.\n", key, bucket)
                err = nil
            case "InvalidArgument":
                if bypassGovernance {
                    log.Printf("You cannot specify bypass governance on a bucket without lock
enabled.")
                    err = nil
                }
            }
        }
    } else {
        err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: aws.String(key)},
            time.Minute)
        if err != nil {

```

```

    log.Printf("Failed attempt to wait for object %s in bucket %s to be deleted.\n",
key, bucket)
    } else {
        deleted = true
    }
}
return deleted, err
}

// DeleteObjects deletes a list of objects from a bucket.
func (actor S3Actions) DeleteObjects(ctx context.Context, bucket string, objects
[]types.ObjectIdentifier, bypassGovernance bool) error {
    if len(objects) == 0 {
        return nil
    }

    input := s3.DeleteObjectsInput{
        Bucket: aws.String(bucket),
        Delete: &types.Delete{
            Objects: objects,
            Quiet:   aws.Bool(true),
        },
    }
    if bypassGovernance {
        input.BypassGovernanceRetention = aws.Bool(true)
    }
    delOut, err := actor.S3Client.DeleteObjects(ctx, &input)
    if err != nil || len(delOut.Errors) > 0 {
        log.Printf("Error deleting objects from bucket %s.\n", bucket)
        if err != nil {
            var noBucket *types.NoSuchBucket
            if errors.As(err, &noBucket) {
                log.Printf("Bucket %s does not exist.\n", bucket)
                err = noBucket
            }
        }
        } else if len(delOut.Errors) > 0 {
            for _, outErr := range delOut.Errors {
                log.Printf("%s: %s\n", *outErr.Key, *outErr.Message)
            }
            err = fmt.Errorf("%s", *delOut.Errors[0].Message)
        }
    } else {

```



```

for _, del0bjs := range del0ut.Deleted {
    err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
        ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: del0bjs.Key},
        time.Minute)
    if err != nil {
        log.Printf("Failed attempt to wait for object %s to be deleted.\n",
            *del0bjs.Key)
    } else {
        log.Printf("Deleted %s.\n", *del0bjs.Key)
    }
}
}
return err
}

```

Eliminación de recursos.

```

import (
    "context"
    "log"
    "s3_object_lock/actions"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// DemoBucket contains metadata for buckets used in this example.
type DemoBucket struct {
    name            string
    retentionEnabled bool
    objectKeys      []string
}

// Resources keeps track of AWS resources created during the ObjectLockScenario and
// handles
// cleanup when the scenario finishes.

```

```

type Resources struct {
    demoBuckets map[string]*DemoBucket

    s3Actions  *actions.S3Actions
    questioner demotools.IQuestioner
}

// init initializes objects in the Resources struct.
func (resources *Resources) init(s3Actions *actions.S3Actions, questioner
    demotools.IQuestioner) {
    resources.s3Actions = s3Actions
    resources.questioner = questioner
    resources.demoBuckets = map[string]*DemoBucket{}
}

// Cleanup deletes all AWS resources created during the ObjectLockScenario.
func (resources *Resources) Cleanup(ctx context.Context) {
    defer func() {
        if r := recover(); r != nil {
            log.Printf("Something went wrong during cleanup.\n%v\n", r)
            log.Println("Use the AWS Management Console to remove any remaining resources " +
                "that were created for this scenario.")
        }
    }()

    wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
    resources that were created "+
        "during this demo (y/n)?", "y")
    if !wantDelete {
        log.Println("Be sure to remove resources when you're done with them to avoid
        unexpected charges!")
        return
    }

    log.Println("Removing objects from S3 buckets and deleting buckets...")
    resources.deleteBuckets(ctx)
    //resources.deleteRetentionObjects(resources.retentionBucket,
    resources.retentionObjects)

    log.Println("Cleanup complete.")
}

// deleteBuckets empties and then deletes all buckets created during the
    ObjectLockScenario.

```

```

func (resources *Resources) deleteBuckets(ctx context.Context) {
    for _, info := range createInfo {
        bucket := resources.demoBuckets[info.name]
        resources.deleteObjects(ctx, bucket)
        _, err := resources.s3Actions.S3Client.DeleteBucket(ctx, &s3.DeleteBucketInput{
            Bucket: aws.String(bucket.name),
        })
        if err != nil {
            panic(err)
        }
    }
    for _, info := range createInfo {
        bucket := resources.demoBuckets[info.name]
        err := s3.NewBucketNotExistsWaiter(resources.s3Actions.S3Client).Wait(
            ctx, &s3.HeadBucketInput{Bucket: aws.String(bucket.name)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for bucket %s to be deleted.\n", bucket.name)
        } else {
            log.Printf("Deleted %s.\n", bucket.name)
        }
    }
    resources.demoBuckets = map[string]*DemoBucket{}
}

// deleteObjects deletes all objects in the specified bucket.
func (resources *Resources) deleteObjects(ctx context.Context, bucket *DemoBucket) {
    lockConfig, err := resources.s3Actions.GetObjectLockConfiguration(ctx, bucket.name)
    if err != nil {
        panic(err)
    }
    versions, err := resources.s3Actions.ListObjectVersions(ctx, bucket.name)
    if err != nil {
        switch err.(type) {
        case *types.NoSuchBucket:
            log.Printf("No objects to get from %s.\n", bucket.name)
        default:
            panic(err)
        }
    }
    delObjects := make([]types.ObjectIdentifier, len(versions))
    for i, version := range versions {
        if lockConfig != nil && lockConfig.ObjectLockEnabled ==
            types.ObjectLockEnabledEnabled {

```

```

    status, err := resources.s3Actions.GetObjectLegalHold(ctx, bucket.name,
*version.Key, *version.VersionId)
    if err != nil {
        switch err.(type) {
            case *types.NoSuchKey:
                log.Printf("Couldn't determine legal hold status for %s in %s.\n",
*version.Key, bucket.name)
            default:
                panic(err)
        }
    } else if status != nil && *status == types.ObjectLockLegalHoldStatusOn {
        err = resources.s3Actions.PutObjectLegalHold(ctx, bucket.name, *version.Key,
*version.VersionId, types.ObjectLockLegalHoldStatusOff)
        if err != nil {
            switch err.(type) {
                case *types.NoSuchKey:
                    log.Printf("Couldn't turn off legal hold for %s in %s.\n", *version.Key,
bucket.name)
                default:
                    panic(err)
            }
        }
    }
}
del0bjects[i] = types.ObjectIdentifier{Key: version.Key, VersionId:
version.VersionId}
}
err = resources.s3Actions.DeleteObjects(ctx, bucket.name, del0bjects,
bucket.retentionEnabled)
if err != nil {
    switch err.(type) {
        case *types.NoSuchBucket:
            log.Println("Nothing to delete.")
        default:
            panic(err)
    }
}
}

```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Go .

- [GetObjectLegalHold](#)
- [GetObjectLockConfiguration](#)
- [GetObjectRetention](#)
- [PutObjectLegalHold](#)
- [PutObjectLockConfiguration](#)
- [PutObjectRetention](#)

Cargar o descargar archivos grandes

En el siguiente ejemplo de código se muestra cómo cargar o descargar archivos grandes en y desde Amazon S3.

Para obtener información, consulte [Carga de un objeto con carga multiparte](#).

SDK para Go V2

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Crear funciones que utilicen administradores de carga y descarga para dividir los datos en partes y transferirlos simultáneamente.

```
import (  
    "bytes"  
    "context"  
    "errors"  
    "fmt"  
    "io"  
    "log"  
    "os"  
    "time"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"  
    "github.com/aws/aws-sdk-go-v2/service/s3"
```

```

    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go"
)

// BucketBasics encapsulates the Amazon Simple Storage Service (Amazon S3) actions
// used in the examples.
// It contains S3Client, an Amazon S3 service client that is used to perform bucket
// and object actions.
type BucketBasics struct {
    S3Client *s3.Client
}

// UploadLargeObject uses an upload manager to upload data to an object in a bucket.
// The upload manager breaks large data into parts and uploads the parts
// concurrently.
func (basics BucketBasics) UploadLargeObject(ctx context.Context, bucketName string,
    objectKey string, largeObject []byte) error {
    largeBuffer := bytes.NewReader(largeObject)
    var partMiBs int64 = 10
    uploader := manager.NewUploader(basics.S3Client, func(u *manager.Uploader) {
        u.PartSize = partMiBs * 1024 * 1024
    })
    _, err := uploader.Upload(ctx, &s3.PutObjectInput{
        Bucket: aws.String(bucketName),
        Key:     aws.String(objectKey),
        Body:    largeBuffer,
    })
    if err != nil {
        var apiErr smithy.APIError
        if errors.As(err, &apiErr) && apiErr.ErrorCode() == "EntityTooLarge" {
            log.Printf("Error while uploading object to %s. The object is too large.\n"+
                "The maximum size for a multipart upload is 5TB.", bucketName)
        } else {
            log.Printf("Couldn't upload large object to %v:%v. Here's why: %v\n",
                bucketName, objectKey, err)
        }
    } else {
        err = s3.NewObjectExistsWaiter(basics.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucketName), Key:
                aws.String(objectKey)}, time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to exist.\n", objectKey)
        }
    }
}

```

```

    }
}

return err
}

// DownloadLargeObject uses a download manager to download an object from a bucket.
// The download manager gets the data in parts and writes them to a buffer until all
// of
// the data has been downloaded.
func (basics BucketBasics) DownloadLargeObject(ctx context.Context, bucketName
    string, objectKey string) ([]byte, error) {
    var partMiBs int64 = 10
    downloader := manager.NewDownloader(basics.S3Client, func(d *manager.Downloader) {
        d.PartSize = partMiBs * 1024 * 1024
    })
    buffer := manager.NewWriteAtBuffer([]byte{})
    _, err := downloader.Download(ctx, buffer, &s3.GetObjectInput{
        Bucket: aws.String(bucketName),
        Key:    aws.String(objectKey),
    })
    if err != nil {
        log.Printf("Couldn't download large object from %v:%v. Here's why: %v\n",
            bucketName, objectKey, err)
    }
    return buffer.Bytes(), err
}

```

Ejecutar un escenario interactivo en el que se muestra cómo utilizar los administradores de carga y descarga en contexto.

```

import (
    "context"
    "crypto/rand"
    "log"
    "strings"

    "github.com/aws/aws-sdk-go-v2/aws"

```

```

"github.com/aws/aws-sdk-go-v2/service/s3"
"github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
"github.com/awsdocs/aws-doc-sdk-examples/gov2/s3/actions"
)

// RunLargeObjectScenario is an interactive example that shows you how to use Amazon
// Simple Storage Service (Amazon S3) to upload and download large objects.
//
// 1. Create a bucket.
// 3. Upload a large object to the bucket by using an upload manager.
// 5. Download a large object by using a download manager.
// 8. Delete all objects in the bucket.
// 9. Delete the bucket.
//
// This example creates an Amazon S3 service client from the specified sdkConfig so
// that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
// example.
// This package can be found in the ..\..\demotools folder of this repo.
func RunLargeObjectScenario(ctx context.Context, sdkConfig aws.Config, questioner
demotools.IQuestioner) {
defer func() {
if r := recover(); r != nil {
log.Println("Something went wrong with the demo.")
_, isMock := questioner.(*demotools.MockQuestioner)
if isMock || questioner.AskBool("Do you want to see the full error message (y/
n)?", "y") {
log.Println(r)
}
}
}()

log.Println(strings.Repeat("-", 88))
log.Println("Welcome to the Amazon S3 large object demo.")
log.Println(strings.Repeat("-", 88))

s3Client := s3.NewFromConfig(sdkConfig)
bucketBasics := actions.BucketBasics{S3Client: s3Client}

bucketName := questioner.Ask("Let's create a bucket. Enter a name for your
bucket:",
demotools.NotEmpty{})

```



```

bucketExists, err := bucketBasics.BucketExists(ctx, bucketName)
if err != nil {
    panic(err)
}
if !bucketExists {
    err = bucketBasics.CreateBucket(ctx, bucketName, sdkConfig.Region)
    if err != nil {
        panic(err)
    } else {
        log.Println("Bucket created.")
    }
}
log.Println(strings.Repeat("-", 88))

mibs := 30
log.Printf("Let's create a slice of %v MiB of random bytes and upload it to your
bucket. ", mibs)
questioner.Ask("Press Enter when you're ready.")
largeBytes := make([]byte, 1024*1024*mibs)
_, _ = rand.Read(largeBytes)
largeKey := "doc-example-large"
log.Println("Uploading...")
err = bucketBasics.UploadLargeObject(ctx, bucketName, largeKey, largeBytes)
if err != nil {
    panic(err)
}
log.Printf("Uploaded %v MiB object as %v", mibs, largeKey)
log.Println(strings.Repeat("-", 88))

log.Printf("Let's download the %v MiB object.", mibs)
questioner.Ask("Press Enter when you're ready.")
log.Println("Downloading...")
largeDownload, err := bucketBasics.DownloadLargeObject(ctx, bucketName, largeKey)
if err != nil {
    panic(err)
}
log.Printf("Downloaded %v bytes.", len(largeDownload))
log.Println(strings.Repeat("-", 88))

if questioner.AskBool("Do you want to delete your bucket and all of its "+
"contents? (y/n)", "y") {
    log.Println("Deleting object.")
    err = bucketBasics.DeleteObjects(ctx, bucketName, []string{largeKey})
    if err != nil {

```

```
    panic(err)
}
log.Println("Deleting bucket.")
err = bucketBasics.DeleteBucket(ctx, bucketName)
if err != nil {
    panic(err)
}
} else {
    log.Println("Okay. Don't forget to delete objects from your bucket to avoid charges.")
}
log.Println(strings.Repeat("-", 88))

log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}
```

Ejemplos de tecnología sin servidor

Invocación de una función de Lambda desde un desencadenador de Amazon S3

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento activado al cargar un objeto en un bucket de S3. La función recupera el nombre del bucket de S3 y la clave del objeto del parámetro de evento y llama a la API de Amazon S3 para recuperar y registrar el tipo de contenido del objeto.

SDK para Go V2

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos sin servidor](#).

Uso de un evento de S3 con Lambda mediante Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main
```

```
import (  
    "context"  
    "log"  
  
    "github.com/aws/aws-lambda-go/events"  
    "github.com/aws/aws-lambda-go/lambda"  
    "github.com/aws/aws-sdk-go-v2/config"  
    "github.com/aws/aws-sdk-go-v2/service/s3"  
)  
  
func handler(ctx context.Context, s3Event events.S3Event) error {  
    sdkConfig, err := config.LoadDefaultConfig(ctx)  
    if err != nil {  
        log.Printf("failed to load default config: %s", err)  
        return err  
    }  
    s3Client := s3.NewFromConfig(sdkConfig)  
  
    for _, record := range s3Event.Records {  
        bucket := record.S3.Bucket.Name  
        key := record.S3.Object.URLDecodedKey  
        headOutput, err := s3Client.HeadObject(ctx, &s3.HeadObjectInput{  
            Bucket: &bucket,  
            Key:    &key,  
        })  
        if err != nil {  
            log.Printf("error getting head of object %s/%s: %s", bucket, key, err)  
            return err  
        }  
        log.Printf("successfully retrieved %s/%s of type %s", bucket, key,  
            *headOutput.ContentType)  
    }  
  
    return nil  
}  
  
func main() {  
    lambda.Start(handler)  
}
```

Ejemplos de Amazon SNS usando SDK para Go V2

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso de la AWS SDK para Go V2 con 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](#)
- [Ejemplos de tecnología sin servidor](#)

Introducción

Hola Amazon SNS

El siguiente ejemplo de código muestra cómo empezar a utilizar Amazon SNS.

SDK para Go V2

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](#).

```
package main

import (
```

```

"context"
"fmt"
"log"

"github.com/aws/aws-sdk-go-v2/config"
"github.com/aws/aws-sdk-go-v2/service/sns"
"github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
        fmt.Println(err)
        return
    }
    snsClient := sns.NewFromConfig(sdkConfig)
    fmt.Println("Let's list the topics for your account.")
    var topics []types.Topic
    paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
    for paginator.HasMorePages() {
        output, err := paginator.NextPage(ctx)
        if err != nil {
            log.Printf("Couldn't get topics. Here's why: %v\n", err)
            break
        } else {
            topics = append(topics, output.Topics...)
        }
    }
    if len(topics) == 0 {
        fmt.Println("You don't have any topics!")
    } else {
        for _, topic := range topics {
            fmt.Printf("\t%v\n", *topic.TopicArn)
        }
    }
}

```

- Para obtener más información sobre la API, consulta [ListTopics](#) la Referencia AWS SDK para Go de la API.

Acciones

CreateTopic

En el siguiente ejemplo de código, se muestra cómo utilizar CreateTopic.

SDK para Go V2

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 (
    "context"
    "encoding/json"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/sns"
    "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
// actions
// used in the examples.
type SnsActions struct {
    SnsClient *sns.Client
}

// CreateTopic creates an Amazon SNS topic with the specified name. You can
// optionally
```

```
// specify that the topic is created as a FIFO topic and whether it uses content-
// based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(ctx context.Context, topicName string,
    isFifoTopic bool, contentBasedDeduplication bool) (string, error) {
    var topicArn string
    topicAttributes := map[string]string{}
    if isFifoTopic {
        topicAttributes["FifoTopic"] = "true"
    }
    if contentBasedDeduplication {
        topicAttributes["ContentBasedDeduplication"] = "true"
    }
    topic, err := actor.SnsClient.CreateTopic(ctx, &sns.CreateTopicInput{
        Name:      aws.String(topicName),
        Attributes: topicAttributes,
    })
    if err != nil {
        log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
    } else {
        topicArn = *topic.TopicArn
    }

    return topicArn, err
}
```

- Para obtener más información sobre la API, consulta [CreateTopic](#) la Referencia AWS SDK para Go de la API.

DeleteTopic

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteTopic.

SDK para Go V2

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 (  
    "context"  
    "encoding/json"  
    "log"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/sns"  
    "github.com/aws/aws-sdk-go-v2/service/sns/types"  
)  
  
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)  
// actions  
// used in the examples.  
type SnsActions struct {  
    SnsClient *sns.Client  
}  
  
// DeleteTopic delete an Amazon SNS topic.  
func (actor SnsActions) DeleteTopic(ctx context.Context, topicArn string) error {  
    _, err := actor.SnsClient.DeleteTopic(ctx, &sns.DeleteTopicInput{  
        TopicArn: aws.String(topicArn)})  
    if err != nil {  
        log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)  
    }  
    return err  
}
```

- Para obtener más información sobre la API, consulta [DeleteTopic](#) la Referencia AWS SDK para Go de la API.

ListTopics

En el siguiente ejemplo de código, se muestra cómo utilizar ListTopics.

SDK para Go V2

 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](#).

```
package main

import (
    "context"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/sns"
    "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
        fmt.Println(err)
        return
    }
    snsClient := sns.NewFromConfig(sdkConfig)
    fmt.Println("Let's list the topics for your account.")
    var topics []types.Topic
    paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
    for paginator.HasMorePages() {
        output, err := paginator.NextPage(ctx)
        if err != nil {
            log.Printf("Couldn't get topics. Here's why: %v\n", err)
        }
    }
}
```

```
    break
  } else {
    topics = append(topics, output.Topics...)
  }
}
if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
} else {
  for _, topic := range topics {
    fmt.Printf("\t%v\n", *topic.TopicArn)
  }
}
}
```

- Para obtener más información sobre la API, consulta [ListTopics](#) la Referencia AWS SDK para Go de la API.

Publish

En el siguiente ejemplo de código, se muestra cómo utilizar Publish.

SDK para Go V2

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 (
    "context"
    "encoding/json"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/sns"
    "github.com/aws/aws-sdk-go-v2/service/sns/types"
)
```

```

// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
// actions
// used in the examples.
type SnsActions struct {
    SnsClient *sns.Client
}

// Publish publishes a message to an Amazon SNS topic. The message is then sent to
// all
// subscribers. When the topic is a FIFO topic, the message must also contain a
// group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
// value
// filter attribute can be specified so that the message can be filtered according
// to
// a filter policy.
func (actor SnsActions) Publish(ctx context.Context, topicArn string, message
    string, groupId string, dedupId string, filterKey string, filterValue string) error
{
    publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
    aws.String(message)}
    if groupId != "" {
        publishInput.MessageGroupId = aws.String(groupId)
    }
    if dedupId != "" {
        publishInput.MessageDeduplicationId = aws.String(dedupId)
    }
    if filterKey != "" && filterValue != "" {
        publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
            filterKey: {DataType: aws.String("String"), StringValue:
            aws.String(filterValue)},
        }
    }
    _, err := actor.SnsClient.Publish(ctx, &publishInput)
    if err != nil {
        log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn, err)
    }
    return err
}

```

- Para obtener detalles sobre la API, consulte [Publish](#) en la Referencia de la API de AWS SDK para Go .

Subscribe

En el siguiente ejemplo de código, se muestra cómo utilizar `Subscribe`.

SDK para Go V2

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Suscriba una cola a un tema con filtros opcionales.

```
import (  
    "context"  
    "encoding/json"  
    "log"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/sns"  
    "github.com/aws/aws-sdk-go-v2/service/sns/types"  
)  
  
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)  
// actions  
// used in the examples.  
type SnsActions struct {  
    SnsClient *sns.Client  
}  
  
// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to an  
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter  
// policy
```

```
// so that messages are only sent to the queue when the message has the specified
// attributes.
func (actor SnsActions) SubscribeQueue(ctx context.Context, topicArn string,
queueArn string, filterMap map[string][]string) (string, error) {
    var subscriptionArn string
    var attributes map[string]string
    if filterMap != nil {
        filterBytes, err := json.Marshal(filterMap)
        if err != nil {
            log.Printf("Couldn't create filter policy, here's why: %v\n", err)
            return "", err
        }
        attributes = map[string]string{"FilterPolicy": string(filterBytes)}
    }
    output, err := actor.SnsClient.Subscribe(ctx, &sns.SubscribeInput{
        Protocol:      aws.String("sqs"),
        TopicArn:      aws.String(topicArn),
        Attributes:    attributes,
        Endpoint:      aws.String(queueArn),
        ReturnSubscriptionArn: true,
    })
    if err != nil {
        log.Printf("Couldn't subscribe queue %v to topic %v. Here's why: %v\n",
            queueArn, topicArn, err)
    } else {
        subscriptionArn = *output.SubscriptionArn
    }

    return subscriptionArn, err
}
```

- Para obtener detalles sobre la API, consulte [Subscribe](#) en la Referencia de la API de AWS SDK para Go .

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 Go V2

Note

Hay más información en GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el [Repositorio de ejemplos de código de AWS](#).

Ejecutar un escenario interactivo en un símbolo del sistema.

```
import (  
    "context"  
    "encoding/json"  
    "fmt"  
    "log"  
    "strings"  
    "topics_and_queues/actions"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/sns"  
    "github.com/aws/aws-sdk-go-v2/service/sqs"  
    "github.com/aws/aws-sdk-go-v2/service/sqs/types"  
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"  
)  
  
const FIFO_SUFFIX = ".fifo"  
const TONE_KEY = "tone"  
  
var ToneChoices = []string{"cheerful", "funny", "serious", "sincere"}  
  
// MessageBody is used to deserialize the body of a message from a JSON string.  
type MessageBody struct {  
    Message string  
}
```

```
// ScenarioRunner separates the steps of this scenario into individual functions so
// that
// they are simpler to read and understand.
type ScenarioRunner struct {
    questioner demotools.IQuestioner
    snsActor    *actions.SnsActions
    sqsActor    *actions.SqsActions
}

func (runner ScenarioRunner) CreateTopic(ctx context.Context) (string, string, bool,
bool) {
    log.Println("SNS topics can be configured as FIFO (First-In-First-Out) or standard.
\n" +
        "FIFO topics deliver messages in order and support deduplication and message
filtering.")
    isFifoTopic := runner.questioner.AskBool("\nWould you like to work with FIFO
topics? (y/n) ", "y")

    contentBasedDeduplication := false
    if isFifoTopic {
        log.Println(strings.Repeat("-", 88))
        log.Println("Because you have chosen a FIFO topic, deduplication is supported.\n"
+
            "Deduplication IDs are either set in the message or are automatically generated
\n" +
            "from content using a hash function. If a message is successfully published to\n"
+
            "an SNS FIFO topic, any message published and determined to have the same\n" +
            "deduplication ID, within the five-minute deduplication interval, is accepted\n"
+
            "but not delivered. For more information about deduplication, see:\n" +
            "\thttps://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.")
        contentBasedDeduplication = runner.questioner.AskBool(
            "\nDo you want to use content-based deduplication instead of entering a
deduplication ID? (y/n) ", "y")
    }
    log.Println(strings.Repeat("-", 88))

    topicName := runner.questioner.Ask("Enter a name for your SNS topic. ")
    if isFifoTopic {
        topicName = fmt.Sprintf("%v%v", topicName, FIFO_SUFFIX)
        log.Printf("Because you have selected a FIFO topic, '%v' must be appended to\n"+
            "the topic name.", FIFO_SUFFIX)
    }
}
```

```

topicArn, err := runner.snsActor.CreateTopic(ctx, topicName, isFifoTopic,
contentBasedDeduplication)
if err != nil {
    panic(err)
}
log.Printf("Your new topic with the name '%v' and Amazon Resource Name (ARN) \n"+
    "'%v' has been created.", topicName, topicArn)

return topicName, topicArn, isFifoTopic, contentBasedDeduplication
}

func (runner ScenarioRunner) CreateQueue(ctx context.Context, ordinal string,
isFifoTopic bool) (string, string) {
    queueName := runner.questioner.Ask(fmt.Sprintf("Enter a name for the %v SQS queue.
", ordinal))
    if isFifoTopic {
        queueName = fmt.Sprintf("%v%v", queueName, FIFO_SUFFIX)
        if ordinal == "first" {
            log.Printf("Because you are creating a FIFO SQS queue, '%v' must "+
                "be appended to the queue name.\n", FIFO_SUFFIX)
        }
    }
    queueUrl, err := runner.sqsActor.CreateQueue(ctx, queueName, isFifoTopic)
    if err != nil {
        panic(err)
    }
    log.Printf("Your new SQS queue with the name '%v' and the queue URL "+
        "'%v' has been created.", queueName, queueUrl)

    return queueName, queueUrl
}

func (runner ScenarioRunner) SubscribeQueueToTopic(
    ctx context.Context, queueName string, queueUrl string, topicName string, topicArn
    string, ordinal string,
    isFifoTopic bool) (string, bool) {

    queueArn, err := runner.sqsActor.GetQueueArn(ctx, queueUrl)
    if err != nil {
        panic(err)
    }
    log.Printf("The ARN of your queue is: %v.\n", queueArn)

```



```

err = runner.sqsActor.AttachSendMessagePolicy(ctx, queueUrl, queueArn, topicArn)
if err != nil {
    panic(err)
}
log.Println("Attached an IAM policy to the queue so the SNS topic can send " +
    "messages to it.")
log.Println(strings.Repeat("-", 88))

var filterPolicy map[string][]string
if isFifoTopic {
    if ordinal == "first" {
        log.Println("Subscriptions to a FIFO topic can have filters.\n" +
            "If you add a filter to this subscription, then only the filtered messages\n" +
            "will be received in the queue.\n" +
            "For information about message filtering, see\n" +
            "\thttps://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n" +
            "For this example, you can filter messages by a \"tone\" attribute.")
    }

    wantFiltering := runner.questioner.AskBool(
        fmt.Sprintf("Do you want to filter messages that are sent to \"%v\"\n"+
            "from the %v topic? (y/n) ", queueName, topicName), "y")
    if wantFiltering {
        log.Println("You can filter messages by one or more of the following \"tone\"
attributes.")

        var toneSelections []string
        askAboutTones := true
        for askAboutTones {
            toneIndex := runner.questioner.AskChoice(
                "Enter the number of the tone you want to filter by:\n", ToneChoices)
            toneSelections = append(toneSelections, ToneChoices[toneIndex])
            askAboutTones = runner.questioner.AskBool("Do you want to add another tone to
the filter? (y/n) ", "y")
        }
        log.Printf("Your subscription will be filtered to only pass the following tones:
%v\n", toneSelections)
        filterPolicy = map[string][]string{TONE_KEY: toneSelections}
    }
}

subscriptionArn, err := runner.snsActor.SubscribeQueue(ctx, topicArn, queueArn,
filterPolicy)
if err != nil {

```

```

    panic(err)
}
log.Printf("The queue %v is now subscribed to the topic %v with the subscription
ARN %v.\n",
    queueName, topicName, subscriptionArn)

return subscriptionArn, filterPolicy != nil
}

func (runner ScenarioRunner) PublishMessages(ctx context.Context, topicArn string,
    isFifoTopic bool, contentBasedDeduplication bool, usingFilters bool) {
    var message string
    var groupId string
    var dedupId string
    var toneSelection string
    publishMore := true
    for publishMore {
        groupId = ""
        dedupId = ""
        toneSelection = ""
        message = runner.questioner.Ask("Enter a message to publish: ")
        if isFifoTopic {
            log.Println("Because you are using a FIFO topic, you must set a message group ID.
\n" +
                "All messages within the same group will be received in the order they were
published.")
            groupId = runner.questioner.Ask("Enter a message group ID: ")
            if !contentBasedDeduplication {
                log.Println("Because you are not using content-based deduplication,\n" +
                    "you must enter a deduplication ID.")
                dedupId = runner.questioner.Ask("Enter a deduplication ID: ")
            }
        }
        if usingFilters {
            if runner.questioner.AskBool("Add a tone attribute so this message can be
filtered? (y/n) ", "y") {
                toneIndex := runner.questioner.AskChoice(
                    "Enter the number of the tone you want to filter by:\n", ToneChoices)
                toneSelection = ToneChoices[toneIndex]
            }
        }

        err := runner.snsActor.Publish(ctx, topicArn, message, groupId, dedupId, TONE_KEY,
            toneSelection)
    }
}

```

```

    if err != nil {
        panic(err)
    }
    log.Println("Your message was published.")

    publishMore = runner.questioner.AskBool("Do you want to publish another message?
(y/n) ", "y")
}
}

func (runner ScenarioRunner) PollForMessages(ctx context.Context, queueUrls
[]string) {
    log.Println("Polling queues for messages...")
    for _, queueUrl := range queueUrls {
        var messages []types.Message
        for {
            currentMsgs, err := runner.sqsActor.GetMessages(ctx, queueUrl, 10, 1)
            if err != nil {
                panic(err)
            }
            if len(currentMsgs) == 0 {
                break
            }
            messages = append(messages, currentMsgs...)
        }
        if len(messages) == 0 {
            log.Printf("No messages were received by queue %v.\n", queueUrl)
        } else if len(messages) == 1 {
            log.Printf("One message was received by queue %v:\n", queueUrl)

        } else {
            log.Printf("%v messages were received by queue %v:\n", len(messages), queueUrl)
        }
        for msgIndex, message := range messages {
            messageBody := MessageBody{}
            err := json.Unmarshal([]byte(*message.Body), &messageBody)
            if err != nil {
                panic(err)
            }
            log.Printf("Message %v: %v\n", msgIndex+1, messageBody.Message)
        }

        if len(messages) > 0 {
            log.Printf("Deleting %v messages from queue %v.\n", len(messages), queueUrl)

```

```

    err := runner.sqsActor.DeleteMessages(ctx, queueUrl, messages)
    if err != nil {
        panic(err)
    }
}
}
}

// RunTopicsAndQueuesScenario is an interactive example that shows you how to use
// the
// AWS SDK for Go to create and use Amazon SNS topics and Amazon SQS queues.
//
// 1. Create a topic (FIFO or non-FIFO).
// 2. Subscribe several queues to the topic with an option to apply a filter.
// 3. Publish messages to the topic.
// 4. Poll the queues for messages received.
// 5. Delete the topic and the queues.
//
// This example creates service clients from the specified sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
// example.
// This package can be found in the ../..demotools folder of this repo.
func RunTopicsAndQueuesScenario(
    ctx context.Context, sdkConfig aws.Config, questioner demotools.IQuestioner) {
    resources := Resources{}
    defer func() {
        if r := recover(); r != nil {
            log.Println("Something went wrong with the demo.\n" +
                "Cleaning up any resources that were created...")
            resources.Cleanup(ctx)
        }
    }()
    queueCount := 2

    log.Println(strings.Repeat("-", 88))
    log.Printf("Welcome to messaging with topics and queues.\n\n"+
        "In this scenario, you will create an SNS topic and subscribe %v SQS queues to the\n"+
        "topic. You can select from several options for configuring the topic and the\n"+
        "subscriptions for the queues. You can then post to the topic and see the results\n"+
        "in the queues.\n", queueCount)

```

```

log.Println(strings.Repeat("-", 88))

runner := ScenarioRunner{
    questioner: questioner,
    snsActor:   &actions.SnsActions{SnsClient: sns.NewFromConfig(sdkConfig)},
    sqsActor:   &actions.SqsActions{SqsClient: sqs.NewFromConfig(sdkConfig)},
}
resources.snsActor = runner.snsActor
resources.sqsActor = runner.sqsActor

topicName, topicArn, isFifoTopic, contentBasedDeduplication :=
runner.CreateTopic(ctx)
resources.topicArn = topicArn
log.Println(strings.Repeat("-", 88))

log.Printf("Now you will create %v SQS queues and subscribe them to the topic.\n",
queueCount)
ordinals := []string{"first", "next"}
usingFilters := false
for _, ordinal := range ordinals {
    queueName, queueUrl := runner.CreateQueue(ctx, ordinal, isFifoTopic)
    resources.queueUrls = append(resources.queueUrls, queueUrl)

    _, filtering := runner.SubscribeQueueToTopic(ctx, queueName, queueUrl, topicName,
topicArn, ordinal, isFifoTopic)
    usingFilters = usingFilters || filtering
}

log.Println(strings.Repeat("-", 88))
runner.PublishMessages(ctx, topicArn, isFifoTopic, contentBasedDeduplication,
usingFilters)
log.Println(strings.Repeat("-", 88))
runner.PollForMessages(ctx, resources.queueUrls)

log.Println(strings.Repeat("-", 88))

wantCleanup := questioner.AskBool("Do you want to remove all AWS resources created
for this scenario? (y/n) ", "y")
if wantCleanup {
    log.Println("Cleaning up resources...")
    resources.Cleanup(ctx)
}

```

```

log.Println(strings.Repeat("-", 88))
log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}

```

Defina una estructura que encapsule las acciones de Amazon SNS utilizadas en este ejemplo.

```

import (
    "context"
    "encoding/json"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/sns"
    "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
// actions
// used in the examples.
type SnsActions struct {
    SnsClient *sns.Client
}

// CreateTopic creates an Amazon SNS topic with the specified name. You can
// optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
// based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(ctx context.Context, topicName string,
    isFifoTopic bool, contentBasedDeduplication bool) (string, error) {
    var topicArn string
    topicAttributes := map[string]string{}
    if isFifoTopic {
        topicAttributes["FifoTopic"] = "true"
    }
    if contentBasedDeduplication {
        topicAttributes["ContentBasedDeduplication"] = "true"
    }
}

```

```

}
topic, err := actor.SnsClient.CreateTopic(ctx, &sns.CreateTopicInput{
    Name:      aws.String(topicName),
    Attributes: topicAttributes,
})
if err != nil {
    log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
} else {
    topicArn = *topic.TopicArn
}

return topicArn, err
}

// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(ctx context.Context, topicArn string) error {
    _, err := actor.SnsClient.DeleteTopic(ctx, &sns.DeleteTopicInput{
        TopicArn: aws.String(topicArn)})
    if err != nil {
        log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
    }
    return err
}

// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
// policy
// so that messages are only sent to the queue when the message has the specified
// attributes.
func (actor SnsActions) SubscribeQueue(ctx context.Context, topicArn string,
    queueArn string, filterMap map[string][]string) (string, error) {
    var subscriptionArn string
    var attributes map[string]string
    if filterMap != nil {
        filterBytes, err := json.Marshal(filterMap)
        if err != nil {
            log.Printf("Couldn't create filter policy, here's why: %v\n", err)
            return "", err
        }
        attributes = map[string]string{"FilterPolicy": string(filterBytes)}
    }

```

```

}
output, err := actor.SnsClient.Subscribe(ctx, &sns.SubscribeInput{
    Protocol:          aws.String("sqs"),
    TopicArn:          aws.String(topicArn),
    Attributes:         attributes,
    Endpoint:           aws.String(queueArn),
    ReturnSubscriptionArn: true,
})
if err != nil {
    log.Printf("Couldn't subscribe queue %v to topic %v. Here's why: %v\n",
        queueArn, topicArn, err)
} else {
    subscriptionArn = *output.SubscriptionArn
}

return subscriptionArn, err
}

// Publish publishes a message to an Amazon SNS topic. The message is then sent to
// all
// subscribers. When the topic is a FIFO topic, the message must also contain a
// group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
// value
// filter attribute can be specified so that the message can be filtered according
// to
// a filter policy.
func (actor SnsActions) Publish(ctx context.Context, topicArn string, message
    string, groupId string, dedupId string, filterKey string, filterValue string) error
{
    publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
    aws.String(message)}
    if groupId != "" {
        publishInput.MessageGroupId = aws.String(groupId)
    }
    if dedupId != "" {
        publishInput.MessageDeduplicationId = aws.String(dedupId)
    }
    if filterKey != "" && filterValue != "" {
        publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
            filterKey: {DataType: aws.String("String"), StringValue:
            aws.String(filterValue)},

```



```

    }
}
_, err := actor.SnsClient.Publish(ctx, &publishInput)
if err != nil {
    log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn, err)
}
return err
}

```

Defina una estructura que encapsule las acciones de Amazon SQS utilizadas en este ejemplo.

```

import (
    "context"
    "encoding/json"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
    "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
    SqsClient *sqs.Client
}

// CreateQueue creates an Amazon SQS queue with the specified name. You can specify
// whether the queue is created as a FIFO queue.
func (actor SqsActions) CreateQueue(ctx context.Context, queueName string,
    isFifoQueue bool) (string, error) {
    var queueUrl string
    queueAttributes := map[string]string{}
    if isFifoQueue {
        queueAttributes["FifoQueue"] = "true"
    }
    queue, err := actor.SqsClient.CreateQueue(ctx, &sqs.CreateQueueInput{

```

```

    QueueName: aws.String(queueName),
    Attributes: queueAttributes,
})
if err != nil {
    log.Printf("Couldn't create queue %v. Here's why: %v\n", queueName, err)
} else {
    queueUrl = *queue.QueueUrl
}

return queueUrl, err
}

// GetQueueArn uses the GetQueueAttributes action to get the Amazon Resource Name
// (ARN)
// of an Amazon SQS queue.
func (actor SqsActions) GetQueueArn(ctx context.Context, queueUrl string) (string,
error) {
    var queueArn string
    arnAttributeName := types.QueueAttributeNameQueueArn
    attribute, err := actor.SqsClient.GetQueueAttributes(ctx,
&sqs.GetQueueAttributesInput{
    QueueUrl:      aws.String(queueUrl),
    AttributeNames: []types.QueueAttributeName{arnAttributeName},
})
    if err != nil {
        log.Printf("Couldn't get ARN for queue %v. Here's why: %v\n", queueUrl, err)
    } else {
        queueArn = attribute.Attributes[string(arnAttributeName)]
    }
    return queueArn, err
}

// AttachSendMessagePolicy uses the SetQueueAttributes action to attach a policy to
// an
// Amazon SQS queue that allows the specified Amazon SNS topic to send messages to
// the
// queue.
func (actor SqsActions) AttachSendMessagePolicy(ctx context.Context, queueUrl
string, queueArn string, topicArn string) error {
    policyDoc := PolicyDocument{

```

```

    Version: "2012-10-17",
    Statement: []PolicyStatement{{
        Effect:    "Allow",
        Action:    "sqs:SendMessage",
        Principal: map[string]string{"Service": "sns.amazonaws.com"},
        Resource:  aws.String(queueArn),
        Condition: PolicyCondition{"ArnEquals": map[string]string{"aws:SourceArn":
topicArn}},
    }},
}
policyBytes, err := json.Marshal(policyDoc)
if err != nil {
    log.Printf("Couldn't create policy document. Here's why: %v\n", err)
    return err
}
_, err = actor.SqsClient.SetQueueAttributes(ctx, &sqs.SetQueueAttributesInput{
    Attributes: map[string]string{
        string(types.QueueAttributeNamePolicy): string(policyBytes),
    },
    QueueUrl: aws.String(queueUrl),
})
if err != nil {
    log.Printf("Couldn't set send message policy on queue %v. Here's why: %v\n",
queueUrl, err)
}
return err
}

// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
    Version    string
    Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
    Effect    string
    Action    string
    Principal map[string]string `json:",omitempty"`
    Resource  *string             `json:",omitempty"`
    Condition PolicyCondition     `json:",omitempty"`
}

```

```
// PolicyCondition defines a condition in a policy.
type PolicyCondition map[string]map[string]string

// GetMessages uses the ReceiveMessage action to get messages from an Amazon SQS
// queue.
func (actor SqsActions) GetMessages(ctx context.Context, queueUrl string,
    maxMessages int32, waitTime int32) ([]types.Message, error) {
    var messages []types.Message
    result, err := actor.SqsClient.ReceiveMessage(ctx, &sqs.ReceiveMessageInput{
        QueueUrl:      aws.String(queueUrl),
        MaxNumberOfMessages: maxMessages,
        WaitTimeSeconds: waitTime,
    })
    if err != nil {
        log.Printf("Couldn't get messages from queue %v. Here's why: %v\n", queueUrl, err)
    } else {
        messages = result.Messages
    }
    return messages, err
}

// DeleteMessages uses the DeleteMessageBatch action to delete a batch of messages
// from
// an Amazon SQS queue.
func (actor SqsActions) DeleteMessages(ctx context.Context, queueUrl string,
    messages []types.Message) error {
    entries := make([]types.DeleteMessageBatchRequestEntry, len(messages))
    for msgIndex := range messages {
        entries[msgIndex].Id = aws.String(fmt.Sprintf("%v", msgIndex))
        entries[msgIndex].ReceiptHandle = messages[msgIndex].ReceiptHandle
    }
    _, err := actor.SqsClient.DeleteMessageBatch(ctx, &sqs.DeleteMessageBatchInput{
        Entries: entries,
        QueueUrl: aws.String(queueUrl),
    })
    if err != nil {
        log.Printf("Couldn't delete messages from queue %v. Here's why: %v\n", queueUrl, err)
    }
    return err
}
```

```

}

// DeleteQueue deletes an Amazon SQS queue.
func (actor SqsActions) DeleteQueue(ctx context.Context, queueUrl string) error {
    _, err := actor.SqsClient.DeleteQueue(ctx, &sqs.DeleteQueueInput{
        QueueUrl: aws.String(queueUrl)})
    if err != nil {
        log.Printf("Couldn't delete queue %v. Here's why: %v\n", queueUrl, err)
    }
    return err
}

```

Eliminación de recursos.

```

import (
    "context"
    "fmt"
    "log"
    "topics_and_queues/actions"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
    topicArn  string
    queueUrls []string
    snsActor  *actions.SnsActions
    sqsActor  *actions.SqsActions
}

// Cleanup deletes all AWS resources created during an example.
func (resources Resources) Cleanup(ctx context.Context) {
    defer func() {
        if r := recover(); r != nil {
            fmt.Println("Something went wrong during cleanup. Use the AWS Management Console\n" +
                "to remove any remaining resources that were created for this scenario.")
        }
    }()
}

```

```
}{()

var err error
if resources.topicArn != "" {
    log.Printf("Deleting topic %v.\n", resources.topicArn)
    err = resources.snsActor.DeleteTopic(ctx, resources.topicArn)
    if err != nil {
        panic(err)
    }
}

for _, queueUrl := range resources.queueUrls {
    log.Printf("Deleting queue %v.\n", queueUrl)
    err = resources.sqsActor.DeleteQueue(ctx, queueUrl)
    if err != nil {
        panic(err)
    }
}
}
```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Go .
 - [CreateQueue](#)
 - [CreateTopic](#)
 - [DeleteMessageBatch](#)
 - [DeleteQueue](#)
 - [DeleteTopic](#)
 - [GetQueueAttributes](#)
 - [Publish](#)
 - [ReceiveMessage](#)
 - [SetQueueAttributes](#)
 - [Subscribe](#)
 - [Unsubscribe](#)

Ejemplos de tecnología sin servidor

Invocación de una función de Lambda desde un desencadenador de Amazon SNS

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento activado al recibir mensajes de un tema de SNS. La función recupera los mensajes del parámetro de eventos y registra el contenido de cada mensaje.

SDK para Go V2

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Uso de un evento de SNS con Lambda mediante Go

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
    "context"
    "fmt"

    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, snsEvent events.SNSEvent) {
    for _, record := range snsEvent.Records {
        processMessage(record)
    }
    fmt.Println("done")
}

func processMessage(record events.SNSEventRecord) {
    message := record.SNS.Message
    fmt.Printf("Processed message: %s\n", message)
    // TODO: Process your record here
}
```

```
func main() {  
    lambda.Start(handler)  
}
```

Ejemplos de Amazon SQS usando SDK para Go V2

Los siguientes ejemplos de código muestran cómo realizar acciones e implementar escenarios comunes mediante el uso de la AWS SDK para Go V2 con 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](#)
- [Ejemplos de tecnología sin servidor](#)

Introducción

Introducción a Amazon SQS

El siguiente ejemplo de código muestra cómo empezar a utilizar Amazon SQS.

SDK para Go V2

 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](#).

```
package main

import (
    "context"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Queue Service
// (Amazon SQS) client and list the queues in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
        fmt.Println(err)
        return
    }
    sqsClient := sqs.NewFromConfig(sdkConfig)
    fmt.Println("Let's list the queues for your account.")
    var queueUrls []string
    paginator := sqs.NewListQueuesPaginator(sqsClient, &sqs.ListQueuesInput{})
    for paginator.HasMorePages() {
        output, err := paginator.NextPage(ctx)
        if err != nil {
            log.Printf("Couldn't get queues. Here's why: %v\n", err)
            break
        }
    }
}
```

```

} else {
    queueUrls = append(queueUrls, output.QueueUrls...)
}
}

if len(queueUrls) == 0 {
    fmt.Println("You don't have any queues!")
} else {
    for _, queueUrl := range queueUrls {
        fmt.Printf("\t%v\n", queueUrl)
    }
}
}
}

```

- Para obtener más información sobre la API, consulta [ListQueues](#) la Referencia AWS SDK para Go de la API.

Acciones

CreateQueue

En el siguiente ejemplo de código, se muestra cómo utilizar `CreateQueue`.

SDK para Go V2

 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 (
    "context"
    "encoding/json"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
```

```

    "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
    SqsClient *sqs.Client
}

// CreateQueue creates an Amazon SQS queue with the specified name. You can specify
// whether the queue is created as a FIFO queue.
func (actor SqsActions) CreateQueue(ctx context.Context, queueName string,
    isFifoQueue bool) (string, error) {
    var queueUrl string
    queueAttributes := map[string]string{}
    if isFifoQueue {
        queueAttributes["FifoQueue"] = "true"
    }
    queue, err := actor.SqsClient.CreateQueue(ctx, &sqs.CreateQueueInput{
        QueueName:  aws.String(queueName),
        Attributes: queueAttributes,
    })
    if err != nil {
        log.Printf("Couldn't create queue %v. Here's why: %v\n", queueName, err)
    } else {
        queueUrl = *queue.QueueUrl
    }

    return queueUrl, err
}

```

- Para obtener más información sobre la API, consulta [CreateQueue](#) la Referencia AWS SDK para Go de la API.

DeleteMessageBatch

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteMessageBatch.

SDK para Go V2

 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 (  
    "context"  
    "encoding/json"  
    "fmt"  
    "log"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/sqs"  
    "github.com/aws/aws-sdk-go-v2/service/sqs/types"  
)  
  
// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions  
// used in the examples.  
type SqsActions struct {  
    SqsClient *sqs.Client  
}  
  
// DeleteMessages uses the DeleteMessageBatch action to delete a batch of messages  
// from  
// an Amazon SQS queue.  
func (actor SqsActions) DeleteMessages(ctx context.Context, queueUrl string,  
    messages []types.Message) error {  
    entries := make([]types.DeleteMessageBatchRequestEntry, len(messages))  
    for msgIndex := range messages {  
        entries[msgIndex].Id = aws.String(fmt.Sprintf("%v", msgIndex))  
        entries[msgIndex].ReceiptHandle = messages[msgIndex].ReceiptHandle  
    }  
    _, err := actor.SqsClient.DeleteMessageBatch(ctx, &sqs.DeleteMessageBatchInput{  
        Entries:  entries,  
        QueueUrl: aws.String(queueUrl),  
    })
```

```
if err != nil {
    log.Printf("Couldn't delete messages from queue %v. Here's why: %v\n", queueUrl,
err)
}
return err
}
```

- Para obtener más información sobre la API, consulta [DeleteMessageBatch](#) la Referencia AWS SDK para Go de la API.

DeleteQueue

En el siguiente ejemplo de código, se muestra cómo utilizar DeleteQueue.

SDK para Go V2

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 (
    "context"
    "encoding/json"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
    "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
    SqsClient *sqs.Client
}
```

```
// DeleteQueue deletes an Amazon SQS queue.
func (actor SqsActions) DeleteQueue(ctx context.Context, queueUrl string) error {
    _, err := actor.SqsClient.DeleteQueue(ctx, &sqs.DeleteQueueInput{
        QueueUrl: aws.String(queueUrl)})
    if err != nil {
        log.Printf("Couldn't delete queue %v. Here's why: %v\n", queueUrl, err)
    }
    return err
}
```

- Para obtener más información sobre la API, consulta [DeleteQueue](#) la Referencia AWS SDK para Go de la API.

GetQueueAttributes

En el siguiente ejemplo de código, se muestra cómo utilizar `GetQueueAttributes`.

SDK para Go V2

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 (
    "context"
    "encoding/json"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
    "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)
```

```
// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
    SqsClient *sqs.Client
}

// GetQueueArn uses the GetQueueAttributes action to get the Amazon Resource Name
// (ARN)
// of an Amazon SQS queue.
func (actor SqsActions) GetQueueArn(ctx context.Context, queueUrl string) (string,
error) {
    var queueArn string
    arnAttributeName := types.QueueAttributeNameQueueArn
    attribute, err := actor.SqsClient.GetQueueAttributes(ctx,
&sqs.GetQueueAttributesInput{
    QueueUrl:      aws.String(queueUrl),
    AttributeNames: []types.QueueAttributeName{arnAttributeName},
})
    if err != nil {
        log.Printf("Couldn't get ARN for queue %v. Here's why: %v\n", queueUrl, err)
    } else {
        queueArn = attribute.Attributes[string(arnAttributeName)]
    }
    return queueArn, err
}
```

- Para obtener más información sobre la API, consulta [GetQueueAttributes](#) la Referencia AWS SDK para Go de la API.

ListQueues

En el siguiente ejemplo de código, se muestra cómo utilizar `ListQueues`.

SDK para Go V2

 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](#).

```
package main

import (
    "context"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Queue Service
// (Amazon SQS) client and list the queues in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
        fmt.Println(err)
        return
    }
    sqsClient := sqs.NewFromConfig(sdkConfig)
    fmt.Println("Let's list the queues for your account.")
    var queueUrls []string
    paginator := sqs.NewListQueuesPaginator(sqsClient, &sqs.ListQueuesInput{})
    for paginator.HasMorePages() {
        output, err := paginator.NextPage(ctx)
        if err != nil {
            log.Printf("Couldn't get queues. Here's why: %v\n", err)
            break
        }
    }
}
```



```

} else {
    queueUrls = append(queueUrls, output.QueueUrls...)
}
}

if len(queueUrls) == 0 {
    fmt.Println("You don't have any queues!")
} else {
    for _, queueUrl := range queueUrls {
        fmt.Printf("\t%v\n", queueUrl)
    }
}
}
}

```

- Para obtener más información sobre la API, consulta [ListQueues](#) la Referencia AWS SDK para Go de la API.

ReceiveMessage

En el siguiente ejemplo de código, se muestra cómo utilizar `ReceiveMessage`.

SDK para Go V2

 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 (
    "context"
    "encoding/json"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
    "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)
```

```
// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
    SqsClient *sqs.Client
}

// GetMessages uses the ReceiveMessage action to get messages from an Amazon SQS
// queue.
func (actor SqsActions) GetMessages(ctx context.Context, queueUrl string,
    maxMessages int32, waitTime int32) ([]types.Message, error) {
    var messages []types.Message
    result, err := actor.SqsClient.ReceiveMessage(ctx, &sqs.ReceiveMessageInput{
        QueueUrl:      aws.String(queueUrl),
        MaxNumberOfMessages: maxMessages,
        WaitTimeSeconds: waitTime,
    })
    if err != nil {
        log.Printf("Couldn't get messages from queue %v. Here's why: %v\n", queueUrl, err)
    } else {
        messages = result.Messages
    }
    return messages, err
}
```

- Para obtener más información sobre la API, consulta [ReceiveMessage](#) la Referencia AWS SDK para Go de la API.

SetQueueAttributes

En el siguiente ejemplo de código, se muestra cómo utilizar SetQueueAttributes.

SDK para Go V2

 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 (  
    "context"  
    "encoding/json"  
    "fmt"  
    "log"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/sqs"  
    "github.com/aws/aws-sdk-go-v2/service/sqs/types"  
)  
  
// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions  
// used in the examples.  
type SqsActions struct {  
    SqsClient *sqs.Client  
}  
  
// AttachSendMessagePolicy uses the SetQueueAttributes action to attach a policy to  
// an  
// Amazon SQS queue that allows the specified Amazon SNS topic to send messages to  
// the  
// queue.  
func (actor SqsActions) AttachSendMessagePolicy(ctx context.Context, queueUrl  
    string, queueArn string, topicArn string) error {  
    policyDoc := PolicyDocument{  
        Version: "2012-10-17",  
        Statement: []PolicyStatement{{  
            Effect:    "Allow",  
            Action:    "sqs:SendMessage",  
            Principal: map[string]string{"Service": "sns.amazonaws.com"},  
            Resource:  aws.String(queueArn),
```

```

    Condition: PolicyCondition{"ArnEquals": map[string]string{"aws:SourceArn":
topicArn}},
    }},
}
policyBytes, err := json.Marshal(policyDoc)
if err != nil {
    log.Printf("Couldn't create policy document. Here's why: %v\n", err)
    return err
}
_, err = actor.SqsClient.SetQueueAttributes(ctx, &sqs.SetQueueAttributesInput{
    Attributes: map[string]string{
        string(types.QueueAttributeNamePolicy): string(policyBytes),
    },
    QueueUrl: aws.String(queueUrl),
})
if err != nil {
    log.Printf("Couldn't set send message policy on queue %v. Here's why: %v\n",
queueUrl, err)
}
return err
}

// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
    Version    string
    Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
    Effect    string
    Action    string
    Principal map[string]string `json:",omitempty"`
    Resource  *string            `json:",omitempty"`
    Condition PolicyCondition    `json:",omitempty"`
}

// PolicyCondition defines a condition in a policy.
type PolicyCondition map[string]map[string]string

```

- Para obtener más información sobre la API, consulta [SetQueueAttributes](#) la Referencia AWS SDK para Go 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 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 Go V2

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](#).

Ejecutar un escenario interactivo en un símbolo del sistema.

```
import (  
    "context"  
    "encoding/json"  
    "fmt"  
    "log"  
    "strings"  
    "topics_and_queues/actions"  
  
    "github.com/aws/aws-sdk-go-v2/aws"  
    "github.com/aws/aws-sdk-go-v2/service/sns"  
    "github.com/aws/aws-sdk-go-v2/service/sqs"  
    "github.com/aws/aws-sdk-go-v2/service/sqs/types"  
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"  
)
```

```

const FIFO_SUFFIX = ".fifo"
const TONE_KEY = "tone"

var ToneChoices = []string{"cheerful", "funny", "serious", "sincere"}

// MessageBody is used to deserialize the body of a message from a JSON string.
type MessageBody struct {
    Message string
}

// ScenarioRunner separates the steps of this scenario into individual functions so
// that
// they are simpler to read and understand.
type ScenarioRunner struct {
    questioner demotools.IQuestioner
    snsActor   *actions.SnsActions
    sqsActor   *actions.SqsActions
}

func (runner ScenarioRunner) CreateTopic(ctx context.Context) (string, string, bool,
bool) {
    log.Println("SNS topics can be configured as FIFO (First-In-First-Out) or standard.
\n" +
        "FIFO topics deliver messages in order and support deduplication and message
filtering.")
    isFifoTopic := runner.questioner.AskBool("\nWould you like to work with FIFO
topics? (y/n) ", "y")

    contentBasedDeduplication := false
    if isFifoTopic {
        log.Println(strings.Repeat("-", 88))
        log.Println("Because you have chosen a FIFO topic, deduplication is supported.\n"
+
            "Deduplication IDs are either set in the message or are automatically generated
\n" +
            "from content using a hash function. If a message is successfully published to\n"
+
            "an SNS FIFO topic, any message published and determined to have the same\n" +
            "deduplication ID, within the five-minute deduplication interval, is accepted\n"
+
            "but not delivered. For more information about deduplication, see:\n" +
            "\thttps://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.")
        contentBasedDeduplication = runner.questioner.AskBool(

```

```

    "\nDo you want to use content-based deduplication instead of entering a
deduplication ID? (y/n) ", "y")
}
log.Println(strings.Repeat("-", 88))

topicName := runner.questioner.Ask("Enter a name for your SNS topic. ")
if isFifoTopic {
    topicName = fmt.Sprintf("%v%v", topicName, FIFO_SUFFIX)
    log.Printf("Because you have selected a FIFO topic, '%v' must be appended to\n"+
        "the topic name.", FIFO_SUFFIX)
}

topicArn, err := runner.snsActor.CreateTopic(ctx, topicName, isFifoTopic,
contentBasedDeduplication)
if err != nil {
    panic(err)
}
log.Printf("Your new topic with the name '%v' and Amazon Resource Name (ARN) \n"+
    "'%v' has been created.", topicName, topicArn)

return topicName, topicArn, isFifoTopic, contentBasedDeduplication
}

func (runner ScenarioRunner) CreateQueue(ctx context.Context, ordinal string,
isFifoTopic bool) (string, string) {
    queueName := runner.questioner.Ask(fmt.Sprintf("Enter a name for the %v SQS queue.
", ordinal))
    if isFifoTopic {
        queueName = fmt.Sprintf("%v%v", queueName, FIFO_SUFFIX)
        if ordinal == "first" {
            log.Printf("Because you are creating a FIFO SQS queue, '%v' must "+
                "be appended to the queue name.\n", FIFO_SUFFIX)
        }
    }
    queueUrl, err := runner.sqsActor.CreateQueue(ctx, queueName, isFifoTopic)
    if err != nil {
        panic(err)
    }
    log.Printf("Your new SQS queue with the name '%v' and the queue URL "+
        "'%v' has been created.", queueName, queueUrl)

    return queueName, queueUrl
}

```

```

func (runner ScenarioRunner) SubscribeQueueToTopic(
    ctx context.Context, queueName string, queueUrl string, topicName string, topicArn
    string, ordinal string,
    isFifoTopic bool) (string, bool) {

    queueArn, err := runner.sqsActor.GetQueueArn(ctx, queueUrl)
    if err != nil {
        panic(err)
    }
    log.Printf("The ARN of your queue is: %v.\n", queueArn)

    err = runner.sqsActor.AttachSendMessagePolicy(ctx, queueUrl, queueArn, topicArn)
    if err != nil {
        panic(err)
    }
    log.Println("Attached an IAM policy to the queue so the SNS topic can send " +
        "messages to it.")
    log.Println(strings.Repeat("-", 88))

    var filterPolicy map[string][]string
    if isFifoTopic {
        if ordinal == "first" {
            log.Println("Subscriptions to a FIFO topic can have filters.\n" +
                "If you add a filter to this subscription, then only the filtered messages\n" +
                "will be received in the queue.\n" +
                "For information about message filtering, see\n" +
                "\thttps://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n" +
                "For this example, you can filter messages by a \"tone\" attribute.")
        }

        wantFiltering := runner.questioner.AskBool(
            fmt.Sprintf("Do you want to filter messages that are sent to \"%v\"\n"+
                "from the %v topic? (y/n) ", queueName, topicName), "y")
        if wantFiltering {
            log.Println("You can filter messages by one or more of the following \"tone\"
            attributes.")

            var toneSelections []string
            askAboutTones := true
            for askAboutTones {
                toneIndex := runner.questioner.AskChoice(
                    "Enter the number of the tone you want to filter by:\n", ToneChoices)
                toneSelections = append(toneSelections, ToneChoices[toneIndex])
            }
        }
    }
}

```



```

    askAboutTones = runner.questioner.AskBool("Do you want to add another tone to
the filter? (y/n) ", "y")
}
log.Printf("Your subscription will be filtered to only pass the following tones:
%v\n", toneSelections)
filterPolicy = map[string][]string{TONE_KEY: toneSelections}
}
}

subscriptionArn, err := runner.snsActor.SubscribeQueue(ctx, topicArn, queueArn,
filterPolicy)
if err != nil {
    panic(err)
}
log.Printf("The queue %v is now subscribed to the topic %v with the subscription
ARN %v.\n",
    queueName, topicName, subscriptionArn)

return subscriptionArn, filterPolicy != nil
}

func (runner ScenarioRunner) PublishMessages(ctx context.Context, topicArn string,
isFifoTopic bool, contentBasedDeduplication bool, usingFilters bool) {
    var message string
    var groupId string
    var dedupId string
    var toneSelection string
    publishMore := true
    for publishMore {
        groupId = ""
        dedupId = ""
        toneSelection = ""
        message = runner.questioner.Ask("Enter a message to publish: ")
        if isFifoTopic {
            log.Println("Because you are using a FIFO topic, you must set a message group ID.
\n" +
                "All messages within the same group will be received in the order they were
published.")
            groupId = runner.questioner.Ask("Enter a message group ID: ")
            if !contentBasedDeduplication {
                log.Println("Because you are not using content-based deduplication,\n" +
                    "you must enter a deduplication ID.")
                dedupId = runner.questioner.Ask("Enter a deduplication ID: ")
            }
        }
    }
}

```

```

    }
    if usingFilters {
        if runner.questioner.AskBool("Add a tone attribute so this message can be
filtered? (y/n) ", "y") {
            toneIndex := runner.questioner.AskChoice(
                "Enter the number of the tone you want to filter by:\n", ToneChoices)
            toneSelection = ToneChoices[toneIndex]
        }
    }

    err := runner.snsActor.Publish(ctx, topicArn, message, groupId, dedupId, TONE_KEY,
toneSelection)
    if err != nil {
        panic(err)
    }
    log.Println(("Your message was published.))

    publishMore = runner.questioner.AskBool("Do you want to publish another message?
(y/n) ", "y")
}
}

func (runner ScenarioRunner) PollForMessages(ctx context.Context, queueUrls
[]string) {
    log.Println("Polling queues for messages...")
    for _, queueUrl := range queueUrls {
        var messages []types.Message
        for {
            currentMsgs, err := runner.sqsActor.GetMessages(ctx, queueUrl, 10, 1)
            if err != nil {
                panic(err)
            }
            if len(currentMsgs) == 0 {
                break
            }
            messages = append(messages, currentMsgs...)
        }
        if len(messages) == 0 {
            log.Printf("No messages were received by queue %v.\n", queueUrl)
        } else if len(messages) == 1 {
            log.Printf("One message was received by queue %v:\n", queueUrl)
        } else {
            log.Printf("%v messages were received by queue %v:\n", len(messages), queueUrl)
        }
    }
}

```

```

}
for msgIndex, message := range messages {
    messageBody := MessageBody{}
    err := json.Unmarshal([]byte(*message.Body), &messageBody)
    if err != nil {
        panic(err)
    }
    log.Printf("Message %v: %v\n", msgIndex+1, messageBody.Message)
}

if len(messages) > 0 {
    log.Printf("Deleting %v messages from queue %v.\n", len(messages), queueUrl)
    err := runner.sqsActor.DeleteMessages(ctx, queueUrl, messages)
    if err != nil {
        panic(err)
    }
}
}
}

// RunTopicsAndQueuesScenario is an interactive example that shows you how to use
// the
// AWS SDK for Go to create and use Amazon SNS topics and Amazon SQS queues.
//
// 1. Create a topic (FIFO or non-FIFO).
// 2. Subscribe several queues to the topic with an option to apply a filter.
// 3. Publish messages to the topic.
// 4. Poll the queues for messages received.
// 5. Delete the topic and the queues.
//
// This example creates service clients from the specified sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
// example.
// This package can be found in the ../../demotools folder of this repo.
func RunTopicsAndQueuesScenario(
    ctx context.Context, sdkConfig aws.Config, questioner demotools.IQuestioner) {
    resources := Resources{}
    defer func() {
        if r := recover(); r != nil {
            log.Println("Something went wrong with the demo.\n" +
                "Cleaning up any resources that were created...")
            resources.Cleanup(ctx)
        }
    }()

```

```

    }
    }()
    queueCount := 2

    log.Println(strings.Repeat("-", 88))
    log.Printf("Welcome to messaging with topics and queues.\n\n"+
        "In this scenario, you will create an SNS topic and subscribe %v SQS queues to the\n"+
        "topic. You can select from several options for configuring the topic and the\n"+
        "subscriptions for the queues. You can then post to the topic and see the results\n"+
        "in the queues.\n", queueCount)

    log.Println(strings.Repeat("-", 88))

    runner := ScenarioRunner{
        questioner: questioner,
        snsActor:   &actions.SnsActions{SnsClient: sns.NewFromConfig(sdkConfig)},
        sqsActor:   &actions.SqsActions{SqsClient: sqs.NewFromConfig(sdkConfig)},
    }
    resources.snsActor = runner.snsActor
    resources.sqsActor = runner.sqsActor

    topicName, topicArn, isFifoTopic, contentBasedDeduplication :=
    runner.CreateTopic(ctx)
    resources.topicArn = topicArn
    log.Println(strings.Repeat("-", 88))

    log.Printf("Now you will create %v SQS queues and subscribe them to the topic.\n",
        queueCount)
    ordinals := []string{"first", "next"}
    usingFilters := false
    for _, ordinal := range ordinals {
        queueName, queueUrl := runner.CreateQueue(ctx, ordinal, isFifoTopic)
        resources.queueUrls = append(resources.queueUrls, queueUrl)

        _, filtering := runner.SubscribeQueueToTopic(ctx, queueName, queueUrl, topicName,
            topicArn, ordinal, isFifoTopic)
        usingFilters = usingFilters || filtering
    }

    log.Println(strings.Repeat("-", 88))
    runner.PublishMessages(ctx, topicArn, isFifoTopic, contentBasedDeduplication,
        usingFilters)

```

```

log.Println(strings.Repeat("-", 88))
runner.PollForMessages(ctx, resources.queueUrls)

log.Println(strings.Repeat("-", 88))

wantCleanup := questioner.AskBool("Do you want to remove all AWS resources created
for this scenario? (y/n) ", "y")
if wantCleanup {
    log.Println("Cleaning up resources...")
    resources.Cleanup(ctx)
}

log.Println(strings.Repeat("-", 88))
log.Println("Thanks for watching!")
log.Println(strings.Repeat("-", 88))
}

```

Defina una estructura que encapsule las acciones de Amazon SNS utilizadas en este ejemplo.

```

import (
    "context"
    "encoding/json"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/sns"
    "github.com/aws/aws-sdk-go-v2/service/sns/types"
)

// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
// actions
// used in the examples.
type SnsActions struct {
    SnsClient *sns.Client
}

// CreateTopic creates an Amazon SNS topic with the specified name. You can
// optionally

```

```

// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(ctx context.Context, topicName string,
    isFifoTopic bool, contentBasedDeduplication bool) (string, error) {
    var topicArn string
    topicAttributes := map[string]string{}
    if isFifoTopic {
        topicAttributes["FifoTopic"] = "true"
    }
    if contentBasedDeduplication {
        topicAttributes["ContentBasedDeduplication"] = "true"
    }
    topic, err := actor.SnsClient.CreateTopic(ctx, &sns.CreateTopicInput{
        Name:      aws.String(topicName),
        Attributes: topicAttributes,
    })
    if err != nil {
        log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
    } else {
        topicArn = *topic.TopicArn
    }

    return topicArn, err
}

// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(ctx context.Context, topicArn string) error {
    _, err := actor.SnsClient.DeleteTopic(ctx, &sns.DeleteTopicInput{
        TopicArn: aws.String(topicArn)})
    if err != nil {
        log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
    }
    return err
}

// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
policy

```

```
// so that messages are only sent to the queue when the message has the specified
// attributes.
func (actor SnsActions) SubscribeQueue(ctx context.Context, topicArn string,
queueArn string, filterMap map[string][]string) (string, error) {
    var subscriptionArn string
    var attributes map[string]string
    if filterMap != nil {
        filterBytes, err := json.Marshal(filterMap)
        if err != nil {
            log.Printf("Couldn't create filter policy, here's why: %v\n", err)
            return "", err
        }
        attributes = map[string]string{"FilterPolicy": string(filterBytes)}
    }
    output, err := actor.SnsClient.Subscribe(ctx, &sns.SubscribeInput{
        Protocol:      aws.String("sqs"),
        TopicArn:      aws.String(topicArn),
        Attributes:    attributes,
        Endpoint:      aws.String(queueArn),
        ReturnSubscriptionArn: true,
    })
    if err != nil {
        log.Printf("Couldn't subscribe queue %v to topic %v. Here's why: %v\n",
            queueArn, topicArn, err)
    } else {
        subscriptionArn = *output.SubscriptionArn
    }

    return subscriptionArn, err
}

// Publish publishes a message to an Amazon SNS topic. The message is then sent to
// all
// subscribers. When the topic is a FIFO topic, the message must also contain a
// group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
// value
// filter attribute can be specified so that the message can be filtered according
// to
// a filter policy.
```

```

func (actor SnsActions) Publish(ctx context.Context, topicArn string, message
    string, groupId string, dedupId string, filterKey string, filterValue string) error
{
    publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
    aws.String(message)}
    if groupId != "" {
        publishInput.MessageGroupId = aws.String(groupId)
    }
    if dedupId != "" {
        publishInput.MessageDeduplicationId = aws.String(dedupId)
    }
    if filterKey != "" && filterValue != "" {
        publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
            filterKey: {DataType: aws.String("String"), StringValue:
            aws.String(filterValue)},
        }
    }
    _, err := actor.SnsClient.Publish(ctx, &publishInput)
    if err != nil {
        log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn, err)
    }
    return err
}

```

Defina una estructura que encapsule las acciones de Amazon SQS utilizadas en este ejemplo.

```

import (
    "context"
    "encoding/json"
    "fmt"
    "log"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
    "github.com/aws/aws-sdk-go-v2/service/sqs/types"
)

// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {

```



```

    SqsClient *sqs.Client
}

// CreateQueue creates an Amazon SQS queue with the specified name. You can specify
// whether the queue is created as a FIFO queue.
func (actor SqsActions) CreateQueue(ctx context.Context, queueName string,
    isFifoQueue bool) (string, error) {
    var queueUrl string
    queueAttributes := map[string]string{}
    if isFifoQueue {
        queueAttributes["FifoQueue"] = "true"
    }
    queue, err := actor.SqsClient.CreateQueue(ctx, &sqs.CreateQueueInput{
        QueueName:  aws.String(queueName),
        Attributes: queueAttributes,
    })
    if err != nil {
        log.Printf("Couldn't create queue %v. Here's why: %v\n", queueName, err)
    } else {
        queueUrl = *queue.QueueUrl
    }

    return queueUrl, err
}

// GetQueueArn uses the GetQueueAttributes action to get the Amazon Resource Name
// (ARN)
// of an Amazon SQS queue.
func (actor SqsActions) GetQueueArn(ctx context.Context, queueUrl string) (string,
    error) {
    var queueArn string
    arnAttributeName := types.QueueAttributeNameQueueArn
    attribute, err := actor.SqsClient.GetQueueAttributes(ctx,
        &sqs.GetQueueAttributesInput{
            QueueUrl:      aws.String(queueUrl),
            AttributeNames: []types.QueueAttributeName{arnAttributeName},
        })
    if err != nil {
        log.Printf("Couldn't get ARN for queue %v. Here's why: %v\n", queueUrl, err)
    } else {

```

```

    queueArn = attribute.Attributes[string(arnAttributeName)]
}
return queueArn, err
}

// AttachSendMessagePolicy uses the SetQueueAttributes action to attach a policy to
an
// Amazon SQS queue that allows the specified Amazon SNS topic to send messages to
the
// queue.
func (actor SqsActions) AttachSendMessagePolicy(ctx context.Context, queueUrl
string, queueArn string, topicArn string) error {
    policyDoc := PolicyDocument{
        Version: "2012-10-17",
        Statement: []PolicyStatement{{
            Effect:    "Allow",
            Action:  "sqs:SendMessage",
            Principal: map[string]string{"Service": "sns.amazonaws.com"},
            Resource: aws.String(queueArn),
            Condition: PolicyCondition{"ArnEquals": map[string]string{"aws:SourceArn":
topicArn}},
        }},
    }
    policyBytes, err := json.Marshal(policyDoc)
    if err != nil {
        log.Printf("Couldn't create policy document. Here's why: %v\n", err)
        return err
    }
    _, err = actor.SqsClient.SetQueueAttributes(ctx, &sqs.SetQueueAttributesInput{
        Attributes: map[string]string{
            string(types.QueueAttributeNamePolicy): string(policyBytes),
        },
        QueueUrl: aws.String(queueUrl),
    })
    if err != nil {
        log.Printf("Couldn't set send message policy on queue %v. Here's why: %v\n",
queueUrl, err)
    }
    return err
}

// PolicyDocument defines a policy document as a Go struct that can be serialized

```

```
// to JSON.
type PolicyDocument struct {
    Version    string
    Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
    Effect    string
    Action    string
    Principal map[string]string `json:",omitempty"`
    Resource  *string             `json:",omitempty"`
    Condition PolicyCondition     `json:",omitempty"`
}

// PolicyCondition defines a condition in a policy.
type PolicyCondition map[string]map[string]string

// GetMessages uses the ReceiveMessage action to get messages from an Amazon SQS
// queue.
func (actor SqsActions) GetMessages(ctx context.Context, queueUrl string,
    maxMessages int32, waitTime int32) ([]types.Message, error) {
    var messages []types.Message
    result, err := actor.SqsClient.ReceiveMessage(ctx, &sqs.ReceiveMessageInput{
        QueueUrl:          aws.String(queueUrl),
        MaxNumberOfMessages: maxMessages,
        WaitTimeSeconds:    waitTime,
    })
    if err != nil {
        log.Printf("Couldn't get messages from queue %v. Here's why: %v\n", queueUrl, err)
    } else {
        messages = result.Messages
    }
    return messages, err
}

// DeleteMessages uses the DeleteMessageBatch action to delete a batch of messages
// from
// an Amazon SQS queue.
```

```

func (actor SqsActions) DeleteMessages(ctx context.Context, queueUrl string,
    messages []types.Message) error {
    entries := make([]types.DeleteMessageBatchRequestEntry, len(messages))
    for msgIndex := range messages {
        entries[msgIndex].Id = aws.String(fmt.Sprintf("%v", msgIndex))
        entries[msgIndex].ReceiptHandle = messages[msgIndex].ReceiptHandle
    }
    _, err := actor.SqsClient.DeleteMessageBatch(ctx, &sqs.DeleteMessageBatchInput{
        Entries:  entries,
        QueueUrl: aws.String(queueUrl),
    })
    if err != nil {
        log.Printf("Couldn't delete messages from queue %v. Here's why: %v\n", queueUrl, err)
    }
    return err
}

// DeleteQueue deletes an Amazon SQS queue.
func (actor SqsActions) DeleteQueue(ctx context.Context, queueUrl string) error {
    _, err := actor.SqsClient.DeleteQueue(ctx, &sqs.DeleteQueueInput{
        QueueUrl: aws.String(queueUrl)})
    if err != nil {
        log.Printf("Couldn't delete queue %v. Here's why: %v\n", queueUrl, err)
    }
    return err
}

```

Eliminación de recursos.

```

import (
    "context"
    "fmt"
    "log"
    "topics_and_queues/actions"
)

// Resources keeps track of AWS resources created during an example and handles

```

```
// cleanup when the example finishes.
type Resources struct {
    topicArn  string
    queueUrls []string
    snsActor  *actions.SnsActions
    sqsActor  *actions.SqsActions
}

// Cleanup deletes all AWS resources created during an example.
func (resources Resources) Cleanup(ctx context.Context) {
    defer func() {
        if r := recover(); r != nil {
            fmt.Println("Something went wrong during cleanup. Use the AWS Management Console\n" +
                "to remove any remaining resources that were created for this scenario.")
        }
    }()

    var err error
    if resources.topicArn != "" {
        log.Printf("Deleting topic %v.\n", resources.topicArn)
        err = resources.snsActor.DeleteTopic(ctx, resources.topicArn)
        if err != nil {
            panic(err)
        }
    }

    for _, queueUrl := range resources.queueUrls {
        log.Printf("Deleting queue %v.\n", queueUrl)
        err = resources.sqsActor.DeleteQueue(ctx, queueUrl)
        if err != nil {
            panic(err)
        }
    }
}
```

- Para obtener detalles sobre la API, consulte los siguientes temas en la Referencia de la API de AWS SDK para Go .
 - [CreateQueue](#)
 - [CreateTopic](#)

- [DeleteMessageBatch](#)
- [DeleteQueue](#)
- [DeleteTopic](#)
- [GetQueueAttributes](#)
- [Publish](#)
- [ReceiveMessage](#)
- [SetQueueAttributes](#)
- [Subscribe](#)
- [Unsubscribe](#)

Ejemplos de tecnología sin servidor

Invocar una función de Lambda desde un desencadenador de Amazon SQS

En el siguiente ejemplo de código se muestra cómo implementar una función de Lambda que recibe un evento activado al recibir mensajes de una cola de SQS. La función recupera los mensajes del parámetro de eventos y registra el contenido de cada mensaje.

SDK para Go V2

Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Uso de un evento de SQS con Lambda mediante Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package integration_sqs_to_lambda

import (
    "fmt"
    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
)
```

```
func handler(event events.SQSEvent) error {
    for _, record := range event.Records {
        err := processMessage(record)
        if err != nil {
            return err
        }
    }
    fmt.Println("done")
    return nil
}

func processMessage(record events.SQSMessage) error {
    fmt.Printf("Processed message %s\n", record.Body)
    // TODO: Do interesting work based on the new message
    return nil
}

func main() {
    lambda.Start(handler)
}
```

Notificación de los errores de los elementos del lote de las funciones de Lambda mediante un desencadenador de Amazon SQS.

En el siguiente ejemplo de código se muestra cómo implementar una respuesta por lotes parcial para funciones de Lambda que reciben eventos de una cola de SQS. La función informa los errores de los elementos del lote en la respuesta y le indica a Lambda que vuelva a intentar esos mensajes más adelante.

SDK para Go V2

 Note

Hay más información GitHub. Busque el ejemplo completo y aprenda a configurar y ejecutar en el repositorio de [ejemplos de tecnología sin servidor](#).

Notificación de los errores de los elementos del lote de SQS con Lambda mediante Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
```

```
// SPDX-License-Identifier: Apache-2.0
package main

import (
    "context"
    "fmt"
    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, sqsEvent events.SQSEvent) (map[string]interface{},
    error) {
    batchItemFailures := []map[string]interface{}{}

    for _, message := range sqsEvent.Records {
        if len(message.Body) > 0 {
            // Your message processing condition here
            fmt.Printf("Successfully processed message: %s\n", message.Body)
        } else {
            // Message processing failed
            fmt.Printf("Failed to process message %s\n", message.MessageId)
            batchItemFailures = append(batchItemFailures, map[string]interface{}{
                "itemIdentifier": message.MessageId})
        }
    }

    sqsBatchResponse := map[string]interface{}{
        "batchItemFailures": batchItemFailures,
    }
    return sqsBatchResponse, nil
}

func main() {
    lambda.Start(handler)
}
```


Migre a la AWS SDK para Go versión 2

Versión mínima de Go

AWS SDK para Go Requiere una versión mínima de Go 1.23. La última versión de Go se puede descargar en la página de [descargas](#). Consulte el [historial de versiones](#) para obtener más información sobre cada lanzamiento de versión de Go, así como la información pertinente necesaria para la actualización.

Modularización

Se AWS SDK para Go ha actualizado para aprovechar los módulos Go, que se convirtieron en el modo de desarrollo predeterminado en Go 1.13. Varios de los paquetes proporcionados por el SDK se han modularizado y ahora se versionan y publican de forma independiente. Gracias a este cambio, puede mejorarse el modelado de dependencias de las aplicaciones y se permite que el SDK proporcione nuevas características y funcionalidades que siguen la estrategia de control de versiones de los módulos de Go.

En la lista siguiente se incluyen algunos módulos de Go proporcionados por el SDK:

Módulo	Description (Descripción)
<code>github.com/aws/aws-sdk-go-v2</code>	El núcleo del SDK
<code>github.com/aws/aws-sdk-go-v2/config</code>	Carga de la configuración compartida
<code>github.com/aws/aws-sdk-go-v2/credentials</code>	AWS Proveedores de credenciales
<code>github.com/aws/aws-sdk-go-v2/feature/ec2/imds</code>	Cliente de Amazon EC2 Instance Metadata Service

Los clientes de servicio y los módulos de utilidades de nivel superior del SDK se anidan en las siguientes rutas de importación:

Raíz de la importación	Description (Descripción)
<code>github.com/aws/aws-sdk-go-v2/service/</code>	Módulos del cliente de servicio
<code>github.com/aws/aws-sdk-go-v2/feature/</code>	Utilidades de alto nivel para servicios como el administrador de transferencia de Amazon S3

Carga de la configuración

El paquete [session](#) y la funcionalidad asociada se sustituyen por un sistema de configuración simplificado que proporciona el paquete [config](#). El paquete `config` es un módulo de Go independiente y se puede incluir en las dependencias de la aplicación con `go get`.

```
go get github.com/aws/aws-sdk-go-v2/config
```

[La sesión. Nueva, sesión.](#) `NewSession` [NewSessionWithOptions](#), y `session.MUST` deben migrarse a [config](#). `LoadDefaultConfig`.

El paquete `config` proporciona varias funciones auxiliares que contribuyen a reemplazar la configuración compartida que se carga mediante programación. Estos nombres de función llevan el prefijo `With`, seguido de la opción que reemplazan. Veamos algunos ejemplos de cómo migrar el uso del paquete `session`.

Para obtener más información sobre cómo cargar la configuración compartida, consulte [Configurar el SDK](#).

Ejemplos

Migrar de a `NewSession` `LoadDefaultConfig`

En el ejemplo siguiente se muestra cómo se migra el uso de `session.NewSession` sin parámetros de argumento adicionales a `config.LoadDefaultConfig`.

```
// V1 using NewSession

import "github.com/aws/aws-sdk-go/aws/session"
```

```
// ...

sess, err := session.NewSession()
if err != nil {
    // handle error
}
```

```
// V2 using LoadDefaultConfig

import "context"
import "github.com/aws/aws-sdk-go-v2/config"

// ...

cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    // handle error
}
```

Migrar desde NewSession con las opciones de AWS.config

En el ejemplo se muestra cómo migrar el reemplazo de valores `aws.Config` durante la carga de la configuración. Se pueden proporcionar una o varias funciones auxiliares `config.With*` a `config.LoadDefaultConfig` para reemplazar los valores de configuración cargados. [En este ejemplo, se anula AWS la región para usar la configuración. `us-west-2 WithRegion`](#) función auxiliar.

```
// V1

import "github.com/aws/aws-sdk-go/aws"
import "github.com/aws/aws-sdk-go/aws/session"

// ...

sess, err := session.NewSession(aws.Config{
    Region: aws.String("us-west-2")
})
if err != nil {
    // handle error
}
```

```
// V2
```

```
import "context"
import "github.com/aws/aws-sdk-go-v2/config"

// ...

cfg, err := config.LoadDefaultConfig(context.TODO(),
    config.WithRegion("us-west-2"),
)
if err != nil {
    // handle error
}
```

Migrando desde NewSessionWithOptions

En este ejemplo se muestra cómo migrar el reemplazo de valores durante la carga de la configuración. Se pueden proporcionar cero o más funciones auxiliares `config.With*` a `config.LoadDefaultConfig` para anular los valores de configuración cargados. En este ejemplo, mostramos cómo anular el perfil de destino que se utiliza al cargar la configuración compartida del AWS SDK.

```
// V1

import "github.com/aws/aws-sdk-go/aws"
import "github.com/aws/aws-sdk-go/aws/session"

// ...

sess, err := session.NewSessionWithOptions(aws.Config{
    Profile: "my-application-profile"
})
if err != nil {
    // handle error
}
```

```
// V2

import "context"
import "github.com/aws/aws-sdk-go-v2/config"

// ...
```

```

cfg, err := config.LoadDefaultConfig(context.TODO(),
    config.WithSharedConfigProfile("my-application-profile"),
)
if err != nil {
    // handle error
}

```

Simulación e ***iface**

Los paquetes `*iface` y las interfaces incluidas (por ejemplo, [s3iface.S3API](#)) se han eliminado. Estas definiciones de interfaz no son estables, ya que se invalidan cada vez que un servicio agrega una nueva operación.

Para las operaciones de servicio que se utilizan, se debe sustituir el uso de `*iface` por interfaces definidas por el intermediario con un ámbito específico:

```

// V1

import "io"

import "github.com/aws/aws-sdk-go/service/s3"
import "github.com/aws/aws-sdk-go/service/s3/s3iface"

func GetObjectBytes(client s3iface.S3API, bucket, key string) ([]byte, error) {
    object, err := client.GetObject(&s3.GetObjectInput{
        Bucket: &bucket,
        Key:    &key,
    })
    if err != nil {
        return nil, err
    }
    defer object.Body.Close()

    return io.ReadAll(object.Body)
}

```

```

// V2

import "context"
import "io"

```

```
import "github.com/aws/aws-sdk-go-v2/service/s3"

type GetObjectAPIClient interface {
    GetObject(context.Context, *s3.GetObjectInput, ...func(*s3.Options))
    (*s3.GetObjectOutput, error)
}

func GetObjectBytes(ctx context.Context, client GetObjectAPIClient, bucket, key string)
([]byte, error) {
    object, err := client.GetObject(ctx, &s3.GetObjectInput{
        Bucket: &bucket,
        Key:    &key,
    })
    if err != nil {
        return nil, err
    }
    defer object.Body.Close()

    return io.ReadAll(object.Body)
}
```

Consulte [Pruebas unitarias con AWS SDK para Go v2](#) para obtener más información.

Credenciales y proveedores de credenciales

El paquete [aws/credenciales](#) y los proveedores de credenciales asociados se han reubicado en el emplazamiento del paquete [credentials](#). El paquete `credentials` es un módulo de Go que se recupera mediante `go get`.

```
go get github.com/aws/aws-sdk-go-v2/credentials
```

La versión AWS SDK para Go v2 actualiza los proveedores de AWS credenciales para proporcionar una interfaz coherente para recuperar AWS las credenciales. [Cada proveedor implementa las leyes. CredentialsProvider](#)interfaz, que define un `Retrieve` método que devuelve `un(aws.Credentials, error)`. [AWS.Credentials, que es análogo al tipo AWS SDK para Go v1 Credentials.value](#).

[Debe empaquetar los objetos con aws. aws.CredentialsProvider CredentialsCache](#)para permitir que se almacene en caché las credenciales. Se usa [NewCredentialsCache](#)para construir un `aws.CredentialsCache` objeto. De forma predeterminada, las credenciales configuradas por `config.LoadDefaultConfig` se encapsulan con `aws.CredentialsCache`.

La siguiente tabla muestra los cambios de ubicación de los proveedores de AWS credenciales de la AWS SDK para Go versión 1 a la versión 2.

Name	Importación de V1	Importación de V2
Credenciales de rol de Amazon EC2 IAM	<code>github.com/aws/aws-sdk-go/aws/credentials/ec2rolecreds</code>	<code>github.com/aws/aws-sdk-go-v2/credentials/ec2rolecreds</code>
Credenciales de los puntos de conexión	<code>github.com/aws/aws-sdk-go/aws/credentials/endpointcreds</code>	<code>github.com/aws/aws-sdk-go-v2/credentials/endpointcreds</code>
Credenciales de proceso	<code>github.com/aws/aws-sdk-go/aws/credentials/processcreds</code>	<code>github.com/aws/aws-sdk-go-v2/credentials/processcreds</code>
AWS Security Token Service	<code>github.com/aws/aws-sdk-go/aws/credentials/stscreds</code>	<code>github.com/aws/aws-sdk-go-v2/credentials/stscreds</code>

Credenciales estáticas

Aplicaciones que utilizan [credenciales](#). `NewStaticCredentials` [para construir una credencial estática mediante programación debe usar credenciales](#). `NewStaticCredentialsProvider`.

Ejemplo

```
// V1

import "github.com/aws/aws-sdk-go/aws/credentials"

// ...

appCreds := credentials.NewStaticCredentials(accessKey, secretKey, sessionToken)
value, err := appCreds.Get()
if err != nil {
    // handle error
}
```

```
// V2

import "context"
import "github.com/aws/aws-sdk-go-v2/aws"
import "github.com/aws/aws-sdk-go-v2/credentials"

// ...

appCreds := aws.NewCredentialsCache(credentials.NewStaticCredentialsProvider(accessKey,
    secretKey, sessionToken))
value, err := appCreds.Retrieve(context.TODO())
if err != nil {
    // handle error
}
```

Credenciales de rol de Amazon EC2 IAM

Debe migrar el uso de [NewCredentials](#) y [NewCredentialsWithClient](#) a usar [New](#).

El `ec2rolecreds` paquete `ec2rolecreds.New` toma las opciones funcionales de [EC2Rolecreds.options](#) como entrada, lo que le permite anular el cliente específico de Amazon EC2 Instance Metadata Service que va a utilizar o anular la ventana de caducidad de las credenciales.

Ejemplo

```
// V1

import "github.com/aws/aws-sdk-go/aws/credentials/ec2rolecreds"

// ...

appCreds := ec2rolecreds.NewCredentials(sess)
value, err := appCreds.Get()
if err != nil {
    // handle error
}
```

```
// V2

import "context"
import "github.com/aws/aws-sdk-go-v2/aws"
```



```
import "github.com/aws/aws-sdk-go-v2/credentials/ec2rolecreds"

// ...

// New returns an object of a type that satisfies the aws.CredentialProvider interface
appCreds := aws.NewCredentialsCache(ec2rolecreds.New())
value, err := appCreds.Retrieve(context.TODO())
if err != nil {
    // handle error
}
```

Credenciales de los puntos de conexión

[Debe migrar el uso de New y usar New. NewCredentialsClientNewProviderClient](#)

La función `New` del paquete `endpointcreds` utiliza un argumento de cadena que contiene la dirección URL de un punto de conexión HTTP o HTTPS del que se recuperan las credenciales, así como opciones funcionales de [endpointcreds.Options](#) para mutar el proveedor de credenciales y reemplazar opciones de configuración específicas.

Credenciales de proceso

Debe migrar el uso de [NewCredentialsNewCredentialsCommand](#), y [NewCredentialsTimeout](#) para usar [NewProvider](#) o [NewProviderCommand](#).

La función `NewProvider` del paquete `processcreds` toma un argumento de cadena que es el comando que se va a ejecutar en el intérprete de comandos del entorno de host, así como opciones funcionales de [Options](#) para mutar el proveedor de credenciales y reemplazar opciones de configuración específicas.

`NewProviderCommand` adopta una implementación de la [NewCommandBuilder](#) interfaz que define comandos de proceso más complejos que pueden aceptar uno o más argumentos de línea de comandos o tener ciertos requisitos de entorno de ejecución. [DefaultNewCommandBuilder](#) implementa esta interfaz y define un generador de comandos para un proceso que requiere varios argumentos de línea de comandos.

Ejemplo

```
// V1

import "github.com/aws/aws-sdk-go/aws/credentials/processcreds"
```

```
// ...

appCreds := processcreds.NewCredentials("/path/to/command")
value, err := appCreds.Get()
if err != nil {
    // handle error
}
```

```
// V2

import "context"
import "github.com/aws/aws-sdk-go-v2/aws"
import "github.com/aws/aws-sdk-go-v2/credentials/processcreds"

// ...

appCreds := aws.NewCredentialsCache(processcreds.NewProvider("/path/to/command"))
value, err := appCreds.Retrieve(context.TODO())
if err != nil {
    // handle error
}
```

AWS Security Token Service Credenciales

AssumeRole

Debe migrar el uso de [NewCredentials](#) y [NewCredentialsWithClient](#) para usar [NewAssumeRoleProvider](#)

Se debe llamar a la `NewAssumeRoleProvider` función del `stscreds` paquete con un [STS.Client](#) y el AWS Identity and Access Management ARN del rol debe asumirse a partir de las credenciales configuradas del proveedor. `sts.Client` También puede proporcionar un conjunto de opciones funcionales [AssumeRoleOptions](#) para modificar otras configuraciones opcionales del proveedor.

Ejemplo

```
// V1

import "github.com/aws/aws-sdk-go/aws/credentials/stscreds"

// ...
```

```
appCreds := stscreds.NewCredentials(sess, "arn:aws:iam::123456789012:role/demo")
value, err := appCreds.Get()
if err != nil {
    // handle error
}
```

```
// V2

import "context"
import "github.com/aws/aws-sdk-go-v2/credentials/stscreds"
import "github.com/aws/aws-sdk-go-v2/service/sts"

// ...

client := sts.NewFromConfig(cfg)

appCreds := stscreds.NewAssumeRoleProvider(client, "arn:aws:iam::123456789012:role/
demo")
value, err := appCreds.Retrieve(context.TODO())
if err != nil {
    // handle error
}
```

AssumeRoleWithWebIdentity

Debe migrar el uso de [NewWebIdentityCredentials](#) [NewWebIdentityRoleProvider](#), y [NewWebIdentityRoleProviderWithToken](#) para usar [NewWebIdentityRoleProvider](#).

Se debe llamar a la `NewWebIdentityRoleProvider` función del `stscreds` paquete con un [STS.Client](#) y se debe asumir el AWS Identity and Access Management ARN del rol utilizando las credenciales configuradas `sts.Client` del proveedor, y una implementación de a [IdentityTokenRetriever](#) para proporcionar el token 2.0 o OAuth OpenID Connect ID. [IdentityTokenFile](#) es una `IdentityTokenRetriever` que se puede utilizar para proporcionar el token de identidad web desde un archivo ubicado en el sistema de archivos host de la aplicación. También puede proporcionar un conjunto de opciones funcionales [WebIdentityRoleOptions](#) para modificar otras configuraciones opcionales para el proveedor.

Ejemplo

```
// V1
```

```
import "github.com/aws/aws-sdk-go/aws/credentials/stscreds"

// ...

appCreds := stscreds.NewWebIdentityRoleProvider(sess, "arn:aws:iam::123456789012:role/
demo", "sessionName", "/path/to/token")
value, err := appCreds.Get()
if err != nil {
    // handle error
}
```

```
// V2

import "context"
import "github.com/aws/aws-sdk-go-v2/aws"
import "github.com/aws/aws-sdk-go-v2/credentials/stscreds"
import "github.com/aws/aws-sdk-go-v2/service/sts"

// ...

client := sts.NewFromConfig(cfg)

appCreds := aws.NewCredentialsCache(stscreds.NewWebIdentityRoleProvider(
    client,
    "arn:aws:iam::123456789012:role/demo",
    stscreds.IdentityTokenFile("/path/to/file"),
    func(o *stscreds.WebIdentityRoleOptions) {
        o.RoleSessionName = "sessionName"
    }))
value, err := appCreds.Retrieve(context.TODO())
if err != nil {
    // handle error
}
```

Clientes de servicio

AWS SDK para Go proporciona módulos de cliente de servicio anidados en la ruta de `github.com/aws/aws-sdk-go-v2/service` importación. Cada cliente de servicio está incluido en un paquete Go que utiliza el identificador único de cada servicio. En la tabla siguiente se muestran algunos ejemplos de rutas de importación de servicios en AWS SDK para Go.

Nombre del servicio	Ruta de importación de V1	Ruta de importación de V2
Amazon S3	github.com/aws/aws-sdk-go/service/s3	github.com/aws/aws-sdk-go-v2/service/s3
Amazon DynamoDB	github.com/aws/aws-sdk-go/service/dynamodb	github.com/aws/aws-sdk-go-v2/service/dynamodb
Amazon CloudWatch Logs	github.com/aws/aws-sdk-go/service/cloudwatchlogs	github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs

Cada paquete de clientes de servicio es un módulo Go versionado de forma independiente. Para agregar el cliente de servicio como dependencia de su aplicación, use el comando `go get` con la ruta de importación del servicio. Por ejemplo, para agregar el cliente de Amazon S3 a sus dependencias, utilice:

```
go get github.com/aws/aws-sdk-go-v2/service/s3
```

Creación de clientes

Puede crear clientes AWS SDK para Go utilizando las funciones `New` o `NewFromConfig` constructoras del paquete del cliente. Al migrar desde la AWS SDK para Go versión 1, le recomendamos que utilice la `NewFromConfig` variante, que devolverá un nuevo cliente de servicio con los valores de `anaws.Config`. El valor `aws.Config` se habrá creado al cargar la configuración compartida del SDK mediante `config.LoadDefaultConfig`. Para obtener más información sobre la creación de clientes de servicio, consulte [Uso de AWS SDK para Go v2 con los servicios de AWS](#).

Ejemplo 1

```
// V1

import "github.com/aws/aws-sdk-go/aws/session"
import "github.com/aws/aws-sdk-go/service/s3"

// ...
```

```
sess, err := session.NewSession()
if err != nil {
    // handle error
}

client := s3.New(sess)
```

```
// V2

import "context"
import "github.com/aws/aws-sdk-go-v2/config"
import "github.com/aws/aws-sdk-go-v2/service/s3"

// ...

cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    // handle error
}

client := s3.NewFromConfig(cfg)
```

Ejemplo 2: Reemplazo de la configuración del cliente

```
// V1

import "github.com/aws/aws-sdk-go/aws"
import "github.com/aws/aws-sdk-go/aws/session"
import "github.com/aws/aws-sdk-go/service/s3"

// ...

sess, err := session.NewSession()
if err != nil {
    // handle error
}

client := s3.New(sess, &aws.Config{
    Region: aws.String("us-west-2"),
})
```

```
// V2
```

```
import "context"
import "github.com/aws/aws-sdk-go-v2/config"
import "github.com/aws/aws-sdk-go-v2/service/s3"

// ...

cfg, err := config.LoadDefaultConfig(context.TODO())
if err != nil {
    // handle error
}

client := s3.NewFromConfig(cfg, func(o *s3.Options) {
    o.Region = "us-west-2"
})
```

Puntos de conexión

El paquete de [puntos de conexión](#) ya no existe en AWS SDK para Go. Cada cliente de servicio ahora incorpora los metadatos de AWS punto final necesarios en el paquete del cliente. Esto reduce el tamaño binario global de las aplicaciones compiladas al dejar de incluir los metadatos de punto de conexión de los servicios que la aplicación no utiliza.

Además, ahora cada servicio expone su propia interfaz para la resolución de puntos de conexión en `EndpointResolverV2`. Cada API toma un conjunto único de parámetros para un servicio `EndpointParameters`, cuyos valores los obtiene el SDK de varias ubicaciones al invocarse una operación.

De forma predeterminada, los clientes de servicio utilizan su AWS región configurada para resolver el punto final del servicio en la región de destino. Si su aplicación requiere un punto de conexión personalizado, puede especificar un comportamiento personalizado en el campo `EndpointResolverV2` de la estructura `aws.Config`. Si la aplicación implementa un objeto [endpoints.Resolver](#) personalizado, debe migrarlo para que se ajuste a esta nueva interfaz por servicio.

Para obtener más información acerca de los puntos de conexión y de cómo implementar un solucionador personalizado, consulte [Configuración de puntos de conexión del cliente](#).

Autenticación

AWS SDK para Go admite un comportamiento de autenticación más avanzado, lo que permite el uso de funciones de AWS servicio más recientes, como codecatalyst y S3 Express One Zone. Además, este comportamiento se puede personalizar para cada cliente.

Invocación de operaciones de la API

La cantidad de métodos de operación del cliente de servicio se ha reducido significativamente. Los métodos `<OperationName>Request`, `<OperationName>WithContext` y `<OperationName>` se han consolidado en un solo método de operación, `<OperationName>`.

Ejemplo

El siguiente ejemplo muestra cómo se migrarían las llamadas a la `PutObject` operación Amazon S3 de la AWS SDK para Go versión 1 a la versión 2.

```
// V1

import "context"
import "github.com/aws/aws-sdk-go/service/s3"

// ...

client := s3.New(sess)

// Pattern 1
output, err := client.PutObject(&s3.PutObjectInput{
    // input parameters
})

// Pattern 2
output, err := client.PutObjectWithContext(context.TODO(), &s3.PutObjectInput{
    // input parameters
})

// Pattern 3
req, output := client.PutObjectRequest(context.TODO(), &s3.PutObjectInput{
    // input parameters
})
err := req.Send()
```



```
// V2

import "context"
import "github.com/aws/aws-sdk-go-v2/service/s3"

// ...

client := s3.NewFromConfig(cfg)

output, err := client.PutObject(context.TODO(), &s3.PutObjectInput{
    // input parameters
})
```

Tipos de datos de servicio

Los tipos de entrada y salida de nivel superior de una operación se encuentran en el paquete de clientes de servicio. El tipo de entrada y salida de una operación determinada sigue el patrón de `<OperationName>Input` y `<OperationName>Output`, donde `OperationName` es el nombre de la operación que invoca. Por ejemplo, las formas de entrada y salida de la `PutObject` operación de Amazon S3 son [PutObjectInput](#) y [PutObjectOutput](#), respectivamente.

Todos los demás tipos de datos de servicio, excepto los tipos de entrada y salida, se han migrado al paquete `types`, ubicado en la jerarquía de rutas de importación del paquete de clientes de servicio. Por ejemplo, el [s3. AccessControlPolicy](#) el tipo ahora se encuentra en [types. AccessControlPolicy](#).

Valores de enumeración

Ahora el SDK proporciona una experiencia con tipos para todos los campos de enumeración de las API. En lugar de usar un valor literal de cadena copiado de la documentación de referencia de la API del servicio, ahora puede usar uno de los tipos concretos que se encuentran en el paquete `types` del cliente de servicio. Por ejemplo, puede proporcionar a la `PutObject` operación Amazon S3 una ACL para aplicarla a un objeto. En la AWS SDK para Go versión 1, este parámetro era un `*string` tipo. En el AWS SDK para Go, este parámetro ahora es un [tipo. ObjectCannedACL](#). El paquete `types` proporciona constantes generadas para los valores de enumeración válidos que pueden asignarse a este campo. Por ejemplo, [tipos. ObjectCannedACLPrivate](#) es la constante del valor de la ACL predefinida «privada». Este valor se puede usar en lugar de administrar constantes de cadena en la aplicación.

Parámetros de puntero

La AWS SDK para Go versión 1 exigía que se pasaran referencias de puntero para todos los parámetros de entrada a las operaciones de servicio. La AWS SDK para Go versión 2 ha simplificado la experiencia con la mayoría de los servicios al eliminar la necesidad de pasar los valores de entrada como punteros siempre que es posible. Este cambio significa que muchas de las operaciones de clientes de servicio ya no requieren que la aplicación pase referencias de puntero para los tipos siguientes: `uint8`, `uint16`, `uint32`, `int8`, `int16`, `int32`, `float32`, `float64` y `bool`. Del mismo modo, los tipos de elementos de sector y mapa se han actualizado en consecuencia para reflejar si sus elementos deben pasarse como referencias del puntero.

El paquete [aws](#) contiene funciones auxiliares destinadas a crear punteros para los tipos integrados de Go. Estas funciones deben utilizarse con el fin de gestionar la creación de tipos de puntero para estos tipos Go de forma más fácil. Del mismo modo, se proporcionan métodos auxiliares para eliminar de forma segura toda referencia a los valores de puntero de estos tipos. Por ejemplo, la función [aws.String](#) convierte de `string` a `*string`. [A la inversa, las leyes. ToString](#) convierte desde `*string` a `string`. Al actualizar su aplicación de AWS SDK para Go v1 a la versión 2, debe migrar el uso de los elementos auxiliares para convertir de los tipos de puntero a las variantes sin puntero. Por ejemplo, [aws.StringValue](#) debe actualizarse a `aws.ToString`.

Tipos de errores

AWS SDK para Go Aprovecha al máximo la funcionalidad de corrección de errores [introducida en Go 1.13](#). Los servicios que modelan respuestas de error han generado tipos disponibles en el paquete `types` del cliente que pueden utilizarse para probar si alguno de estos tipos ha sido responsable de provocar un error de operación de cliente. Por ejemplo, la operación `GetObject` de Amazon S3 puede devolver un error `NoSuchKey` si se intenta recuperar una clave de objeto que no existe. [Puedes usar Errors.as para comprobar si el error de operación devuelto es un tipo. NoSuchKey](#) error. En el caso de que un servicio no modele un tipo específico para un error, puede utilizar la [herramienta. APIError](#) tipo de interfaz para inspeccionar el código de error y el mensaje devueltos por el servicio. Esta funcionalidad reemplaza a [Aserr.error](#) y a las demás funciones de [awserr](#) de la versión 1. AWS SDK para Go Para obtener más información sobre la gestión de errores, consulte [Gestión de errores en AWS SDK para Go V2](#).

Ejemplo

```
// v1
```

```

import "github.com/aws/aws-sdk-go/aws/awserr"
import "github.com/aws/aws-sdk-go/service/s3"

// ...

client := s3.New(sess)

output, err := s3.GetObject(&s3.GetObjectInput{
    // input parameters
})
if err != nil {
    if awsErr, ok := err.(awserr.Error); ok {
        if awsErr.Code() == "NoSuchKey" {
            // handle NoSuchKey
        } else {
            // handle other codes
        }
        return
    }
    // handle a error
}

```

```

// V2

import "context"
import "github.com/aws/aws-sdk-go-v2/service/s3"
import "github.com/aws/aws-sdk-go-v2/service/s3/types"
import "github.com/aws/smithy-go"

// ...

client := s3.NewFromConfig(cfg)

output, err := client.GetObject(context.TODO(), &s3.GetObjectInput{
    // input parameters
})
if err != nil {
    var nsd *types.NoSuchKey
    if errors.As(err, &nsd) {
        // handle NoSuchKey error
        return
    }
    var apiErr smithy.APIError

```

```
if errors.As(err, &apiErr) {
    code := apiErr.ErrorCode()
    message := apiErr.ErrorMessage()
    // handle error code
    return
}
// handle error
return
}
```

Paginadores

Los paginadores de operaciones de servicio ya no se invocan como métodos en el cliente de servicio. A fin de usar un paginador para una operación, debe crear un paginador con este fin mediante uno de los métodos del creador de paginadores. [Por ejemplo, para usar `paginate` sobre la `ListObjectsV2` operación Amazon S3, debe construir su paginador con `s3.NewListObjectsV2Paginator`](#). Este constructor devuelve un [ListObjectsV2Paginator](#) que proporciona los métodos `HasMorePages` y permite determinar si hay más páginas que recuperar e `NextPage` invocar la operación para recuperar la página siguiente, respectivamente. En [Uso de paginadores de operaciones](#) encontrará más detalles sobre el uso de los paginadores del SDK.

Veamos un ejemplo de cómo migrar de un paginador de la versión 1 al equivalente de la AWS SDK para Go versión 2. AWS SDK para Go

Ejemplo

```
// V1

import "fmt"
import "github.com/aws/aws-sdk-go/service/s3"

// ...

client := s3.New(sess)

params := &s3.ListObjectsV2Input{
    // input parameters
}

totalObjects := 0
err := client.ListObjectsV2Pages(params, func(output *s3.ListObjectsV2Output, lastPage bool) bool {
```

```

    totalObjects += len(output.Contents)
    return !lastPage
})
if err != nil {
    // handle error
}
fmt.Println("total objects:", totalObjects)

```

```

// V2

import "context"
import "fmt"
import "github.com/aws/aws-sdk-go-v2/service/s3"

// ...

client := s3.NewFromConfig(cfg)

params := &s3.ListObjectsV2Input{
    // input parameters
}

totalObjects := 0
paginator := s3.NewListObjectsV2Paginator(client, params)
for paginator.HasMorePages() {
    output, err := paginator.NextPage(context.TODO())
    if err != nil {
        // handle error
    }
    totalObjects += len(output.Contents)
}
fmt.Println("total objects:", totalObjects)

```

Waiters

Los esperadores de operaciones de servicio ya no se invocan como métodos en el cliente de servicio. Para utilizar un esperador, primero debe crear el tipo de esperador deseado y, a continuación, invocar el método de espera. Por ejemplo, para esperar a que exista un bucket de Amazon S3, debe crear un esperador `BucketExists`. [Usa el `s3.NewBucketExistsWaiter`](#) constructor para crear un [`s3.BucketExistsWaiter`](#). `s3.BucketExistsWaiter` proporciona un método `Wait` que se puede utilizar para esperar a que un bucket se encuentre disponible.

Solicitudes prefiradas

Técnicamente, el SDK de la versión V1 permitía prefirar cualquier operación del AWS SDK; sin embargo, esto no representa con precisión lo que realmente se admite a nivel de servicio (y, en realidad, la mayoría de las operaciones de AWS servicio no admiten la prefirma).

AWS SDK para Go resuelve este problema exponiendo `PresignClient` implementaciones específicas en paquetes de servicios con operaciones predefinidas específicas APIs para cada tipo de soporte.

Nota: Si a un servicio no se le permite prefirar una operación que estabas utilizando correctamente en el SDK v1, comunícanoslo mediante la notificación de problemas. [GitHub](#)

Utiliza [Presign](#) y [PresignRequest](#) debe convertirse para utilizar clientes de presignación específicos del servicio.

El siguiente ejemplo muestra cómo migrar la prefirma de una solicitud de S3: `GetObject`

```
// V1

import (
    "fmt"
    "time"

    "github.com/aws/aws-sdk-go/aws"
    "github.com/aws/aws-sdk-go/aws/session"
    "github.com/aws/aws-sdk-go/service/s3"
)

func main() {
    sess := session.Must(session.NewSessionWithOptions(session.Options{
        SharedConfigState: session.SharedConfigEnable,
    }))

    svc := s3.New(sess)
    req, _ := svc.GetObjectRequest(&s3.GetObjectInput{
        Bucket: aws.String("amzn-s3-demo-bucket"),
        Key:    aws.String("key"),
    })

    // pattern 1
    url1, err := req.Presign(20 * time.Minute)
    if err != nil {
```

```
        panic(err)
    }
    fmt.Println(url1)

    // pattern 2
    url2, header, err := req.PresignRequest(20 * time.Minute)
    if err != nil {
        panic(err)
    }
    fmt.Println(url2, header)
}
```

```
// V2

import (
    "context"
    "fmt"
    "time"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"
)

func main() {
    cfg, err := config.LoadDefaultConfig(context.Background())
    if err != nil {
        panic(err)
    }

    svc := s3.NewPresignClient(s3.NewFromConfig(cfg))
    req, err := svc.PresignGetObject(context.Background(), &s3.GetObjectInput{
        Bucket: aws.String("amzn-s3-demo-bucket"),
        Key:    aws.String("key"),
    }, func(o *s3.PresignOptions) {
        o.Expires = 20 * time.Minute
    })
    if err != nil {
        panic(err)
    }

    fmt.Println(req.Method, req.URL, req.SignedHeader)
}
```

Personalización de solicitudes

La API [request.Request](#) monolítica se ha vuelto a compartimentar.

Entrada/salida de la operación

Ahora se puede acceder a los campos `Request` opacos `Params` y `Data`, que contienen las estructuras de entrada y salida de la operación, respectivamente, en fases del middleware específicas como entrada/salida:

Los controladores de solicitudes que hagan referencia a `Request.Params` y `Request.Data` deben migrarse al middleware.

migración de **Params**

```
// V1

import (
    "github.com/aws/aws-sdk-go/aws"
    "github.com/aws/aws-sdk-go/aws/request"
    "github.com/aws/aws-sdk-go/aws/session"
    "github.com/aws/aws-sdk-go/service/s3"
)

func withPutObjectDefaultACL(acl string) request.Option {
    return func(r *request.Request) {
        in, ok := r.Params.(*s3.PutObjectInput)
        if !ok {
            return
        }

        if in.ACL == nil {
            in.ACL = aws.String(acl)
        }
        r.Params = in
    }
}

func main() {
    sess := session.Must(session.NewSession())

    sess.Handlers.Validate.PushBack(withPutObjectDefaultACL(s3.ObjectCannedACLBucketOwnerFullControl))
}
```



```
// ...
}
```

```
// V2

import (
    "context"

    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go/middleware"
    smithyhttp "github.com/aws/smithy-go/transport/http"
)

type withPutObjectDefaultACL struct {
    acl types.ObjectCannedACL
}

// implements middleware.InitializeMiddleware, which runs BEFORE a request has
// been serialized and can act on the operation input
var _ middleware.InitializeMiddleware = (*withPutObjectDefaultACL)(nil)

func (*withPutObjectDefaultACL) ID() string {
    return "withPutObjectDefaultACL"
}

func (m *withPutObjectDefaultACL) HandleInitialize(ctx context.Context, in
    middleware.InitializeInput, next middleware.InitializeHandler) (
    out middleware.InitializeOutput, metadata middleware.Metadata, err error,
) {
    input, ok := in.Parameters.(*s3.PutObjectInput)
    if !ok {
        return next.HandleInitialize(ctx, in)
    }

    if len(input.ACL) == 0 {
        input.ACL = m.acl
    }
    in.Parameters = input
    return next.HandleInitialize(ctx, in)
}
```

```
// create a helper function to simplify instrumentation of our middleware
func WithPutObjectDefaultACL(acl types.ObjectCannedACL) func (*s3.Options) {
    return func(o *s3.Options) {
        o.APIOptions = append(o.APIOptions, func (s *middleware.Stack) error {
            return s.Initialize.Add(&withPutObjectDefaultACL{acl: acl},
                middleware.After)
        })
    }
}

func main() {
    cfg, err := config.LoadDefaultConfig(context.Background())
    if err != nil {
        // ...
    }

    svc := s3.NewFromConfig(cfg,
        WithPutObjectDefaultACL(types.ObjectCannedACLBucketOwnerFullControl))
    // ...
}
```

migración de **Data**

```
// V1

import (
    "github.com/aws/aws-sdk-go/aws"
    "github.com/aws/aws-sdk-go/aws/request"
    "github.com/aws/aws-sdk-go/aws/session"
    "github.com/aws/aws-sdk-go/service/s3"
)

func readPutObjectOutput(r *request.Request) {
    output, ok := r.Data.(*s3.PutObjectOutput)
    if !ok {
        return
    }

    // ...
}

func main() {
```

```

sess := session.Must(session.NewSession())
sess.Handlers.Unmarshal.PushBack(readPutObjectOutput)

svc := s3.New(sess)
// ...
}

```

```

// V2

import (
    "context"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/smithy-go/middleware"
    smithyhttp "github.com/aws/smithy-go/transport/http"
)

type readPutObjectOutput struct{}

var _ middleware.DeserializeMiddleware = (*readPutObjectOutput)(nil)

func (*readPutObjectOutput) ID() string {
    return "readPutObjectOutput"
}

func (*readPutObjectOutput) HandleDeserialize(ctx context.Context, in
    middleware.DeserializeInput, next middleware.DeserializeHandler) (
    out middleware.DeserializeOutput, metadata middleware.Metadata, err error,
) {
    out, metadata, err = next.HandleDeserialize(ctx, in)
    if err != nil {
        // ...
    }

    output, ok := in.Parameters.(*s3.PutObjectOutput)
    if !ok {
        return out, metadata, err
    }

    // inspect output...

    return out, metadata, err
}

```

```

}

func WithReadPutObjectOutput(o *s3.Options) {
    o.APIOptions = append(o.APIOptions, func (s *middleware.Stack) error {
        return s.Initialize.Add(&withReadPutObjectOutput{}, middleware.Before)
    })
}

func main() {
    cfg, err := config.LoadDefaultConfig(context.Background())
    if err != nil {
        // ...
    }

    svc := s3.NewFromConfig(cfg, WithReadPutObjectOutput)
    // ...
}

```

Solicitud/respuesta HTTP

Ahora, los campos `HTTPRequest` y `HTTPResponse` de `Request` se exponen en fases específicas del middleware. Como el middleware es independiente del transporte, debe realizar una aserción de tipo en la entrada o salida del middleware para revelar la solicitud o la respuesta HTTP subyacente.

Los controladores de solicitudes que hagan referencia a `Request.HTTPRequest` y `Request.HTTPResponse` deben migrarse al middleware.

migración de `HTTPRequest`

```

// V1

import (
    "github.com/aws/aws-sdk-go/aws/request"
    "github.com/aws/aws-sdk-go/aws/session"
)

func withHeader(header, val string) request.Option {
    return func(r *request.Request) {
        request.HTTPRequest.Header.Set(header, val)
    }
}

```

```
func main() {
    sess := session.Must(session.NewSession())
    sess.Handlers.Build.PushBack(withHeader("x-user-header", "..."))

    svc := s3.New(sess)
    // ...
}
```

```
// V2
```

```
import (
    "context"
    "fmt"

    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/smithy-go/middleware"
    smithyhttp "github.com/aws/smithy-go/transport/http"
)

type withHeader struct {
    header, val string
}

// implements middleware.BuildMiddleware, which runs AFTER a request has been
// serialized and can operate on the transport request
var _ middleware.BuildMiddleware = (*withHeader)(nil)

func (*withHeader) ID() string {
    return "withHeader"
}

func (m *withHeader) HandleBuild(ctx context.Context, in middleware.BuildInput, next
    middleware.BuildHandler) (
    out middleware.BuildOutput, metadata middleware.Metadata, err error,
) {
    req, ok := in.Request.(*smithyhttp.Request)
    if !ok {
        return out, metadata, fmt.Errorf("unrecognized transport type %T", in.Request)
    }

    req.Header.Set(m.header, m.val)
    return next.HandleBuild(ctx, in)
```

```

}

func WithHeader(header, val string) func (*s3.Options) {
    return func(o *s3.Options) {
        o.APIOptions = append(o.APIOptions, func (s *middleware.Stack) error {
            return s.Build.Add(&withHeader{
                header: header,
                val: val,
            }, middleware.After)
        })
    }
}

func main() {
    cfg, err := config.LoadDefaultConfig(context.Background())
    if err != nil {
        // ...
    }

    svc := s3.NewFromConfig(cfg, WithHeader("x-user-header", "..."))
    // ...
}

```

Fases del controlador

Las fases del middleware del SDK v2 son las sucesoras de las fases del controlador de la versión 1.

En la tabla siguiente se proporciona una correspondencia aproximada entre las fases del controlador de la versión 1 y su ubicación equivalente en la pila de middleware de la versión 2:

Nombre del controlador de v1	Fase de middleware de v2
Valide	Initialize
Build	Serialize
Sign	Finalize
Send	n/a (1)
ValidateResponse	Deserialize

Nombre del controlador de v1	Fase de middleware de v2
Unmarshal	Deserialize
UnmarshalMetadata	Deserialize
UnmarshalError	Deserialize
Retry	Finalize, posterior al middleware "Retry" (2)
AfterRetry	Finalize, anterior al middleware "Retry", posterior a <code>next.HandleFinalize()</code> (2, 3)
CompleteAttempt	Finalize, final del paso
Completado	Initialize, comienzo del paso, posterior a <code>next.HandleInitialize()</code> (3)

(1) La fase Send de la versión 1 es, en efecto, el ciclo de ida y vuelta del cliente HTTP encapsulado de la versión 2. El campo `HTTPClient` controla este comportamiento en las opciones de cliente.

(2) Cualquier middleware posterior al middleware "Retry" del paso Finalize formará parte del bucle de reintentos.

(3) Durante la operación, la “pila” de middleware se integra en una función de controlador repetidamente decorada. Cada controlador es responsable de llamar al siguiente de la cadena. Esto significa implícitamente que un paso de middleware también puede actuar DESPUÉS de que se haya llamado a su siguiente paso.

Por ejemplo, en el caso del paso Initialize, que se encuentra en la parte superior de la pila, significa que los middlewares de Initialize que actúan tras llamar al siguiente controlador funcionan de manera efectiva al final de la solicitud:

```
// V2

import (
    "context"

    "github.com/aws/smithy-go/middleware"
```

```

)

type onComplete struct{}

var _ middleware.InitializeMiddleware = (*onComplete)(nil)

func (*onComplete) ID() string {
    return "onComplete"
}

func (*onComplete) HandleInitialize(ctx context.Context, in middleware.InitializeInput,
    next middleware.InitializeHandler) (
    out middleware.InitializeOutput, metadata middleware.Metadata, err error,
) {
    out, metadata, err = next.HandleInitialize(ctx, in)

    // the entire operation was invoked above - the deserialized response is
    // available opaquely in out.Result, run post-op actions here...

    return out, metadata, err
}

```

Características

Servicio de metadatos de EC2 instancias de Amazon

AWS SDK para Go Proporciona un cliente de Amazon EC2 Instance Metadata Service (IMDS) que puede utilizar para consultar el IMDS local al ejecutar la aplicación en una instancia de Amazon EC2 . El cliente IMDS es un módulo de Go independiente que se puede agregar a la aplicación mediante:

```
go get github.com/aws/aws-sdk-go-v2/feature/ec2/imds
```

El creador de clientes y las operaciones de método se han actualizado para ajustarse al diseño de los demás clientes de servicio del SDK.

Ejemplo

```
// V1
```



```
import "github.com/aws/aws-sdk-go/aws/ec2metadata"

// ...

client := ec2metadata.New(sess)

region, err := client.Region()
if err != nil {
    // handle error
}
```

```
// V2

import "context"
import "github.com/aws/aws-sdk-go-v2/feature/ec2/imds"

// ...

client := imds.NewFromConfig(cfg)

region, err := client.GetRegion(context.TODO())
if err != nil {
    // handle error
}
```

Amazon S3 Transfer Manager

El administrador de transferencias de Amazon S3 está disponible para administrar cargas y descargas de objetos de forma simultánea. Este paquete se encuentra en un módulo de Go fuera de la ruta de importación del cliente de servicio. Este módulo se puede recuperar mediante `go get github.com/aws/aws-sdk-go-v2/feature/s3/manager`.

[s3. NewUploader](#) y [s3. NewUploaderWithClient](#) han sido reemplazados por el [administrador de métodos del constructor. NewUploader](#) para crear un cliente de gestión de subidas.

[s3. NewDownloader](#) y [s3. NewDownloaderWithClient](#) han sido reemplazados por un único [administrador de métodos de construcción. NewDownloader](#) para crear un cliente gestor de descargas.

Utilidades de CloudFront firma de Amazon

AWS SDK para Go Proporciona las utilidades de CloudFront firma de Amazon en un módulo Go fuera de la ruta de importación del cliente del servicio. Este módulo se puede recuperar mediante `go get`.

```
go get github.com/aws/aws-sdk-go-v2/feature/cloudfront/sign
```

Cliente de cifrado de Amazon S3

A partir de AWS SDK para Go ahora, el cliente de cifrado Amazon S3 es un módulo independiente de [AWS Crypto Tools](#). La última versión del cliente de cifrado S3 para Go, 3.x, ya está disponible en [GitHub](#). Este módulo se puede recuperar mediante `go get`:

```
go get github.com/aws/amazon-s3-encryption-client-go/v3
```

Las versiones independientes `EncryptionClient` ([v1](#), [v2](#)) y `DecryptionClient` ([v1](#), [v2](#)) se APIs han sustituido por un único cliente, el [S3 EncryptionClient V3](#), que ofrece funciones tanto de cifrado como de descifrado.

Al igual que otros clientes de servicios AWS SDK para Go, la operación APIs se ha reducido:

- El `GetObjectGetObjectRequest`, y el `GetObjectWithContext` descifrado APIs se sustituyen por [GetObject](#).
- El `PutObjectPutObjectRequest`, y el `PutObjectWithContext` cifrado APIs se sustituyen por [PutObject](#).

Para obtener información sobre cómo migrar a la versión principal del cliente de cifrado (3.x), consulte [esta guía](#).

Cambios en las personalizaciones de servicios

Amazon S3

Al migrar de la versión AWS SDK para Go 1 a la versión 2, un cambio importante que hay que tener en cuenta es el manejo del material `SSECustomerKey` utilizado para el cifrado del lado del servidor con claves proporcionadas por el cliente (SSE-C). En la AWS SDK para Go versión 1, el

SDK gestionaba internamente la codificación SSECustomerKey para Base64. En el SDK v2, esta codificación automática se ha eliminado y ahora es necesario codificar SSECustomerKey de forma manual en Base64 antes de pasarlo al SDK.

Ajuste de ejemplo:

```
// V1

import (
    "context"
    "encoding/base64"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/service/s3"
)
// ... more code

plainTextKey := "12345678901234567890123456789012" // 32 bytes in length

// calculate md5..

_, err = client.PutObjectWithContext(context.Background(), &s3.PutObjectInput{
    Bucket: aws.String("amzn-s3-demo-bucket"),
    Key:    aws.String("your-object-key"),
    Body:   strings.NewReader("hello-world"),
    SSECustomerKey:      &plainTextKey,
    SSECustomerKeyMD5:   &base64Md5,
    SSECustomerAlgorithm: aws.String("AES256"),
})

// ... more code
```

```
// V2

import (
    "github.com/aws/aws-sdk-go/aws"
    "github.com/aws/aws-sdk-go/aws/session"
    "github.com/aws/aws-sdk-go/service/s3"
)

// ... more code

plainTextKey := "12345678901234567890123456789012" // 32 bytes in length
base64EncodedKey := base64.StdEncoding.EncodeToString([]byte(plainTextKey))
```

```
// calculate md5..  
  
_, err = client.PutObject(context.Background(), &s3.PutObjectInput{  
    Bucket: aws.String("amzn-s3-demo-bucket"),  
    Key:    aws.String("your-object-key"),  
    Body:   strings.NewReader("hello-world"),  
    SSECustomerKey: &base64EncodedKey,  
    SSECustomerKeyMD5: &base64Md5,  
    SSECustomerAlgorithm: aws.String("AES256"),  
})  
  
// ... more code
```

Seguridad en AWS SDK para Go

La seguridad en AWS es la principal prioridad. Como cliente de AWS, se beneficiará de una arquitectura de red y de centros de datos diseñados para satisfacer los requisitos de seguridad de las organizaciones más exigentes.

La seguridad es una responsabilidad compartida entre AWS y usted. El [modelo de responsabilidad compartida](#) la describe como seguridad de la nube y seguridad en la nube:

- Seguridad de la nube: AWS es responsable de proteger la infraestructura que ejecuta servicios de AWS en la Nube de AWS. AWS también le proporciona servicios que puede utilizar de forma segura. Los auditores externos prueban y verifican periódicamente la eficacia de nuestra seguridad como parte de los [Programas de conformidad de AWS](#). Para obtener información sobre los programas de conformidad que se aplican a AWS SDK para Go, consulte [AWS Services in Scope by Compliance Program](#).
- Seguridad en la nube: su responsabilidad viene determinada por el servicio de AWS que utilice. También es responsable de otros factores, incluida la confidencialidad de los datos, los requisitos de la empresa y la legislación y la normativa aplicables.

Esta documentación le ayuda a comprender cómo aplicar el modelo de responsabilidad compartida a la hora de utilizar AWS SDK para Go. Los siguientes temas le mostrarán cómo configurar AWS SDK para Go para satisfacer sus objetivos de seguridad y de conformidad. También puede aprender a utilizar otros servicios de AWS que lo ayuden a monitorizar y proteger los recursos de AWS SDK para Go.

Temas

- [Protección de los datos en AWS SDK para Go](#)
- [Validación de conformidad en AWS SDK para Go](#)
- [Resiliencia en AWS SDK para Go](#)

Protección de los datos en AWS SDK para Go

El [modelo de responsabilidad compartida](#) de AWS se aplica a la protección de datos de AWS SDK para Go. 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. Esto incluye las situaciones en las que debe trabajar con la AWS SDK para Go u otros Servicios de AWS a través de la consola, la API, la AWS CLI o los 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.

Validación de conformidad en AWS SDK para Go

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](#).

Resiliencia en AWS SDK para Go

La infraestructura global de AWS se divide en 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 conmutan automáticamente entre zonas sin interrupción. Las zonas de disponibilidad tienen una mayor disponibilidad, tolerancia a errores y escalabilidad que las infraestructuras tradicionales de uno o varios centros de datos.

Para obtener más información sobre las Regiones de AWS y las zonas de disponibilidad, consulte la [Infraestructura global de AWS](#).

Historial de documentos de la Guía para desarrolladores de AWS SDK para Go v2

En la tabla siguiente se describen las versiones de la documentación de AWS SDK para Go v2.

Cambio	Descripción	Fecha
Interceptores HTTP	Nueva sección sobre los interceptores HTTP, sus ventajas sobre el middleware, los enlaces disponibles y los métodos de registro.	4 de septiembre de 2025
Protección de la integridad de los datos con sumas de comprobación	Se ha actualizado el contenido con detalles sobre el cálculo automático de sumas de comprobación.	16 de enero de 2025
Versión inicial	Versión inicial de la Guía para desarrolladores de AWS SDK para Go v2	31 de octubre de 2024

Las traducciones son generadas a través de traducción automática. En caso de conflicto entre la traducción y la version original de inglés, prevalecerá la version en inglés.