aws

Panduan Developerr

AWS SDK for Go v2

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

AWS SDK for Go v2: Panduan Developerr

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

Merek dagang dan tampilan dagang Amazon tidak boleh digunakan sehubungan dengan produk atau layanan apa pun yang bukan milik Amazon, dengan cara apa pun yang dapat menyebabkan kebingungan di antara pelanggan, atau dengan cara apa pun yang merendahkan atau mendiskreditkan Amazon. Semua merek dagang lain yang tidak dimiliki oleh Amazon merupakan hak milik masing-masing pemiliknya, yang mungkin atau mungkin tidak terafiliasi, terkait dengan, atau disponsori oleh Amazon.

Table of Contents

Apa itu AWS SDK for Go v2? 1
Pemeliharaan dan dukungan untuk versi utama SDK 1
Memulai 2
Dapatkan Akun Amazon 2
Instal AWS SDK for Go v2 2
Dapatkan kunci AWS akses Anda 3
Untuk mendapatkan ID kunci akses dan kunci akses rahasia Anda. 3
Topik terkait 4
Memanggil Operasi 4
Konfigurasikan SDK 6
Memuat File Konfigurasi AWS Bersama 6
Menentukan Wilayah AWS 7
Konfigurasikan Wilayah dengan Variabel Lingkungan 7
Tentukan Wilayah Secara Terprogram 7
Menentukan Kredensial 7
Peran IAM untuk Tugas 9
Peran IAM untuk Instans Amazon EC2 9
Kredensi dan Konfigurasi Bersama 9
Variabel lingkungan 12
Tentukan Kredensial Secara Terprogram 13
Autentikasi 17
Definisi 17
Alur kerja resolusi skema autentikasi 20
S yang didukung secara asli `AuthScheme` 21
Titik Akhir Klien 24
Kustomisasi 24
V2: `EndpointResolverV2` + `BaseEndpoint` 24
V1: `EndpointResolver` 28
Migrasi 31
Klien HTTP 35
Mengganti Selama Pemuatan Konfigurasi 35
Waktu habis 36
Dialer 36
Transportasi 37

Pencegat 40
Pencegat vs middleware 40
Kait pencegat yang tersedia 41
Pendaftaran pencegat 42
Konfigurasi pencegat global 43
Pencatatan log 43
Pencatat 43
ClientLogMode 44
Coba Ulang dan Batas Waktu 44
Retryer Standar 45
NopRetryer 45
Menyesuaikan Perilaku 45
Timeout 50
Menggunakan SDK 52
Membangun Klien Layanan 52
NewFromConfig 52
Baru 53
Operasi Layanan Panggilan 55
Melewati Parameter ke Operasi Layanan 55
Mengganti Opsi Klien Untuk Panggilan Operasi 56
Menangani Tanggapan Operasi 57
Serentak Menggunakan Klien Layanan 59
Menggunakan Operation Paginators 61
Menggunakan Pelayan 63
Mengesampingkan konfigurasi pelayan 64
Penggantian konfigurasi pelayan tingkat lanjut 66
Menangani Kesalahan dalam SDK 67
Kesalahan Pencatatan 67
Kesalahan Layanan Klien 68
Mengambil Pengidentifikasi Permintaan 69
Gunakan AWS layanan 71
Checksum Amazon S3 71
Mengunggah objek 72
Unduh objek 75
Utilitas SDK 76
Utilitas Amazon RDS 76

IAM Authentication. 76
CloudFront Utilitas Amazon 77
Penandatangan CloudFront URL Amazon 77
Layanan EC2 Metadata Instans Amazon 78
Utilitas Amazon S3 79
Manajer Transfer Amazon S3 79
Masukan Streaming yang Tidak Dapat Dicari 89
Middleware 91
Menulis Middleware Kustom 92
Melampirkan Middleware ke Semua Klien 94
Melampirkan Middleware ke Operasi Tertentu 94
Melewati Metadata ke Tumpukan 95
Metadata Disediakan oleh SDK 97
Melewati Metadata Ke Tumpukan 97
Pertanyaan yang Sering Diajukan 100
Bagaimana cara mengkonfigurasi klien HTTP SDK saya? Apakah ada pedoman atau praktik terbaik? 100
Bagaimana saya harus mengonfigurasi batas waktu operasi? 100
Permintaan yang dibuat oleh SDK habis waktu atau terlalu lama, bagaimana cara memperbaikinya? 101
Bagaimana cara memperbaiki `read: connection reset` kesalahan? 101
Mengapa saya mendapatkan kesalahan “tanda tangan tidak valid” saat menggunakan proxy HTTP dengan SDK? 102
Pengaturan waktu operasi SDK 102
Tes Unit 109
Mengejek Operasi Klien 109
Mengejek Paginator 111
Contoh kode 114
API Gateway 115
AWS kontribusi komunitas 115
Aurora 116
Memulai 116
Hal-hal mendasar 118
Tindakan 136
Amazon Bedrock 156
Memulai 116

Tindakan 136
Runtime Amazon Bedrock 159
Memulai 116
Skenario 162
Generator Gambar Amazon Titan 166
Antropik Claude 168
CloudFormation 176
Tindakan 136
CloudWatch Log 177
Tindakan 136
Penyedia Identitas Amazon Cognito 180
Memulai 116
Tindakan 136
Skenario 162
Amazon DocumentDB 262
Contoh nirserver 262
DynamoDB 264
Hal-hal mendasar 118
Tindakan 136
Skenario 162
Contoh nirserver 262
AWS kontribusi komunitas 115
IAM 337
Memulai 116
Hal-hal mendasar 118
Tindakan 136
Kinesis 397
Contoh nirserver 262
Lambda 400
Memulai 116
Hal-hal mendasar 118
Tindakan 136
Skenario 162
Contoh nirserver 262
AWS kontribusi komunitas 115
Amazon MSK 512

Contoh nirserver 262
Pusat Partner 514
Tindakan 136
Amazon RDS 517
Memulai 116
Hal-hal mendasar 118
Tindakan 136
Contoh nirserver 262
Amazon Redshift 552
Memulai 116
Hal-hal mendasar 118
Tindakan 136
Amazon S3 571
Memulai 116
Hal-hal mendasar 118
Tindakan 136
Skenario 162
Contoh nirserver 262
Amazon SNS 660
Memulai 116
Tindakan 136
Skenario 162
Contoh nirserver 262
Amazon SQS 689
Memulai 116
Tindakan 136
Skenario 162
Contoh nirserver 262
Migrasikan ke v2 721
Versi Go Minimum 721
Modularisasi 721
Pemuatan Konfigurasi 722
Contoh 26
Mengejek dan `*iface` 725
Penyedia Kredensyal dan Kredensyal 726
Kredensyal Statis 14

Kredensyal Peran Amazon EC2 IAM 728
Kredensyal Endpoint 729
Kredensyal Proses 729
AWS Security Token Service Kredensyal 730
Layanan Klien 732
Konstruksi Klien 733
Titik akhir 735
Autentikasi 735
Memanggil Operasi API 736
Jenis Data Layanan 737
Parameter Pointer 737
Jenis Kesalahan 738
Paginator 740
Pelayan 741
Permintaan yang Ditandatangani 741
Minta kustomisasi 743
Input/output operasi 744
Permintaan/tanggapan HTTP 748
Fase handler 750
Fitur 752
Layanan EC2 Metadata Instans Amazon 752
Manajer Transfer Amazon S3 753
Utilitas CloudFront Penandatanganan Amazon 753
Klien Enkripsi Amazon S3 753
Perubahan Kustomisasi Layanan 754
Amazon S3 754
Keamanan 756
Perlindungan data 756
Validasi kepatuhan 758
Ketahanan 758
Riwayat dokumen 759
..... dcclx

Apa itu AWS SDK for Go v2?

AWS SDK for Go V2 menyediakan APIs dan utilitas yang dapat digunakan pengembang untuk membangun aplikasi Go yang menggunakan AWS layanan, seperti Amazon Elastic Compute Cloud (Amazon EC2) dan Amazon Simple Storage Service (Amazon S3).

SDK menghilangkan kompleksitas pengkodean langsung terhadap antarmuka layanan web. Ini menyembunyikan banyak pipa ledeng tingkat rendah, seperti otentikasi, percobaan ulang permintaan, dan penanganan kesalahan.

SDK juga mencakup utilitas yang bermanfaat. Misalnya, pengelola unduhan dan unggah Amazon S3 dapat secara otomatis memecah objek besar menjadi beberapa bagian dan mentransfernya secara paralel.

Gunakan Panduan AWS SDK for Go Pengembang untuk membantu Anda menginstal, mengonfigurasi, dan menggunakan SDK. Panduan ini menyediakan informasi konfigurasi, kode contoh, dan pengantar utilitas SDK.

Pemeliharaan dan dukungan untuk versi utama SDK

Untuk informasi tentang pemeliharaan dan dukungan untuk versi utama SDK dan dependensi yang mendasarinya, lihat berikut ini di Panduan Referensi Alat AWS SDKs dan Alat berikut:

- AWS SDKs dan kebijakan pemeliharaan alat
- AWS SDKs dan matriks dukungan versi alat

Memulai dengan AWS SDK for Go

Ini AWS SDK for Go membutuhkan versi minimum Go 1.23. Anda dapat melihat versi Go Anda saat ini dengan menjalankan perintah berikut:

```
go version
```

Untuk informasi tentang menginstal atau memutakhirkan versi Go Anda, lihat Mengunduh dan menginstal di dokumentasi Go.

Dapatkan Akun Amazon

Sebelum Anda dapat menggunakan AWS SDK for Go v2, Anda harus memiliki akun Amazon. Lihat Membuat AWS akun untuk detailnya.

Instal AWS SDK for Go v2

AWS SDK for Go V2 menggunakan modul Go, yang merupakan fitur yang diperkenalkan di Go 1.11. Inisialisasi proyek lokal Anda dengan menjalankan perintah Go berikut.

```
go mod init example
```

Setelah menginisialisasi proyek modul Go Anda, Anda akan dapat mengambil SDK dan dependensi yang diperlukan menggunakan perintah. `go get` Dependensi ini akan direkam dalam `go.mod` file yang dibuat oleh perintah sebelumnya.

Perintah berikut menunjukkan cara mengambil set standar modul SDK yang akan digunakan dalam aplikasi Anda.

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

Ini akan mengambil modul SDK inti, dan modul konfigurasi yang digunakan untuk memuat konfigurasi bersama. AWS

Selanjutnya Anda dapat menginstal satu atau lebih klien API AWS layanan yang diperlukan oleh aplikasi Anda. Semua klien API berada di bawah hierarki `github.com/aws/aws-sdk-go-v2/service` impor. Satu set lengkap klien API yang saat ini didukung dapat ditemukan di sini.

Untuk menginstal klien layanan, jalankan perintah berikut untuk mengambil modul dan merekam ketergantungan dalam file Andago.mod. Dalam contoh ini kita mengambil klien API Amazon S3.

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

Dapatkan kunci AWS akses Anda

Access key terdiri dari access key ID dan secret access key, yang digunakan untuk menandatangani permintaan terprogram yang Anda buat ke AWS. Jika Anda tidak memiliki kunci akses, Anda dapat membuatnya dengan menggunakan AWS Management Console. Kami menyarankan Anda menggunakan kunci akses IAM alih-alih kunci akses akun AWS root. IAM memungkinkan Anda mengontrol akses ke AWS layanan dan sumber daya di akun Anda AWS dengan aman.

> **Note**
>
> Untuk membuat access keys Anda juga harus memiliki izin untuk melakukan tindakan IAM yang diperlukan. Untuk informasi selengkapnya, lihat Izin yang diperlukan untuk mengakses sumber daya IAM di Panduan Pengguna IAM.

Untuk mendapatkan ID kunci akses dan kunci akses rahasia Anda.

1. Buka konsol IAM.
2. Di menu navigasi, pilih Pengguna.
3. Pilih nama pengguna IAM Anda (bukan kotak centang).
4. Pilih tab Kredensial keamanan, lalu pilih Buat access key.
5. Untuk melihat access key baru, pilih Tampilkan. Kredensial Anda menyerupai berikut ini:
   - Access key ID: AKIAIOSFODNN7EXAMPLE
   - Secret access key: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
6. Untuk mengunduh pasangan kunci tersebut, pilih Unduh file .csv. Simpan kunci di lokasi yang aman.

> **Warning**
>
> Jaga kerahasiaan kunci untuk melindungi AWS akun Anda, dan jangan pernah membagikannya dengan siapa pun di luar organisasi Anda.

Topik terkait

- Apa itu IAM? di Panduan Pengguna IAM.
- AWS Kredensial Keamanan di Referensi Umum Amazon Web Services.

Memanggil Operasi

Setelah menginstal SDK, Anda mengimpor AWS paket ke aplikasi Go untuk menggunakan SDK, seperti yang ditunjukkan pada contoh berikut, yang mengimpor pustaka, AWS Config, dan Amazon S3. Setelah mengimpor paket SDK, AWS SDK Shared Configuration dimuat, klien dibuat, dan operasi API dipanggil.

```
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)
    }
}
```

Konfigurasikan SDK

Di AWS SDK for Go V2, Anda dapat mengonfigurasi pengaturan umum untuk klien layanan, seperti logger, level log, dan konfigurasi coba lagi. Sebagian besar pengaturan bersifat opsional. Namun, untuk setiap klien layanan, Anda harus menentukan AWS Wilayah dan kredensialnya. SDK menggunakan nilai ini untuk mengirim permintaan ke Wilayah yang benar dan menandatangani permintaan dengan kredensyal yang benar. Anda dapat menentukan nilai-nilai ini secara terprogram dalam kode atau melalui lingkungan eksekusi.

Memuat File Konfigurasi AWS Bersama

Ada sejumlah cara untuk menginisialisasi klien API layanan, tetapi berikut ini adalah pola paling umum yang direkomendasikan kepada pengguna.

Untuk mengonfigurasi SDK agar menggunakan file konfigurasi AWS bersama, gunakan kode berikut:

```
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())`akan membangun AWS.config menggunakan sumber konfigurasi bersama. AWS Ini termasuk mengonfigurasi penyedia kredensi, mengonfigurasi AWS Wilayah, dan memuat konfigurasi khusus layanan. Klien layanan dapat dibangun menggunakan bebanaws.Config, memberikan pola yang konsisten untuk membangun klien.

Untuk informasi selengkapnya tentang file konfigurasi AWS bersama, lihat Konfigurasi di Panduan Referensi Alat AWS SDKs dan Konfigurasi.

Menentukan Wilayah AWS

Ketika Anda menentukan Wilayah, Anda menentukan tempat untuk mengirim permintaan, seperti `us-west-2` atau`us-east-2`. Untuk daftar Wilayah untuk setiap layanan, lihat Titik akhir layanan dan kuota di. Referensi Umum Amazon Web

SDK tidak memiliki Region default. Untuk menentukan Wilayah:

- Atur variabel `AWS_REGION` lingkungan ke Region default.
- Setel wilayah secara eksplisit menggunakan konfigurasi. WithRegionsebagai argumen `config.LoadDefaultConfig` saat memuat konfigurasi.

TINJAUAN: Jika Anda menyetel Wilayah menggunakan semua teknik ini, SDK akan menggunakan Wilayah yang Anda tentukan secara eksplisit.

Konfigurasikan Wilayah dengan Variabel Lingkungan

Linux, macOS, atau Unix

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

Windows

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

Tentukan Wilayah Secara Terprogram

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

Menentukan Kredensial

AWS SDK for Go Memerlukan kredensyal (kunci akses dan kunci akses rahasia) untuk menandatangani permintaan. AWS Anda dapat menentukan kredensyal Anda di beberapa lokasi, tergantung pada kasus penggunaan khusus Anda. Untuk informasi tentang mendapatkan kredensional, lihat. Memulai dengan AWS SDK for Go

Saat Anda menginisialisasi `aws.Config` instance menggunakan`config.LoadDefaultConfig`, SDK menggunakan rantai kredensyal defaultnya untuk menemukan kredensyal. AWS Rantai kredensi default ini mencari kredensi dalam urutan sebagai berikut:

1. Variabel lingkungan.
   1. Kredensial Statis (`AWS_ACCESS_KEY_ID`,,`AWS_SECRET_ACCESS_KEY`) `AWS_SESSION_TOKEN`
   2. Token Identitas Web (`AWS_WEB_IDENTITY_TOKEN_FILE`)
2. File konfigurasi bersama.
   1. SDK default ke `credentials` file di bawah `.aws` folder yang ditempatkan di folder rumah di komputer Anda.
   2. SDK default ke `config` file di bawah `.aws` folder yang ditempatkan di folder rumah di komputer Anda.
3. Jika aplikasi Anda menggunakan definisi tugas Amazon ECS atau operasi RunTask API, peran IAM untuk tugas.
4. Jika aplikasi Anda berjalan di EC2 instans Amazon, peran IAM untuk Amazon EC2.

SDK mendeteksi dan menggunakan penyedia bawaan secara otomatis, tanpa memerlukan konfigurasi manual. Misalnya, jika Anda menggunakan peran IAM untuk EC2 instans Amazon, aplikasi Anda secara otomatis menggunakan kredensyal instans. Anda tidak perlu mengonfigurasi kredensi secara manual di aplikasi Anda.

Sebagai praktik terbaik, AWS merekomendasikan agar Anda menentukan kredensyal dalam urutan berikut:

1. Gunakan peran IAM untuk tugas jika aplikasi Anda menggunakan definisi tugas Amazon ECS atau operasi RunTask API.
2. Gunakan peran IAM untuk Amazon EC2 (jika aplikasi Anda berjalan di EC2 instans Amazon).

   Peran IAM menyediakan aplikasi pada kredensyal keamanan sementara instance untuk melakukan panggilan. AWS Peran IAM menyediakan cara mudah untuk mendistribusikan dan mengelola kredensyal di beberapa instans Amazon. EC2
3. Gunakan kredensi bersama atau file konfigurasi.

   Kredensyal dan file konfigurasi dibagikan di file lain dan file. AWS SDKs AWS CLI Sebagai praktik keamanan terbaik, sebaiknya gunakan file kredensyal untuk menyetel nilai sensitif seperti kunci akses IDs dan kunci rahasia. Berikut adalah persyaratan pemformatan untuk masing-masing file ini.

4. Gunakan variabel lingkungan.

   Menyetel variabel lingkungan berguna jika Anda melakukan pekerjaan pengembangan pada mesin selain EC2 instance Amazon.

Peran IAM untuk Tugas

Jika aplikasi Anda menggunakan definisi atau `RunTask` operasi tugas Amazon ECS, gunakan Peran IAM untuk Tugas untuk menentukan peran IAM yang dapat digunakan oleh kontainer dalam tugas.

Peran IAM untuk Instans Amazon EC2

Jika Anda menjalankan aplikasi di EC2 instans Amazon, gunakan peran IAM instans untuk mendapatkan kredensi keamanan sementara untuk melakukan panggilan. AWS

Jika Anda telah mengonfigurasi instans Anda untuk menggunakan peran IAM, SDK akan menggunakan kredensyal ini untuk aplikasi Anda secara otomatis. Anda tidak perlu menentukan kredensi ini secara manual.

Kredensi dan Konfigurasi Bersama

Kredensyal bersama dan file konfigurasi dapat digunakan untuk berbagi konfigurasi umum antara AWS SDKs dan alat lainnya. Jika Anda menggunakan kredensi yang berbeda untuk alat atau aplikasi yang berbeda, Anda dapat menggunakan profil untuk mengonfigurasi beberapa kunci akses dalam file konfigurasi yang sama.

Anda dapat menyediakan beberapa lokasi file kredensyal atau konfigurasi menggunakan`config.LoadOptions`, secara default SDK memuat file yang disimpan di lokasi default yang disebutkan dalam file. Menentukan Kredensial

```
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"},
    )
)
```

Saat bekerja dengan kredensyal bersama dan file konfigurasi, jika profil duplikat ditentukan, profil tersebut digabungkan untuk menyelesaikan profil. Dalam kasus konflik penggabungan,

1. Jika profil duplikat ditentukan dalam credentials/config file yang sama, properti profil yang ditentukan dalam profil terakhir diutamakan.
2. Jika profil duplikat ditentukan di beberapa file kredensional atau di beberapa file konfigurasi, properti profil diselesaikan sesuai urutan input file ke file. `config.LoadOptions` Properti profil dalam file terakhir diutamakan.
3. Jika profil ada di file kredensyal dan file konfigurasi, properti file kredensyal diutamakan.

Jika perlu, Anda dapat `LogConfigurationWarnings` mengaktifkan `config.LoadOptions` dan mencatat langkah-langkah resolusi profil.

Membuat File Kredensial

Jika Anda tidak memiliki file kredensyal bersama (`.aws/credentials`), Anda dapat menggunakan editor teks apa pun untuk membuatnya di direktori home Anda. Tambahkan konten berikut ke file kredensyal Anda, ganti *<YOUR_ACCESS_KEY_ID>* dan *<YOUR_SECRET_ACCESS_KEY>* dengan kredensyal Anda.

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

`[default]`Judul mendefinisikan kredensyal untuk profil default, yang akan digunakan SDK kecuali Anda mengonfigurasinya untuk menggunakan profil lain.

Anda juga dapat menggunakan kredensyal keamanan sementara dengan menambahkan token sesi ke profil Anda, seperti yang ditunjukkan pada contoh berikut:

```
[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>
```

Nama bagian untuk profil non-default dalam file kredensyal tidak boleh dimulai dengan kata. `profile` Anda dapat membaca lebih lanjut di AWS SDKs dan Panduan Referensi Alat.

Membuat File Config

Jika Anda tidak memiliki file kredensyal bersama (`.aws/config`), Anda dapat menggunakan editor teks apa pun untuk membuatnya di direktori home Anda. Tambahkan konten berikut ke file konfigurasi Anda, ganti *<REGION>* dengan wilayah yang diinginkan.

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

`[default]`Judul mendefinisikan konfigurasi untuk profil default, yang akan digunakan SDK kecuali Anda mengonfigurasinya untuk menggunakan profil lain.

Anda dapat menggunakan profil bernama seperti yang ditunjukkan pada contoh berikut:

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

Nama bagian untuk profil non-default dalam file konfigurasi harus selalu dimulai dengan kata`profile`, diikuti dengan nama profil yang dimaksud. Anda dapat membaca lebih lanjut di Panduan Referensi AWS SDKs dan Alat.

Menentukan Profil

Anda dapat menyertakan beberapa kunci akses dalam file konfigurasi yang sama dengan mengaitkan setiap set kunci akses dengan profil. Misalnya, dalam file kredensial Anda, Anda dapat mendeklarasikan beberapa profil, sebagai berikut.

```
[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>
```

Secara default, SDK memeriksa variabel `AWS_PROFILE` lingkungan untuk menentukan profil mana yang akan digunakan. Jika tidak ada `AWS_PROFILE` variabel yang disetel, SDK menggunakan `default` profil.

Terkadang, Anda mungkin ingin menggunakan profil yang berbeda dengan aplikasi Anda. Misalnya, Anda ingin menggunakan `test-account` kredensialnya dengan aplikasi Anda`myapp`. Anda dapat menggunakan profil ini dengan menggunakan perintah berikut:

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

Anda juga dapat menggunakan instruksikan SDK untuk memilih profil dengan menelepon `os.Setenv("AWS_PROFILE", "test-account")` sebelum meneleponconfig.LoadDefaultConfig, atau dengan meneruskan profil eksplisit sebagai argumen seperti yang ditunjukkan pada contoh berikut:

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

> **Note**
>
> Jika Anda menentukan kredensyal dalam variabel lingkungan, SDK akan selalu menggunakan kredensyal tersebut, apa pun profil yang Anda tentukan.

Variabel lingkungan

Secara default, SDK mendeteksi AWS kredensyal yang disetel di lingkungan Anda dan menggunakannya untuk menandatangani permintaan. AWS Dengan begitu Anda tidak perlu mengelola kredensi di aplikasi Anda.

SDK mencari kredensional dalam variabel lingkungan berikut:

- `AWS_ACCESS_KEY_ID`
- `AWS_SECRET_ACCESS_KEY`
- `AWS_SESSION_TOKEN` (opsional)

Contoh berikut menunjukkan bagaimana Anda mengkonfigurasi variabel lingkungan.

Linux, OS X, atau 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
```

Tentukan Kredensial Secara Terprogram

`config.LoadDefaultConfig`memungkinkan Anda memberikan aws eksplisit. CredentialProvidersaat memuat sumber konfigurasi bersama. Untuk meneruskan penyedia kredensyal eksplisit saat memuat konfigurasi bersama gunakan konfigurasi. WithCredentialsProvider. Misalnya, jika `customProvider` mereferensikan instance `aws.CredentialProvider` implementasi, itu dapat diteruskan selama pemuatan konfigurasi seperti:

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

Jika Anda secara eksplisit memberikan kredensyal, seperti dalam contoh ini, SDK hanya menggunakan kredensyal tersebut.

> **Note**
>
> Semua penyedia kredensi yang diteruskan atau dikembalikan oleh `LoadDefaultConfig` dibungkus secara CredentialsCacheotomatis. Ini memungkinkan rotasi caching dan kredenal yang aman secara konkurensi. Jika Anda secara eksplisit mengonfigurasi penyedia secara

`aws.Config` langsung, Anda juga harus secara eksplisit membungkus penyedia dengan tipe ini menggunakan. NewCredentialsCache

Kredensial Statis

Anda dapat membuat kode keras kredensional dalam aplikasi Anda dengan menggunakan kredensialnya. NewStaticCredentialsProviderpenyedia kredensi untuk secara eksplisit mengatur kunci akses yang akan digunakan. Contoh:

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

Warning

Jangan menanamkan kredensi di dalam aplikasi. Gunakan metode ini hanya untuk tujuan pengujian.

Kredensial Masuk Tunggal

SDK menyediakan penyedia kredensyal untuk mengambil kredensyal sementara AWS menggunakan. AWS IAM Identity Center Dengan menggunakan AWS CLI, Anda mengautentikasi dengan portal AWS akses dan mengotorisasi akses ke kredensyal sementara AWS . Anda kemudian mengonfigurasi aplikasi Anda untuk memuat profil single sign-on (SSO), dan SDK menggunakan kredenal SSO Anda untuk mengambil kredenal sementara AWS yang akan diperpanjang secara otomatis jika kedaluwarsa. Jika kredensyal SSO Anda kedaluwarsa, Anda harus memperbaruinya secara eksplisit dengan masuk ke akun Pusat Identitas IAM Anda lagi menggunakan. AWS CLI

Misalnya, Anda dapat membuat profil,`dev-profile`, mengautentikasi dan mengotorisasi profil tersebut menggunakan AWS CLI, dan mengkonfigurasi aplikasi Anda seperti yang ditunjukkan di bawah ini.

1. Pertama buat `profile` dan `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. Login menggunakan AWS CLI untuk mengautentikasi dan mengotorisasi profil SSO.

```
$ 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
Successully logged into Start URL: https://company-sso-portal.awsapps.com/start
```

3. Selanjutnya konfigurasikan aplikasi Anda untuk menggunakan profil 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
}
```

Untuk informasi selengkapnya tentang mengonfigurasi profil SSO dan mengautentikasi menggunakan AWS CLI lihat Mengonfigurasi yang AWS CLI akan digunakan AWS IAM Identity

Center dalam Panduan Pengguna. AWS CLI Untuk informasi selengkapnya tentang pembuatan penyedia kredensi SSO secara terprogram, lihat dokumentasi referensi API ssocreds.

Kredensi login

Anda dapat menggunakan kredenal masuk Konsol AWS Manajemen yang sudah ada untuk akses terprogram ke layanan. AWS Setelah alur otentikasi berbasis browser, buat AWS kredenal sementara yang berfungsi di seluruh alat pengembangan lokal seperti CLI, dan. AWS AWS Tools for PowerShell AWS SDKs Fitur ini menyederhanakan proses konfigurasi dan pengelolaan kredensional AWS CLI, terutama jika Anda lebih memilih otentikasi interaktif daripada mengelola kunci akses jangka panjang.

1. Memulai alur login menggunakan AWS CLI, dan ikuti petunjuk browser. Dalam contoh ini, kami menyimpan sesi login ke profil barudev-profile, tetapi ini opsional.

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

2. (Opsional) Periksa file konfigurasi AWS bersama Anda untuk memverifikasi bahwa sesi telah dibuat.

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

3. Selanjutnya konfigurasikan aplikasi Anda untuk menggunakan profil login.

```
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
}
```

Untuk informasi selengkapnya tentang mengonfigurasi profil login dan mengautentikasi menggunakan AWS CLI, lihat Login untuk pengembangan AWS lokal menggunakan kredenal konsol.

Penyedia Kredensial Lainnya

SDK menyediakan metode lain untuk mengambil kredensyal dalam modul kredensyal. Misalnya, Anda dapat mengambil kredensyal keamanan sementara dari AWS Security Token Service atau kredensyal dari penyimpanan terenkripsi.

Penyedia Kredensi yang Tersedia:

- ec2rolecreds - Ambil Kredensyal dari Peran Instans Amazon melalui Amazon IMDS. EC2 EC2
- endpointcreds - Ambil Kredensial dari titik akhir HTTP arbitrer.
- processcreds — Ambil Credentials dari proses eksternal yang akan dipanggil oleh shell lingkungan host.
- stscreds - Ambil Kredensi dari AWS STS

Konfigurasikan Otentikasi

AWS SDK for Go Ini menyediakan kemampuan untuk mengkonfigurasi layanan perilaku otentikasi. Dalam kebanyakan kasus, konfigurasi default sudah cukup, tetapi mengonfigurasi otentikasi kustom memungkinkan perilaku tambahan seperti bekerja dengan fitur layanan pra-rilis.

Definisi

Bagian ini memberikan deskripsi tingkat tinggi dari komponen otentikasi di. AWS SDK for Go

AuthScheme

An AuthSchemeadalah antarmuka yang mendefinisikan alur kerja di mana SDK mengambil identitas pemanggil dan melampirkannya ke permintaan operasi.

Skema autentikasi menggunakan komponen-komponen berikut, dijelaskan secara rinci lebih lanjut di bawah ini:

- ID unik yang mengidentifikasi skema
- Penyelesai identitas, yang mengembalikan identitas pemanggil yang digunakan dalam proses penandatanganan (misalnya kredensyal Anda) AWS

- Penandatangan, yang melakukan injeksi identitas pemanggil yang sebenarnya ke dalam permintaan transport operasi (misalnya header `Authorization` HTTP)

Setiap opsi klien layanan menyertakan `AuthSchemes` bidang, yang secara default diisi dengan daftar skema autentikasi yang didukung oleh layanan tersebut.

AuthSchemeResolver

Setiap opsi klien layanan mencakup `AuthSchemeResolver` bidang. Antarmuka ini, didefinisikan per layanan, adalah API yang dipanggil oleh SDK untuk menentukan opsi otentikasi yang mungkin untuk setiap operasi.

> **Important**
>
> Penyelesai skema autentikasi TIDAK menentukan skema autentikasi apa yang digunakan. Ini mengembalikan daftar skema yang dapat digunakan (“opsi”), skema akhir dipilih melalui algoritma tetap yang dijelaskan di sini.

Opsi

Dikembalikan dari panggilan ke`ResolverAuthSchemes`, Option mewakili opsi otentikasi yang mungkin.

Opsi terdiri dari tiga set informasi:

- ID yang mewakili skema yang mungkin
- Satu set properti buram yang akan diberikan kepada penyelesai identitas skema
- Satu set properti buram yang akan diberikan kepada penandatangan skema

Catatan tentang properti

Untuk 99% kasus penggunaan, penelepon tidak perlu peduli dengan properti buram untuk resolusi dan penandatanganan identitas. SDK akan mengeluarkan properti yang diperlukan untuk setiap skema dan meneruskannya ke antarmuka yang diketik dengan kuat yang diekspos di SDK. Misalnya, penyelesai autentikasi default untuk layanan menyandikan opsi SigV4 agar memiliki properti penandatangan untuk nama dan wilayah penandatanganan, yang nilainya diteruskan ke v4 yang dikonfigurasi klien. HTTPSignerimplementasi ketika SigV4 dipilih.

Identitas

Identitas adalah representasi abstrak dari siapa pemanggil SDK.

Jenis identitas yang paling umum digunakan dalam SDK adalah seperangkat. `aws.Credentials` Untuk sebagian besar kasus penggunaan, penelepon tidak perlu menyibukkan diri `Identity` sebagai abstraksi dan dapat bekerja dengan tipe konkret secara langsung.

> **Note**
>
> Untuk mempertahankan kompatibilitas mundur dan mencegah kebingungan API, tipe identitas AWS khusus SDK `aws.Credentials` tidak secara langsung memenuhi antarmuka. `Identity` Pemetaan ini ditangani secara internal.

IdentityResolver

IdentityResolveradalah antarmuka yang melaluinya sebuah `Identity` diambil.

Versi konkret `IdentityResolver` ada di SDK dalam bentuk yang diketik dengan kuat (misalnya aws. CredentialsProvider), SDK menangani pemetaan ini secara internal.

Penelepon hanya perlu mengimplementasikan `IdentityResolver` antarmuka secara langsung saat mendefinisikan skema autentikasi eksternal.

Signer

Penandatangan adalah antarmuka di mana permintaan dilengkapi dengan pemanggil yang diambil. `Identity`

Versi konkret yang `Signer` ada di SDK dalam bentuk yang diketik dengan kuat (misalnya v4. HTTPSigner), SDK menangani pemetaan ini secara internal.

Penelepon hanya perlu mengimplementasikan `Signer` antarmuka secara langsung saat mendefinisikan skema autentikasi eksternal.

AuthResolverParameters

Setiap layanan mengambil satu set input tertentu yang diteruskan ke fungsi resolusinya, yang didefinisikan dalam setiap paket layanan sebagai`AuthResolverParameters`.

Parameter resolver dasar adalah sebagai berikut:

name	jenis	deskripsi
`Operation`	`string`	Nama operasi yang dipanggil.
`Region`	`string`	AWS Wilayah klien. Hanya hadir untuk layanan yang menggunakan SiGv4 [A].

Jika Anda menerapkan resolver Anda sendiri, Anda seharusnya tidak perlu membuat instance Anda sendiri dari parameternya. SDK akan mencari nilai per permintaan ini dan meneruskannya ke implementasi Anda.

Alur kerja resolusi skema autentikasi

Saat Anda memanggil operasi AWS layanan melalui SDK, urutan tindakan berikut akan terjadi setelah permintaan diserialisasi:

1. SDK memanggil `AuthSchemeResolver.ResolveAuthSchemes()` API klien, mencari parameter input seperlunya, untuk mendapatkan daftar Opsi yang mungkin untuk operasi.
2. SDK mengulangi daftar itu dan memilih skema pertama yang memenuhi kondisi berikut.
   - Skema dengan ID yang cocok hadir dalam `AuthSchemes` daftar klien sendiri
   - Penyelesai identitas skema ada (tidak-`nil`) pada Opsi klien (diperiksa melalui `GetIdentityResolver` metode skema, pemetaan ke jenis penyelesai identitas konkret yang dijelaskan di atas ditangani secara internal) (1)
3. Dengan asumsi skema yang layak dipilih, SDK memanggil `GetIdentityResolver()` API-nya untuk mengambil identitas pemanggil. Misalnya, skema autentikasi SiGv4 bawaan akan dipetakan ke penyedia klien secara internal. `Credentials`
4. SDK memanggil penyelesai identitas `GetIdentity()` (misalnya `aws.CredentialProvider.Retrieve()` untuk SigV4).
5. SDK memanggil penyelesai titik akhir `ResolveEndpoint()` untuk menemukan titik akhir permintaan. Titik akhir dapat mencakup metadata tambahan yang memengaruhi proses penandatanganan (misalnya nama penandatanganan unik untuk Objek Lambda S3).
6. SDK memanggil `Signer()` API skema autentikasi untuk mengambil penandatangannya, dan menggunakan `SignRequest()` API-nya untuk menandatangani permintaan dengan identitas pemanggil yang diambil sebelumnya.

(1) Jika SDK menemukan opsi anonim (ID`smithy.api#noAuth`) dalam daftar, itu dipilih secara otomatis, karena tidak ada penyelesai identitas yang sesuai.

S yang didukung secara asli **`AuthScheme`**

Skema autentikasi berikut didukung secara native oleh. AWS SDK for Go

Nama	ID Skema	Penyelesai identitas	Signer	Catatan
SiGv4	`aws.auth#sigv4`	aws. CredentialsProvider	v4. HTTPSigner	Default saat ini untuk sebagian besar operasi AWS layanan.
Sigv4a	`aws.auth#sigv4a`	aws. CredentialsProvider	T/A	Penggunaan Sigv4a terbatas saat ini, implementasi penandatanganan bersifat internal. Lihat pengumuman ini untuk modul keikutsertaan baru aws-http-authyang memperlihatkan tujuan umum APIs untuk menandatanganani permintaan HTTP.
SIGV4Express	`com.amazonaws.s3#s`	s3. ExpressCredentials Provider	v4. HTTPSigner	Digunakan untuk Express One Zone.

Nama	ID Skema	Penyelesai identitas	Signer	Catatan
	`igv4express`			
Pembawa HTTP	`smithy.api#httpBearerAuth`	pembawa smithybearer.TokenProvider	Smithybearer.Signer	Digunakan oleh codecatalyst.
Anonim	`smithy.api#noAuth`	tidak berlaku	T/A	Tidak ada otentikasi - tidak diperlukan identitas, dan permintaan tidak ditandatangani atau diautentikasi.

Konfigurasi identitas

Dalam AWS SDK for Go, komponen identitas skema autentikasi dikonfigurasi dalam klien SDK. `Options` SDK akan secara otomatis mengambil dan menggunakan nilai untuk komponen ini untuk skema yang dipilihnya saat operasi dipanggil.

> **Note**
>
> Untuk alasan kompatibilitas mundur, SDK secara implisit mengizinkan penggunaan skema autentikasi anonim jika tidak ada penyelesai identitas yang dikonfigurasi. Ini dapat dicapai secara manual dengan menyetel semua penyelesai identitas pada klien `Options` ke `nil` (penyelesai identitas sigv4 juga dapat diatur ke). `aws.AnonymousCredentials{}`

Konfigurasi penandatangan

Di AWS SDK for Go, komponen penandatangan skema autentikasi dikonfigurasi di klien SDK. `Options` SDK akan secara otomatis mengambil dan menggunakan nilai untuk komponen ini untuk skema yang dipilihnya saat operasi dipanggil. Tidak diperlukan konfigurasi tambahan.

Skema autentikasi khusus

Untuk menentukan skema autentikasi khusus dan mengonfigurasinya untuk digunakan, pemanggil harus melakukan hal berikut:

1. Mendefinisikan AuthSchemeimplementasi
2. Daftarkan skema pada daftar klien SDK `AuthSchemes`
3. Instrumen klien SDK `AuthSchemeResolver` untuk mengembalikan autentikasi `Option` dengan ID skema jika berlaku

> **Warning**
>
> Layanan berikut memiliki perilaku otentikasi yang unik atau disesuaikan. Kami menyarankan Anda mendelegasikan ke implementasi default dan membungkusnya jika Anda memerlukan perilaku otentikasi khusus:

Layanan	Catatan
S3	Penggunaan bersyarat Sigv4a dan Sigv4Express tergantung pada input operasi.
EventBridge	Penggunaan bersyarat Sigv4a tergantung pada input operasi.
Cognito	Operasi tertentu hanya bersifat anonim.
SSO	Operasi tertentu hanya bersifat anonim.
STS	Operasi tertentu hanya bersifat anonim.

Konfigurasikan Titik Akhir Klien

> **Warning**
>
> Resolusi titik akhir adalah topik SDK lanjutan. Dengan mengubah pengaturan ini, Anda berisiko melanggar kode Anda. Pengaturan default harus berlaku untuk sebagian besar pengguna di lingkungan produksi.

AWS SDK for Go Ini menyediakan kemampuan untuk mengonfigurasi titik akhir kustom yang akan digunakan untuk layanan. Dalam kebanyakan kasus, konfigurasi default sudah cukup. Mengonfigurasi titik akhir kustom memungkinkan perilaku tambahan, seperti bekerja dengan versi layanan pra-rilis.

Kustomisasi

Ada dua “versi” konfigurasi resolusi titik akhir dalam SDK.

- v2, dirilis pada Q3 tahun 2023, dikonfigurasi melalui:
  - `EndpointResolverV2`
  - `BaseEndpoint`
- v1, dirilis bersama SDK, dikonfigurasi melalui:
  - `EndpointResolver`

Kami menyarankan pengguna resolusi titik akhir v1 bermigrasi ke v2 untuk mendapatkan akses ke fitur layanan terkait titik akhir yang lebih baru.

V2: **`EndpointResolverV2`** + **`BaseEndpoint`**

Dalam resolusi v2, `EndpointResolverV2` adalah mekanisme definitif di mana resolusi titik akhir terjadi. `ResolveEndpoint`Metode resolver dipanggil sebagai bagian dari alur kerja untuk setiap permintaan yang Anda buat di SDK. Nama host yang `Endpoint` dikembalikan oleh resolver digunakan apa adanya saat membuat permintaan (serializer operasi masih dapat ditambahkan ke jalur HTTP).

Resolution v2 menyertakan konfigurasi tingkat klien tambahan`BaseEndpoint`, yang digunakan untuk menentukan nama host “dasar” untuk instance layanan Anda. Nilai yang ditetapkan di sini tidak definitif-- itu akhirnya diteruskan sebagai parameter ke klien `EndpointResolverV2` ketika resolusi

akhir terjadi (baca terus untuk informasi lebih lanjut tentang `EndpointResolverV2` parameter). Implementasi resolver kemudian memiliki kesempatan untuk memeriksa dan berpotensi memodifikasi nilai tersebut untuk menentukan titik akhir.

Misalnya, jika Anda melakukan `GetObject` permintaan S3 terhadap bucket tertentu dengan klien yang telah Anda tentukan`BaseEndpoint`, resolver default akan menyuntikkan bucket ke nama host jika kompatibel dengan virtual-host (dengan asumsi Anda belum menonaktifkan virtual-hosting di konfigurasi klien).

Dalam prakteknya, kemungkinan besar `BaseEndpoint` akan digunakan untuk mengarahkan klien Anda pada contoh pengembangan atau pratinjau layanan.

Parameter **`EndpointResolverV2`**

Setiap layanan mengambil satu set input tertentu yang diteruskan ke fungsi resolusinya, yang didefinisikan dalam setiap paket layanan sebagai`EndpointParameters`.

Setiap layanan mencakup parameter dasar berikut, yang digunakan untuk memfasilitasi resolusi titik akhir umum dalam AWS:

name	jenis	deskripsi
`Region`	`string`	AWS Wilayah klien
`Endpoint`	`string`	Nilai yang ditetapkan untuk `BaseEndpoint` konfigurasi klien
`UseFips`	`bool`	Apakah titik akhir FIPS diaktifkan dalam konfigurasi klien
`UseDualStack`	`bool`	Apakah titik akhir tumpukan ganda diaktifkan dalam konfigurasi klien

Layanan dapat menentukan parameter tambahan yang diperlukan untuk resolusi. Misalnya, S3 `EndpointParameters` menyertakan nama bucket, serta beberapa pengaturan fitur khusus S3 seperti apakah pengalamatan host virtual diaktifkan.

Jika Anda menerapkan sendiri`EndpointResolverV2`, Anda seharusnya tidak perlu membangun contoh Anda sendiri. `EndpointParameters` SDK akan mencari nilai per permintaan dan meneruskannya ke implementasi Anda.

Catatan tentang Amazon S3

Amazon S3 adalah layanan yang kompleks dengan banyak fitur-fiturnya yang dimodelkan melalui penyesuaian titik akhir yang kompleks, seperti hosting virtual bucket, S3 MRAP, dan banyak lagi.

Karena itu, kami menyarankan Anda untuk tidak mengganti `EndpointResolverV2` implementasi di klien S3 Anda. Jika Anda perlu memperluas perilaku resolusinya, mungkin dengan mengirimkan permintaan ke tumpukan pengembangan lokal dengan pertimbangan titik akhir tambahan, kami sarankan untuk membungkus implementasi default sedemikian rupa sehingga mendelegasikan kembali ke default sebagai fallback (ditunjukkan dalam contoh di bawah).

Contoh

dengan **`BaseEndpoint`**

Cuplikan kode berikut menunjukkan cara mengarahkan klien S3 Anda ke instance lokal layanan, yang dalam contoh ini di-host pada perangkat loopback di port 8080.

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

dengan **`EndpointResolverV2`**

Cuplikan kode berikut menunjukkan cara menyuntikkan perilaku khusus ke resolusi titik akhir S3 menggunakan. `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{
            // ...
        }
    })
}
```

Dengan keduanya

Program sampel berikut menunjukkan interaksi antara `BaseEndpoint` dan`EndpointResolverV2`. Ini adalah kasus penggunaan lanjutan:

```
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)
}
```

Saat dijalankan, program di atas menghasilkan yang berikut:

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

V1: `EndpointResolver`

> **Warning**
>
> Resolusi titik akhir v1 dipertahankan untuk kompatibilitas mundur dan diisolasi dari perilaku modern dalam resolusi titik akhir v2. Ini hanya akan digunakan jika `EndpointResolver` bidang diatur oleh penelepon.

> Penggunaan v1 kemungkinan besar akan mencegah Anda mengakses fitur layanan terkait titik akhir yang diperkenalkan dengan atau setelah rilis resolusi v2. Lihat “Migrasi” untuk petunjuk tentang cara meningkatkan.

A EndpointResolverdapat dikonfigurasi untuk memberikan logika resolusi titik akhir kustom untuk klien layanan. Anda dapat menggunakan resolver endpoint khusus untuk mengganti logika resolusi titik akhir layanan untuk semua titik akhir, atau titik akhir regional tertentu saja. Penyelesai titik akhir khusus dapat memicu logika resolusi titik akhir layanan untuk mundur jika resolver khusus tidak ingin menyelesaikan titik akhir yang diminta. EndpointResolverWithOptionsFuncdapat digunakan untuk dengan mudah membungkus fungsi untuk memenuhi `EndpointResolverWithOptions` antarmuka.

A `EndpointResolver` dapat dengan mudah dikonfigurasi dengan meneruskan resolver yang dibungkus dengan WithEndpointResolverWithOptionsto LoadDefaultConfig, memungkinkan kemampuan untuk mengganti titik akhir saat memuat kredensi, serta mengonfigurasi hasil dengan resolver titik akhir kustom Anda. `aws.Config`

Penyelesai titik akhir diberikan layanan dan wilayah sebagai string, memungkinkan resolver untuk mendorong perilakunya secara dinamis. Setiap paket klien layanan memiliki `ServiceID` konstanta yang diekspor yang dapat digunakan untuk menentukan klien layanan mana yang memanggil resolver endpoint Anda.

Resolver endpoint dapat menggunakan nilai kesalahan EndpointNotFoundErrorsentinel untuk memicu resolusi fallback ke logika resolusi default klien layanan. Ini memungkinkan Anda untuk secara selektif mengganti satu atau lebih titik akhir dengan mulus tanpa harus menangani logika fallback.

Jika implementasi endpoint resolver Anda mengembalikan kesalahan selain`EndpointNotFoundError`, resolusi endpoint akan berhenti dan operasi layanan mengembalikan error ke aplikasi Anda.

Contoh

Dengan fallback

Cuplikan kode berikut menunjukkan bagaimana titik akhir layanan tunggal dapat diganti untuk DynamoDB dengan perilaku fallback untuk titik akhir lainnya:

```
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))
```

Tanpa fallback

Cuplikan kode berikut menunjukkan bagaimana titik akhir layanan tunggal dapat diganti untuk DynamoDB tanpa perilaku fallback untuk titik akhir lainnya:

```
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))
```

Titik akhir yang tidak dapat diubah

> **Warning**
>
> Menyetel titik akhir sebagai tidak dapat diubah dapat mencegah beberapa fitur klien layanan berfungsi dengan benar, dan dapat mengakibatkan perilaku yang tidak terdefinisi. Perhatian harus diambil saat mendefinisikan titik akhir sebagai tidak dapat diubah.

Beberapa klien layanan, seperti Amazon S3, dapat memodifikasi titik akhir yang dikembalikan oleh resolver untuk operasi layanan tertentu. Misalnya, Amazon S3 akan secara otomatis menangani Virtual Bucket Addressing dengan mengubah titik akhir yang diselesaikan. Anda dapat mencegah SDK mengubah titik akhir kustom Anda dengan menyetelnya. HostnameImmutable`true` Contoh:

```
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))
```

Migrasi

Saat bermigrasi dari v1 ke v2 resolusi titik akhir, prinsip umum berikut berlaku:

- Mengembalikan Endpoint dengan HostnameImmutableset to kira-kira `false` setara dengan pengaturan `BaseEndpoint` ke URL yang dikembalikan semula dari v1 dan meninggalkannya `EndpointResolverV2` sebagai default.
- Mengembalikan Endpoint dengan HostnameImmutable set to kira-kira `true` setara dengan mengimplementasikan `EndpointResolverV2` yang mengembalikan URL yang dikembalikan semula dari v1.

- Pengecualian utama adalah untuk operasi dengan awalan titik akhir yang dimodelkan. Catatan tentang ini diberikan lebih jauh ke bawah.

Contoh untuk kasus-kasus ini disediakan di bawah ini.

> ⚠ Warning
>
> Titik akhir yang tidak dapat diubah V1 dan resolusi V2 tidak setara dalam perilaku. Misalnya, penandatanganan penggantian untuk fitur khusus seperti S3 Object Lambda masih akan disetel untuk titik akhir yang tidak dapat diubah yang dikembalikan melalui kode v1, tetapi hal yang sama tidak akan dilakukan untuk v2.

Catatan tentang awalan host

Beberapa operasi dimodelkan dengan awalan host untuk ditambahkan ke titik akhir yang diselesaikan. Perilaku ini harus bekerja bersama-sama dengan output ResolveEndpoint V2 dan oleh karena itu awalan host akan tetap diterapkan pada hasil itu.

Anda dapat menonaktifkan awalan host endpoint secara manual dengan menerapkan middleware, lihat bagian contoh.

Contoh

Titik akhir yang bisa berubah

Contoh kode berikut menunjukkan cara memigrasikan resolver endpoint v1 dasar yang mengembalikan titik akhir yang dapat dimodifikasi:

```
// 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/")
```

```
})
```

Titik akhir yang tidak dapat diubah

```
// 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{}
})
```

Nonaktifkan awalan 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)
    }
}
```

Sesuaikan Klien HTTP

AWS SDK for Go Menggunakan klien HTTP default dengan nilai konfigurasi default. Meskipun Anda dapat mengubah beberapa nilai konfigurasi ini, klien HTTP default dan transport tidak cukup dikonfigurasi untuk pelanggan yang menggunakan AWS SDK for Go dalam lingkungan dengan throughput tinggi dan persyaratan latensi rendah. Untuk informasi lebih lanjut, silakan lihat rekomendasi konfigurasi Pertanyaan yang Sering Diajukan as bervariasi berdasarkan beban kerja tertentu. Bagian ini menjelaskan cara mengkonfigurasi klien HTTP kustom, dan menggunakan klien tersebut untuk membuat AWS SDK for Go panggilan.

Untuk membantu Anda dalam membuat klien HTTP kustom, bagian ini menjelaskan cara NewBuildableClientuntuk mengkonfigurasi pengaturan kustom, dan menggunakan klien itu dengan klien AWS SDK for Go layanan.

Mari kita definisikan apa yang ingin kita sesuaikan.

Mengganti Selama Pemuatan Konfigurasi

Klien HTTP kustom dapat diberikan saat memanggil LoadDefaultConfigdengan membungkus klien menggunakan With HTTPClient dan meneruskan nilai yang dihasilkan ke`LoadDefaultConfig`. Misalnya, untuk lulus `customClient` sebagai klien kami:

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

Waktu habis

`BuildableHTTPClient`Dapat dikonfigurasi dengan batas waktu tunggu permintaan. Batas waktu ini mencakup waktu untuk terhubung, memproses pengalihan apa pun, dan membaca badan respons lengkap. Misalnya, untuk memodifikasi batas waktu klien:

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

// ...

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

Dialer

`BuildableHTTPClient`Ini menyediakan mekanisme pembangun untuk membangun klien dengan opsi Dialer yang dimodifikasi. Contoh berikut menunjukkan cara mengkonfigurasi `Dialer` pengaturan klien.

```
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
})
```

Pengaturan

Dialer. KeepAlive

Pengaturan ini mewakili periode keep-alive untuk koneksi jaringan aktif.

Setel ke nilai negatif untuk menonaktifkan keep-alives.

Setel ke 0 untuk mengaktifkan keep-alives jika didukung oleh protokol dan sistem operasi.

Protokol jaringan atau sistem operasi yang tidak mendukung keep-alives mengabaikan bidang ini. Secara default, TCP memungkinkan tetap hidup.

Lihat https://golang. org/pkg/net/#Dialer. KeepAlive

Tetapkan `KeepAlive` sebagai time.Duration.

Dialer.Timeout

Pengaturan ini mewakili jumlah maksimum waktu dial menunggu koneksi dibuat.

Defaultnya adalah 30 detik.

Lihat https://golang. org/pkg/net/#Dialer .Timeout

Tetapkan `Timeout` sebagai time.Duration.

Transportasi

`BuildableHTTPClient`Ini menyediakan mekanisme pembangun untuk membangun klien dengan opsi Transportasi yang dimodifikasi.

Mengkonfigurasi Proxy

Jika Anda tidak dapat langsung terhubung ke internet, Anda dapat menggunakan variabel lingkungan yang didukung GO (HTTP_PR0XY/HTTPS_PR0XY) atau membuat klien HTTP khusus untuk mengonfigurasi proxy Anda. Contoh berikut mengonfigurasi klien untuk digunakan PR0XY_URL sebagai titik akhir 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("PR0XY_URL")
    if err != nil {
        log.Fatal(err)
    }
    tr.Proxy = http.ProxyURL(proxyURL)
```

```
})
```

Pengaturan Lainnya

Di bawah ini adalah beberapa `Transport` pengaturan lain yang dapat dimodifikasi untuk menyetel klien HTTP. Pengaturan tambahan apa pun yang tidak dijelaskan di sini dapat ditemukan di dokumentasi tipe Transport. Pengaturan ini dapat diterapkan seperti yang ditunjukkan pada contoh berikut:

```
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
})
```

Transportasi. ExpectContinueTimeout

Jika permintaan memiliki header “Expect: 100-continue”, pengaturan ini mewakili jumlah waktu maksimum untuk menunggu header respons pertama server setelah sepenuhnya menulis header permintaan, Kali ini tidak termasuk waktu untuk mengirim header permintaan. Klien HTTP mengirimkan payloadnya setelah batas waktu ini habis.

Default 1 detik.

Setel ke 0 tanpa batas waktu dan kirim payload permintaan tanpa menunggu. Salah satu kasus penggunaan adalah ketika Anda mengalami masalah dengan proxy atau layanan pihak ketiga yang mengambil sesi yang mirip dengan penggunaan Amazon S3 dalam fungsi yang ditampilkan nanti.

Lihat https://golang. org/pkg/net/http/#Transport. ExpectContinueTimeout

Tetapkan `ExpectContinue` sebagai time.Duration.

Transportasi. IdleConnTimeout

Pengaturan ini mewakili jumlah maksimum waktu untuk menjaga koneksi jaringan idle tetap hidup antara permintaan HTTP.

Setel ke 0 tanpa batas.

Lihat https://golang. org/pkg/net/http/#Transport. IdleConnTimeout

Tetapkan `IdleConnTimeout` sebagai time.Duration.

Transportasi. MaxIdleConns

Pengaturan ini mewakili jumlah maksimum koneksi idle (keep-alive) di semua host. Salah satu kasus penggunaan untuk meningkatkan nilai ini adalah ketika Anda melihat banyak koneksi dalam waktu singkat dari klien yang sama

0 berarti tidak ada batas.

Lihat https://golang. org/pkg/net/http/#Transport. MaxIdleConns

Tetapkan `MaxIdleConns` sebagai int.

Transportasi. MaxIdleConnsPerHost

Pengaturan ini mewakili jumlah maksimum koneksi idle (keep-alive) untuk disimpan per host. Salah satu kasus penggunaan untuk meningkatkan nilai ini adalah ketika Anda melihat banyak koneksi dalam waktu singkat dari klien yang sama

Default adalah dua koneksi idle per host.

Setel ke 0 untuk menggunakan DefaultMaxIdleConnsPerHost (2).

Lihat https://golang. org/pkg/net/http/#Transport. MaxIdleConnsPerHost

Tetapkan `MaxIdleConnsPerHost` sebagai int.

Transportasi. ResponseHeaderTimeout

Pengaturan ini mewakili jumlah waktu maksimum untuk menunggu klien membaca header respons.

Jika klien tidak dapat membaca header respons dalam durasi ini, permintaan gagal dengan kesalahan batas waktu.

Hati-hati menyetel nilai ini saat menggunakan fungsi Lambda yang berjalan lama, karena operasi tidak mengembalikan header respons apa pun hingga fungsi Lambda selesai atau habis waktu. Namun, Anda masih dapat menggunakan opsi ini dengan operasi API InvokeAsync ** **.

Default adalah tidak ada batas waktu; tunggu selamanya.

Lihat https://golang. org/pkg/net/http/#Transport. ResponseHeaderTimeout

Tetapkan `ResponseHeaderTimeout` sebagai time.Duration.

Transportasi. TLSHandshakeBatas waktu

Pengaturan ini mewakili jumlah waktu maksimum menunggu jabat tangan TLS selesai.

Default-nya 10 detik.

Nol berarti tidak ada batas waktu.

Lihat https://golang. org/pkg/net/http/#Transport. TLSHandshakeBatas waktu

Tetapkan `TLSHandshakeTimeout` sebagai time.Duration.

Pencegat HTTP

Anda dapat menggunakan pencegat untuk menghubungkan eksekusi permintaan dan tanggapan API. Pencegat adalah mekanisme terbuka di mana SDK memanggil kode yang Anda tulis untuk menyuntikkan perilaku ke dalam siklus hidup. request/response Dengan cara ini, Anda dapat mengubah permintaan dalam penerbangan, proses permintaan debug, pengecualian tampilan, dan lainnya.

Pencegat vs middleware

AWS SDK for Go V2 menyediakan pencegat dan middleware untuk menyesuaikan pemrosesan permintaan. Meskipun keduanya melayani tujuan yang sama, keduanya dirancang untuk audiens dan kasus penggunaan yang berbeda:

- Interceptor dirancang untuk pengguna SDK yang ingin menyesuaikan request/response pemrosesan dengan API sederhana yang berfokus pada HTTP. Mereka menyediakan titik kait tertentu dalam siklus hidup permintaan dan bekerja secara langsung dengan permintaan dan tanggapan HTTP.
- Middleware adalah sistem transport-agnostik yang lebih canggih yang terutama digunakan secara internal oleh SDK. Meskipun kuat, middleware membutuhkan pengetahuan yang lebih dalam tentang internal SDK dan melibatkan antarmuka yang lebih kompleks.

Keuntungan utama pencegat dibandingkan middleware untuk kasus penggunaan umum:

- Berfokus pada HTTP: Interceptor bekerja secara langsung dengan permintaan dan tanggapan HTTP, menghilangkan kebutuhan untuk memeriksa tipe transport yang dibutuhkan middleware.
- Antarmuka yang lebih sederhana: Setiap kait pencegat memiliki antarmuka yang spesifik dan terfokus daripada pola middleware generik.
- Model eksekusi yang lebih jelas: Pencegat mengeksekusi pada titik yang terdefinisi dengan baik dalam siklus hidup permintaan tanpa memerlukan pengetahuan tentang pemesanan tumpukan middleware.

> **Note**
>
> Pencegat dibangun di atas sistem middleware yang ada, sehingga keduanya dapat hidup berdampingan dalam aplikasi yang sama. Middleware tetap tersedia untuk kasus penggunaan lanjutan yang memerlukan perilaku transport-agnostik atau manipulasi tumpukan kompleks.

Kait pencegat yang tersedia

AWS SDK for Go V2 menyediakan kait pencegat pada berbagai tahap siklus hidup permintaan. Setiap kait sesuai dengan antarmuka tertentu yang dapat Anda terapkan:

- `BeforeExecution`- Hook pertama dipanggil selama eksekusi operasi
- `BeforeSerialization`- Sebelum pesan masukan diserialisasikan ke dalam permintaan transport
- `AfterSerialization`- Setelah pesan masukan diserialisasikan ke dalam permintaan transportasi
- `BeforeRetryLoop`- Sebelum memasuki loop coba lagi
- `BeforeAttempt`- Kait pertama disebut di dalam loop coba lagi
- `BeforeSigning`- Sebelum permintaan transportasi ditandatangani
- `AfterSigning`- Setelah permintaan transportasi ditandatangani
- `BeforeTransmit`- Sebelum permintaan transportasi dikirim
- `AfterTransmit`- Setelah menerima respon transportasi
- `BeforeDeserialization`- Sebelum respon transportasi dideserialisasi

- `AfterDeserialization`- Setelah membongkar respons transportasi
- `AfterAttempt`- Kait terakhir disebut di dalam loop coba lagi
- `AfterExecution`- Hook terakhir dipanggil selama eksekusi operasi

Anda dapat mengimplementasikan beberapa antarmuka dalam satu pencegat untuk menghubungkan ke beberapa tahap siklus hidup permintaan.

Pendaftaran pencegat

Anda mendaftarkan pencegat ketika Anda membangun klien layanan atau ketika Anda mengganti konfigurasi untuk operasi tertentu. Pendaftaran berbeda tergantung pada apakah Anda ingin pencegat berlaku untuk semua operasi untuk klien Anda atau hanya yang spesifik.

Pencegat dikelola melalui registri pencegat yang menyediakan metode untuk menambah dan menghapus pencegat. Contoh berikut menunjukkan pencegat sederhana yang menambahkan header ID AWS X-Ray jejak ke permintaan keluar sebelum proses penandatanganan:

```
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{})
})
```

Registri pencegat ditambahkan ke Opsi klien, yang memungkinkan konfigurasi pencegat per operasi:

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

Konfigurasi pencegat global

Anda juga dapat mendaftarkan pencegat secara global menggunakan `config.LoadDefaultConfig` fungsi dengan `With*` opsi yang sesuai untuk setiap jenis pencegat. Ini berlaku pencegat untuk semua klien AWS layanan yang dibuat dari konfigurasi itu:

```
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)
```

Pencatatan log

Ini AWS SDK for Go memiliki fasilitas pencatatan yang tersedia yang memungkinkan aplikasi Anda mengaktifkan informasi debugging untuk debugging dan mendiagnosis masalah atau kegagalan permintaan. Antarmuka Logger dan ClientLogModemerupakan komponen utama yang tersedia bagi Anda untuk menentukan bagaimana dan apa yang harus dicatat oleh klien.

Pencatat

Saat membuat Config LoadDefaultConfigmenggunakan `Logger` default dikonfigurasi untuk mengirim pesan log ke kesalahan standar proses (stderr). Logger khusus yang memenuhi antarmuka Logger dapat diteruskan sebagai argumen `LoadDefaultConfig` dengan membungkusnya dengan konfigurasi. WithLogger.

Misalnya, untuk mengkonfigurasi klien kami untuk menggunakan`applicationLogger`:

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

Sekarang klien yang dikonfigurasi menggunakan yang dibangun `aws.Config` akan mengirim pesan log ke`applicationLogger`.

Logger Sadar Konteks

Implementasi Logger dapat mengimplementasikan ContextLoggerantarmuka opsional. Logger yang mengimplementasikan antarmuka ini akan memiliki `WithContext` metode mereka dipanggil dengan konteks saat ini. Ini memungkinkan implementasi logging Anda mengembalikan yang baru `Logger` yang dapat menulis metadata logging tambahan berdasarkan nilai yang ada dalam konteks.

ClientLogMode

Secara default, klien layanan tidak menghasilkan pesan log. Untuk mengonfigurasi klien untuk mengirim pesan log untuk tujuan debugging, gunakan ClientLogModeanggota di`Config`. `ClientLogMode`dapat diatur untuk mengaktifkan pesan debugging untuk:

- Tanda Tangan Versi 4 (SiGv4) Penandatanganan
- Minta Coba Ulang
- Permintaan HTTP
- Tanggapan HTTP

Misalnya, untuk mengaktifkan pencatatan permintaan HTTP dan percobaan ulang:

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

Lihat ClientLogModeuntuk berbagai mode log klien yang tersedia.

Coba Ulang dan Batas Waktu

AWS SDK for Go Ini memungkinkan Anda untuk mengonfigurasi perilaku percobaan ulang permintaan ke layanan HTTP. Secara default, klien layanan menggunakan Retry.standard sebagai retryer default mereka. Jika konfigurasi atau perilaku default tidak memenuhi persyaratan aplikasi, Anda dapat menyesuaikan konfigurasi retryer atau menyediakan implementasi retryer Anda sendiri.

AWS SDK for Go Ini menyediakan antarmuka AWS.Retryer yang mendefinisikan kumpulan metode yang diperlukan oleh implementasi coba lagi untuk diimplementasikan. SDK menyediakan dua implementasi untuk percobaan ulang: retry.Standard dan aws. NoOpRetryer.

Retryer Standar

Retry.standard retryer adalah `aws.Retryer` implementasi default yang digunakan oleh klien SDK. Retryer standar adalah retryer terbatas dengan jumlah upaya maksimal yang dapat dikonfigurasi, dan kemampuan untuk menyetel kebijakan mundur permintaan.

Tabel berikut mendefinisikan nilai default untuk retryer ini:

Properti	Default
Jumlah Upaya Maks	3
Penundaan Mundur Maks	20 detik

Ketika kesalahan yang dapat dicoba ulang terjadi saat menjalankan permintaan Anda, retryer standar akan menggunakan konfigurasi yang disediakan untuk menunda dan kemudian mencoba kembali permintaan tersebut. Mencoba lagi menambah latensi keseluruhan permintaan Anda, dan Anda harus mengonfigurasi retryer jika konfigurasi default tidak memenuhi persyaratan aplikasi Anda.

Lihat dokumentasi paket coba ulang untuk detail tentang kesalahan apa yang dianggap dapat dicoba ulang oleh implementasi retryer standar.

NopRetryer

Aw. NopRetryeradalah `aws.Retryer` implementasi yang disediakan jika Anda ingin menonaktifkan semua upaya coba lagi. Saat menjalankan operasi klien layanan, retryer ini hanya akan mengizinkan permintaan untuk dicoba sekali, dan kesalahan apa pun yang dihasilkan akan dikembalikan ke aplikasi panggilan.

Menyesuaikan Perilaku

SDK menyediakan satu set utilitas pembantu yang membungkus `aws.Retryer` implementasi, dan mengembalikan retryer yang disediakan yang dibungkus dengan perilaku coba lagi yang diinginkan.

Anda dapat mengganti retryer default untuk semua klien, per klien, atau per operasi tergantung pada kebutuhan aplikasi Anda. Untuk melihat contoh tambahan yang menunjukkan cara melakukannya, lihat contoh dokumentasi paket coba lagi.

> **Warning**
>
> Jika menentukan `aws.Retryer` implementasi global menggunakan`config.WithRetryer`, Anda harus memastikan bahwa Anda mengembalikan instance baru dari `aws.Retryer` setiap pemanggilan. Ini akan memastikan bahwa Anda tidak akan membuat ember token coba ulang global di semua klien layanan.

Membatasi jumlah maksimal upaya

Anda menggunakan coba lagi. AddWithMaxAttemptsuntuk membungkus `aws.Retryer` implementasi untuk mengatur upaya angka maksimal ke nilai yang Anda inginkan.

> **Warning**
>
> Menggunakan `retry.AddWithMaxAttempts` dengan nilai nol akan memungkinkan SDK untuk mencoba kembali semua kesalahan yang dapat dicoba ulang hingga permintaan berhasil, atau kesalahan yang tidak dapat dicoba kembali dikembalikan. Mengizinkan SDK untuk mencoba lagi tanpa batas dapat mengakibatkan beban kerja yang tidak terkendali dan siklus penagihan yang meningkat.

```
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)
```

Perhatikan bahwa membuat instance retryer menggunakan opsi fungsional untuk disetel MaxAttempts secara langsung akan memiliki perilaku yang sedikit berbeda. Lebih khusus lagi, menyetel nilai kurang dari atau sama dengan nol akan menyebabkan retryer menggunakan maksimum default 3 percobaan, daripada mencoba lagi tanpa batas:

```
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)
```

Membatasi penundaan mundur maksimal

Anda menggunakan coba lagi. AddWithMaxBackoffDelayuntuk membungkus `aws.Retryer` implementasi dan membatasi penundaan mundur maksimal yang diizinkan terjadi antara mencoba kembali permintaan yang gagal.

Misalnya, Anda dapat menggunakan kode berikut untuk membungkus retryer klien standar dengan penundaan maksimal lima detik yang diinginkan:

```
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)
```

Coba lagi kode kesalahan API tambahan

Anda menggunakan coba lagi. AddWithErrorCodesuntuk membungkus `aws.Retryer` implementasi dan menyertakan kode kesalahan API tambahan yang harus dianggap dapat dicoba ulang.

Misalnya, Anda dapat menggunakan kode berikut untuk membungkus retryer klien standar untuk menyertakan pengecualian Amazon `NoSuchBucketException` S3 sebagai dapat dicoba ulang.

```
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)
```

Pembatasan tingkat sisi klien

Ini AWS SDK for Go memperkenalkan mekanisme pembatasan tingkat sisi klien baru dalam kebijakan coba ulang standar untuk menyelaraskan dengan perilaku modern. SDKs Ini adalah perilaku yang dikendalikan oleh RateLimiterbidang pada opsi retryer.

A RateLimiter beroperasi sebagai ember token dengan kapasitas yang ditetapkan, di mana kegagalan upaya operasi menggunakan token. Coba lagi yang mencoba mengkonsumsi lebih banyak token daripada yang tersedia menghasilkan kegagalan operasi dengan a QuotaExceededError.

Implementasi default diparameterisasi sebagai berikut (cara memodifikasi setiap pengaturan):

- kapasitas 500 (mengatur nilai RateLimiter pada StandardOptions penggunaan NewTokenRateLimit)
- coba lagi yang disebabkan oleh biaya batas waktu 10 token (ditetapkan RetryTimeoutCost) StandardOptions
- percobaan lagi yang disebabkan oleh kesalahan lain membutuhkan biaya 5 token (disetel RetryCost) StandardOptions
- operasi yang berhasil pada upaya pertama menambahkan 1 token (diatur NoRetryIncrement) StandardOptions
  - operasi yang berhasil pada upaya ke-2 atau yang lebih baru tidak menambahkan kembali token apa pun

Jika Anda menemukan bahwa perilaku default tidak sesuai dengan kebutuhan aplikasi Anda, Anda dapat menonaktifkannya dengan RateLimit.none.

Contoh: pembatas tingkat yang dimodifikasi

```
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
    })
}))
```

Contoh: tidak ada batas tarif menggunakan 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
    })
}))
```

Timeout

Anda menggunakan paket konteks untuk mengatur batas waktu atau tenggat waktu saat menjalankan operasi klien layanan. Gunakan konteksnya. WithDeadlineuntuk membungkus konteks aplikasi Anda dan menetapkan tenggat waktu ke waktu tertentu dimana operasi yang dipanggil harus diselesaikan. Untuk mengatur batas waktu setelah konteks `time.Duration` penggunaan tertentu. WithTimeout. SDK meneruskan yang disediakan `context.Context` ke klien transport HTTP saat menjalankan API layanan. Jika konteks yang diteruskan ke SDK dibatalkan atau dibatalkan saat

menjalankan operasi, SDK tidak akan mencoba lagi permintaan lebih lanjut dan akan kembali ke aplikasi panggilan. Anda harus menangani pembatalan konteks dengan tepat dalam aplikasi Anda jika konteks yang diberikan kepada SDK telah dibatalkan.

Menetapkan batas waktu

Contoh berikut menunjukkan cara mengatur batas waktu untuk operasi klien layanan.

```
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
}
```

Menggunakan AWS SDK for Go

Pelajari cara pemrograman yang umum dan direkomendasikan dengan aplikasi Anda. AWS SDK for Go

Topik

- Membangun Klien Layanan
- Operasi Layanan Panggilan
- Serentak Menggunakan Klien Layanan
- Menggunakan Operation Paginators
- Menggunakan Pelayan
- Menangani Kesalahan di AWS SDK for Go V2

Membangun Klien Layanan

Klien layanan dapat dibangun menggunakan `NewFromConfig` fungsi `New` atau yang tersedia dalam paket Go klien layanan. Setiap fungsi akan mengembalikan tipe `Client` struct yang berisi metode untuk menjalankan layanan. APIs Ini `New` dan `NewFromConfig` masing-masing menyediakan serangkaian opsi yang dapat dikonfigurasi yang sama untuk membangun klien layanan, tetapi memberikan pola konstruksi yang sedikit berbeda yang akan kita lihat di bagian berikut.

NewFromConfig

`NewFromConfig`fungsi menyediakan antarmuka yang konsisten untuk membangun klien layanan menggunakan AWS.config. An `aws.Config` dapat dimuat menggunakan konfigurasi. LoadDefaultConfig. Untuk informasi lebih lanjut tentang membangun sebuah`aws.Config`, lihatKonfigurasikan SDK. Contoh berikut menunjukkan cara membuat klien layanan Amazon S3 menggunakan dan `aws.Config` `NewFromConfig` fungsi:

```
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)
```

Mengesampingkan Konfigurasi

`NewFromConfig`dapat mengambil satu atau lebih argumen fungsional yang dapat mengubah `Options` struct konfigurasi klien. Ini memungkinkan Anda untuk membuat penggantian tertentu seperti mengubah Wilayah, atau memodifikasi opsi khusus layanan seperti opsi Amazon S3. `UseAccelerate` Misalnya:

```
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
    })
```

Penggantian ke `Options` nilai klien ditentukan oleh urutan argumen fungsional yang diberikan. `NewFromConfig`

Baru

> **Note**
>
> Newdianggap sebagai bentuk konstruksi klien yang lebih maju. Kami menyarankan Anda menggunakan `NewFromConfig` untuk konstruksi klien, karena memungkinkan konstruksi

menggunakan `aws.Config` struct. Ini menghilangkan kebutuhan untuk membuat instance `Options` struct untuk setiap klien layanan yang dibutuhkan aplikasi Anda.

`New`fungsi adalah konstruktor klien menyediakan antarmuka untuk membangun klien hanya menggunakan paket klien `Options` struct untuk mendefinisikan opsi konfigurasi klien. Misalnya, untuk membangun klien Amazon S3 menggunakan: `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,
 "")),
    })
```

Mengesampingkan Konfigurasi

`New`dapat mengambil satu atau lebih argumen fungsional yang dapat mengubah `Options` struct konfigurasi klien. Ini memungkinkan Anda untuk membuat penggantian tertentu seperti mengubah Wilayah atau memodifikasi opsi khusus layanan seperti opsi Amazon S3. `UseAccelerate` Misalnya:

```
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
    })
```

Penggantian ke `Options` nilai klien ditentukan oleh urutan argumen fungsional yang diberikan. `New`

Operasi Layanan Panggilan

Setelah Anda memiliki instance klien layanan, Anda dapat menggunakannya untuk memanggil operasi layanan. Misalnya, untuk memanggil operasi Amazon S3`GetObject`:

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

Saat Anda memanggil operasi layanan, SDK secara sinkron memvalidasi input, membuat serial permintaan, menandatanganinya dengan kredensyal Anda, mengirimkannya ke AWS, dan kemudian melakukan deserialisasi respons atau kesalahan. Dalam kebanyakan kasus, Anda dapat menghubungi operasi layanan secara langsung. Setiap metode klien operasi layanan akan mengembalikan struct respons operasi, dan jenis antarmuka kesalahan. Anda harus selalu memeriksa `error` jenis untuk menentukan apakah terjadi kesalahan sebelum mencoba mengakses struct respons operasi layanan.

Melewati Parameter ke Operasi Layanan

Setiap metode operasi layanan mengambil nilai context.Context yang dapat digunakan untuk menetapkan tenggat waktu permintaan yang akan dihormati oleh SDK. Selain itu, setiap operasi layanan akan mengambil `<OperationName>Input` struct yang ditemukan di paket Go masing-masing layanan. Anda meneruskan parameter input API menggunakan struct input operasi.

Struktur input operasi dapat memiliki parameter input seperti angka Go standar, boolean, string, peta, dan jenis daftar. Dalam operasi API yang lebih kompleks, layanan mungkin memiliki pemodelan parameter input yang lebih kompleks. Jenis lain seperti struktur khusus layanan dan nilai enum ditemukan dalam paket `types` Go layanan.

Selain itu, layanan dapat membedakan antara nilai default tipe Go dan apakah nilai ditetapkan atau tidak oleh pengguna. Dalam kasus ini, parameter input mungkin mengharuskan Anda untuk meneruskan referensi pointer ke jenis yang dimaksud. Untuk tipe Go standar seperti numerik, boolean, dan string ada `<Type>` dan fungsi `From<Type>` kenyamanan yang tersedia di aws untuk memudahkan konversi ini. Misalnya, AWS.String dapat digunakan untuk mengonversi a `string` ke `*string` tipe untuk parameter input yang memerlukan pointer ke string. Sebaliknya, aws. ToStringdapat digunakan untuk mengubah a `*string` ke a `string` sambil memberikan perlindungan dari dereferensi pointer nil. `To<Type>`Fungsinya sangat membantu saat menangani respons layanan.

Mari kita lihat contoh bagaimana kita dapat menggunakan klien Amazon S3 untuk memanggil `GetObject` API, dan membangun input kita menggunakan `types` paket, dan pembantu. `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,
    })
```

Mengganti Opsi Klien Untuk Panggilan Operasi

Mirip dengan bagaimana opsi operasi klien dapat dimodifikasi selama konstruksi klien menggunakan argumen fungsional, opsi klien dapat dimodifikasi pada saat metode operasi dipanggil dengan

memberikan satu atau lebih argumen fungsional untuk metode operasi layanan. Tindakan ini aman konkurensi dan tidak akan mempengaruhi operasi bersamaan lainnya pada klien.

Misalnya, untuk mengganti wilayah klien dari “us-west-2" ke “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"
    })
```

Menangani Tanggapan Operasi

Setiap operasi layanan memiliki struct keluaran terkait yang berisi anggota respons operasi layanan. Output struct mengikuti pola `<OperationName>Output` penamaan berikut. Beberapa operasi mungkin tidak memiliki anggota yang ditentukan untuk output operasi mereka. Setelah memanggil operasi layanan, tipe `error` argumen pengembalian harus selalu diperiksa untuk menentukan apakah terjadi kesalahan saat menjalankan operasi layanan. Kesalahan yang dikembalikan dapat berkisar dari kesalahan validasi masukan sisi klien hingga respons kesalahan sisi layanan yang dikembalikan ke klien. Struct keluaran operasi tidak boleh diakses jika kesalahan non-nil dikembalikan oleh klien.

Misalnya, untuk mencatat kesalahan operasi dan kembali sebelum waktunya dari fungsi panggilan:

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

Untuk informasi selengkapnya tentang penanganan kesalahan, termasuk cara memeriksa jenis kesalahan tertentu, lihat TODO

Tanggapan dengan `io.ReadCloser`

Beberapa operasi API mengembalikan struct respons yang berisi anggota keluaran yang merupakan `io.ReadCloser` file. Ini akan menjadi kasus untuk operasi API yang mengekspos beberapa elemen outputnya di badan respons HTTP itu sendiri.

Misalnya, `GetObject` operasi Amazon S3 mengembalikan respons yang `Body` anggotanya adalah `io.ReadCloser` untuk mengakses payload objek.

> **Warning**
>
> Anda HARUS SELALU `Close()` setiap anggota `io.ReadCloser` output, terlepas dari apakah Anda telah mengkonsumsi kontennya. Kegagalan untuk melakukannya dapat membocorkan sumber daya dan berpotensi menimbulkan masalah dengan membaca badan respons untuk operasi yang disebut di masa depan.

```
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
    }
```

Metadata Respon

Semua struct output operasi layanan termasuk `ResultMetadata` anggota tipe Middleware.metadata. `middleware.Metadata`digunakan oleh middleware SDK untuk memberikan informasi tambahan dari respons layanan yang tidak dimodelkan oleh layanan. Ini termasuk metadata

seperti. `RequestID` Misalnya, untuk mengambil respons `RequestID` terkait dengan layanan untuk membantu AWS Support dalam memecahkan masalah permintaan:

```
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)
```

Serentak Menggunakan Klien Layanan

Anda dapat membuat goroutine yang secara bersamaan menggunakan klien layanan yang sama untuk mengirim beberapa permintaan. Anda dapat menggunakan klien layanan dengan goroutine sebanyak yang Anda inginkan.

Dalam contoh berikut, klien layanan Amazon S3 digunakan di beberapa goroutine. Contoh ini secara bersamaan mengunggah dua objek ke bucket Amazon S3.

```
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))
    }
```

Menggunakan Operation Paginators

Biasanya, saat Anda mengambil daftar item, Anda mungkin perlu memeriksa struct keluaran untuk token atau penanda untuk mengonfirmasi apakah AWS layanan mengembalikan semua hasil dari permintaan Anda. Jika token atau penanda ada, Anda menggunakannya untuk meminta halaman hasil berikutnya. Alih-alih mengelola token atau penanda ini, Anda dapat menggunakan jenis paginator paket layanan yang tersedia.

Pembantu paginator tersedia untuk operasi layanan yang didukung, dan dapat ditemukan dalam paket Go klien layanan. Untuk membangun paginator untuk operasi yang didukung, gunakan fungsi. `New<OperationName>Paginator` Fungsi konstruksi paginator mengambil layanan`Client`, parameter `<OperationName>Input` input operasi, dan serangkaian argumen fungsional opsional yang memungkinkan Anda mengonfigurasi pengaturan paginator opsional lainnya.

Jenis paginator operasi yang dikembalikan menyediakan cara mudah untuk mengulangi operasi paginasi sampai Anda mencapai halaman terakhir, atau Anda telah menemukan item yang dicari aplikasi Anda. Tipe paginator memiliki dua metode: `HasMorePages` dan. `NextPage` `HasMorePages`mengembalikan nilai boolean `true` jika halaman pertama belum diambil, atau jika halaman tambahan tersedia untuk mengambil menggunakan operasi. Untuk mengambil halaman pertama atau selanjutnya dari operasi, `NextPage` operasi harus dipanggil. `NextPage`mengambil `context.Context` dan mengembalikan output operasi dan kesalahan yang sesuai. Seperti parameter pengembalian metode operasi klien, kesalahan pengembalian harus selalu diperiksa sebelum mencoba menggunakan struktur respons yang dikembalikan. Lihat Menangani Tanggapan Operasi.

Contoh berikut menggunakan `ListObjectsV2` paginator untuk daftar hingga tiga halaman kunci objek dari operasi. `ListObjectV2` Setiap halaman terdiri dari hingga 10 kunci, yang ditentukan oleh opsi `Limit` paginator.

```
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++
    }
```

Mirip dengan metode operasi klien, opsi klien seperti Region permintaan dapat dimodifikasi dengan memberikan satu atau lebih argumen fungsional untuk`NextPage`. Untuk informasi selengkapnya tentang mengganti opsi klien saat memanggil operasi, lihatMengganti Opsi Klien Untuk Panggilan Operasi.

Menggunakan Pelayan

Saat berinteraksi dengan AWS APIs yang asinkron, Anda sering harus menunggu sumber daya tertentu tersedia untuk melakukan tindakan lebih lanjut di atasnya.

Misalnya, Amazon `CreateTable` DynamoDB API segera kembali dengan TableStatus CREATING, dan Anda tidak dapat menjalankan operasi baca atau tulis hingga status tabel dialihkan. `ACTIVE`

Logika penulisan untuk terus melakukan polling status tabel dapat menjadi rumit dan rawan kesalahan. Para pelayan membantu menghilangkan kerumitan dari itu dan sederhana APIs yang menangani tugas pemungutan suara untuk Anda.

Misalnya, Anda dapat menggunakan pelayan untuk melakukan polling jika tabel DynamoDB dibuat dan siap untuk operasi tulis.

```
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")
```

Mengesampingkan konfigurasi pelayan

Secara default, SDK menggunakan penundaan minimum dan nilai penundaan maksimum yang dikonfigurasi dengan nilai optimal yang ditentukan oleh AWS layanan untuk yang berbeda APIs. Anda dapat mengganti konfigurasi pelayan dengan memberikan opsi fungsional selama konstruksi pelayan, atau saat menjalankan operasi pelayan.

Misalnya, untuk mengganti konfigurasi pelayan selama konstruksi pelayan

```
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
    })
```

`Wait`Fungsi pada setiap pelayan juga mengambil opsi fungsional. Mirip dengan contoh di atas, Anda dapat mengganti konfigurasi pelayan per `Wait` permintaan.

```
// 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")
```

Penggantian konfigurasi pelayan tingkat lanjut

Anda juga dapat menyesuaikan perilaku default pelayan dengan menyediakan fungsi yang dapat dicoba ulang khusus. Opsi khusus pelayan juga menyediakan `APIOptions` untuk menyesuaikan middlewares operasi.

Misalnya, untuk mengkonfigurasi penggantian pelayan tingkat lanjut.

```
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
    })
```

Menangani Kesalahan di AWS SDK for Go V2

AWS SDK for Go Mengembalikan kesalahan yang memenuhi tipe `error` antarmuka Go. Anda dapat menggunakan `Error()` metode ini untuk mendapatkan string pesan kesalahan SDK yang diformat tanpa penanganan khusus. Kesalahan yang dikembalikan oleh SDK dapat mengimplementasikan `Unwrap` metode. `Unwrap`Metode ini digunakan oleh SDK untuk memberikan informasi kontekstual tambahan untuk kesalahan, sambil memberikan akses ke kesalahan atau rantai kesalahan yang mendasarinya. `Unwrap`Metode ini harus digunakan dengan errors.as untuk menangani rantai kesalahan membuka bungkusan.

Penting agar aplikasi Anda memeriksa apakah terjadi kesalahan setelah menjalankan fungsi atau metode yang dapat mengembalikan jenis `error` antarmuka. Bentuk paling dasar dari penanganan kesalahan terlihat mirip dengan contoh berikut:

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

Kesalahan Pencatatan

Bentuk paling sederhana dari penanganan kesalahan secara tradisional untuk mencatat atau mencetak pesan kesalahan sebelum kembali atau keluar dari aplikasi.

```
import "log"

// ...

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

Kesalahan Layanan Klien

SDK membungkus semua kesalahan yang dikembalikan oleh klien layanan dengan bengkel. OperationErrorjenis kesalahan. `OperationError`memberikan informasi kontekstual tentang nama layanan dan operasi yang terkait dengan kesalahan yang mendasarinya. Informasi ini dapat berguna untuk aplikasi yang melakukan batch operasi ke satu atau lebih layanan, dengan mekanisme penanganan kesalahan terpusat. Aplikasi Anda dapat digunakan `errors.As` untuk mengakses `OperationError` metadata ini.

```
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
}
```

Tanggapan Kesalahan API

Operasi layanan dapat mengembalikan jenis kesalahan yang dimodelkan untuk menunjukkan kesalahan tertentu. Jenis model ini dapat digunakan `errors.As` untuk membuka dan menentukan apakah kegagalan operasi disebabkan oleh kesalahan tertentu. Misalnya, Amazon S3 `CreateBucket` dapat mengembalikan BucketAlreadyExistskesalahan jika ember dengan nama yang sama sudah ada.

Misalnya, untuk memeriksa apakah kesalahan adalah `BucketAlreadyExists` kesalahan:

```
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
}
```

Semua kesalahan respons API layanan mengimplementasikan bengkel. APIErrorjenis antarmuka. Antarmuka ini dapat digunakan untuk menangani respons kesalahan layanan yang dimodelkan atau tidak dimodelkan. Jenis ini menyediakan akses ke kode kesalahan dan pesan yang dikembalikan oleh layanan. Selain itu, jenis ini memberikan indikasi apakah kesalahan kesalahan disebabkan oleh klien atau server jika diketahui.

```
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
}
```

Mengambil Pengidentifikasi Permintaan

Ketika bekerja dengan AWS Support, Anda mungkin diminta untuk memberikan pengenal permintaan yang mengidentifikasi permintaan yang Anda coba untuk memecahkan masalah. Anda dapat menggunakan http. ResponseErrordan gunakan `ServiceRequestID()` metode untuk mengambil pengenal permintaan yang terkait dengan respons kesalahan.

```
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
}
```

Pengidentifikasi Permintaan Amazon S3

Permintaan Amazon S3 berisi pengenal tambahan yang dapat digunakan untuk membantu AWS Support memecahkan masalah permintaan Anda. Anda dapat menggunakan s3. ResponseErrordan panggil `ServiceRequestID()` dan `ServiceHostID()` untuk mengambil ID permintaan dan ID 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
}
```

Gunakan AWS SDK for Go v2 dengan AWS layanan

Untuk melakukan panggilan ke AWS layanan, Anda harus terlebih dahulu membuat instance klien layanan. Klien layanan menyediakan akses tingkat rendah ke setiap tindakan API untuk layanan tersebut. Misalnya, Anda membuat klien layanan Amazon S3 untuk melakukan panggilan ke Amazon S3. APIs

Saat Anda memanggil operasi layanan, Anda meneruskan parameter input sebagai struct. Panggilan yang berhasil akan menghasilkan struct keluaran yang berisi respons API layanan. Misalnya, setelah Anda berhasil memanggil aksi bucket Amazon S3 create, tindakan akan mengembalikan struct keluaran dengan lokasi bucket.

Untuk daftar klien layanan, termasuk metode dan parameternya, lihat Referensi AWS SDK for Go API.

Perlindungan integritas data dengan checksum

Amazon Simple Storage Service (Amazon S3) menyediakan kemampuan untuk menentukan checksum saat Anda mengunggah objek. Ketika Anda menentukan checksum, itu disimpan dengan objek dan dapat divalidasi ketika objek diunduh.

Checksum menyediakan lapisan integritas data tambahan saat Anda mentransfer file. Dengan checksum, Anda dapat memverifikasi konsistensi data dengan mengonfirmasi bahwa file yang diterima cocok dengan file asli. Untuk informasi selengkapnya tentang checksum dengan Amazon S3, lihat Panduan Pengguna Layanan Penyimpanan Sederhana Amazon termasuk algoritme yang didukung.

Anda memiliki fleksibilitas untuk memilih algoritma yang paling sesuai dengan kebutuhan Anda dan membiarkan SDK menghitung checksum. Atau, Anda dapat memberikan nilai checksum yang telah dihitung sebelumnya dengan menggunakan salah satu algoritme yang didukung.

> **Note**
>
> Dimulai dengan versi v1.74.1 dari modul Amazon S3, SDK menyediakan perlindungan integritas default dengan secara otomatis menghitung checksum untuk unggahan. `CRC32` SDK menghitung checksum ini jika Anda tidak memberikan nilai checksum yang telah dihitung sebelumnya atau jika Anda tidak menentukan algoritme yang harus digunakan SDK untuk menghitung checksum.

SDK juga menyediakan pengaturan global untuk perlindungan integritas data yang dapat Anda atur secara eksternal, yang dapat Anda baca di Panduan Referensi Alat AWS SDKs dan Alat.

Kami membahas checksum dalam dua fase permintaan: mengunggah objek dan mengunduh objek.

Mengunggah objek

Saat Anda mengunggah objek dengan `putObject` metode dan menyediakan algoritma checksum, SDK menghitung checksum untuk algoritme yang ditentukan.

Cuplikan kode berikut menunjukkan permintaan untuk mengunggah objek dengan checksum. `CRC32` Ketika SDK mengirim permintaan, ia menghitung `CRC32` checksum dan mengunggah objek. Amazon S3 memvalidasi integritas konten dengan menghitung checksum dan membandingkannya dengan checksum yang disediakan oleh SDK. Amazon S3 kemudian menyimpan checksum dengan objek.

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

Jika Anda tidak menyediakan algoritma checksum dengan permintaan, perilaku checksum bervariasi tergantung pada versi SDK yang Anda gunakan seperti yang ditunjukkan pada tabel berikut.

Perilaku checksum ketika tidak ada algoritma checksum yang disediakan

Amazon S3 versi modul AWS SDK for Go	Perilaku checksum
Lebih awal dari v1.74.1	SDK tidak secara otomatis menghitung checksum berbasis CRC dan menyediakannya dalam permintaan.
v1.74.1 atau yang lebih baru	SDK menggunakan `CRC32` algoritma untuk menghitung checksum dan menyediakannya dalam permintaan. Amazon S3 memvalida si integritas transfer dengan menghitung

Amazon S3 versi modul AWS SDK for Go	Perilaku checksum
	checksumnya sendiri dan membandingkannya dengan CRC32 checksum yang disediakan oleh SDK. Jika checksum cocok, checksum disimpan dengan objek.

Gunakan nilai checksum yang telah dihitung sebelumnya

Nilai checksum yang telah dihitung sebelumnya yang disertakan dengan permintaan menonaktifkan komputasi otomatis oleh SDK dan menggunakan nilai yang disediakan sebagai gantinya.

Contoh berikut menunjukkan permintaan dengan SHA256 checksum yang telah dihitung sebelumnya.

```
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"),
 })
```

Jika Amazon S3 menentukan nilai checksum salah untuk algoritme yang ditentukan, layanan akan mengembalikan respons kesalahan.

Unggahan multipart

Anda juga dapat menggunakan checksum dengan unggahan multipart.

AWS SDK for Go Ini menyediakan dua opsi untuk menggunakan checksum dengan unggahan multibagian. Opsi pertama menggunakan manajer transfer yang menentukan CRC32 algoritma untuk unggahan.

```
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,
```

```
    })
```

Jika Anda tidak menyediakan algoritma checksum saat menggunakan manajer transfer untuk upload, SDK secara otomatis menghitung dan checksum berdasarkan algoritma. `CRC32` SDK melakukan perhitungan ini untuk semua versi SDK.

Opsi kedua menggunakan klien Amazon S3 untuk melakukan pengunggahan multibagian. Jika Anda menentukan checksum dengan pendekatan ini, Anda harus menentukan algoritma yang akan digunakan pada inisiasi unggahan. Anda juga harus menentukan algoritma untuk setiap permintaan bagian dan memberikan checksum yang dihitung untuk setiap bagian setelah diunggah.

```
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)
    }
```

Unduh objek

Saat Anda menggunakan GetObjectmetode untuk mengunduh objek, SDK secara otomatis memvalidasi checksum saat `ChecksumMode` bidang disetel ke. `GetObjectInput` `types.ChecksumModeEnabled`

Permintaan dalam cuplikan berikut mengarahkan SDK untuk memvalidasi checksum dalam respons dengan menghitung checksum dan membandingkan nilainya.

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

Jika objek tidak diunggah dengan checksum, tidak ada validasi yang terjadi.

Menggunakan AWS SDK for Go Utilitas

AWS SDK for Go Ini termasuk utilitas berikut untuk membantu Anda lebih mudah menggunakan AWS layanan. Temukan utilitas SDK dalam paket AWS layanan terkait mereka.

Utilitas Amazon RDS

IAM Authentication.

Paket autentikasi menyediakan utilitas untuk menghasilkan token otentikasi untuk menghubungkan ke instance database Amazon RDS MySQL dan PostgreSQL. Dengan menggunakan BuildAuthTokenmetode ini, Anda membuat token otorisasi basis data dengan menyediakan titik akhir database, AWS Wilayah, nama pengguna, dan aws. CredentialProviderimplementasi yang mengembalikan kredensi IAM dengan izin untuk terhubung ke database menggunakan otentikasi database IAM. Untuk mempelajari selengkapnya tentang mengonfigurasi Amazon RDS dengan autentikasi IAM, lihat sumber daya Panduan Pengembang Amazon RDS berikut:

- Mengaktifkan dan menonaktifkan otentikasi database IAM
- Membuat dan menggunakan kebijakan IAM untuk akses database IAM
- Membuat akun database menggunakan otentikasi IAM

Contoh berikut menunjukkan cara menghasilkan token otentikasi untuk terhubung ke database 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())
}
```

CloudFront Utilitas Amazon

Penandatangan CloudFront URL Amazon

Penandatangan CloudFront URL Amazon menyederhanakan proses pembuatan ditandatangani. URLs URL yang ditandatangani mencakup informasi, seperti tanggal dan waktu kedaluwarsa, yang memungkinkan Anda mengontrol akses ke konten Anda. Ditandatangani URLs berguna ketika Anda ingin mendistribusikan konten melalui internet, tetapi ingin membatasi akses ke pengguna tertentu (misalnya, untuk pengguna yang telah membayar biaya).

Untuk menandatangani URL, buat `URLSigner` instance dengan ID CloudFront key pair Anda dan kunci pribadi terkait. Kemudian panggil `SignWithPolicy` metode `Sign` or dan sertakan URL yang akan ditandatangani. Untuk informasi selengkapnya tentang pasangan CloudFront kunci Amazon, lihat Membuat Pasangan CloudFront Kunci untuk Penandatangan Tepercaya Anda di Panduan CloudFront Pengembang.

Contoh berikut membuat URL yang ditandatangani yang valid selama satu jam setelah dibuat.

```
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
}
```

Untuk informasi selengkapnya tentang utilitas penandatanganan, lihat paket tanda di Referensi AWS SDK for Go API.

Layanan EC2 Metadata Instans Amazon

Anda dapat menggunakan AWS SDK for Go untuk mengakses Layanan Metadata EC2 Instans Amazon. Paket feature/ec2/imdsGo menyediakan tipe Klien yang dapat digunakan untuk mengakses Layanan Metadata EC2 Instans Amazon. Operasi `Client` dan terkait dapat digunakan mirip dengan klien AWS layanan lain yang disediakan oleh SDK. Untuk mempelajari informasi selengkapnya tentang cara mengonfigurasi SDK, dan menggunakan klien layanan, lihat Konfigurasikan SDK danGunakan AWS SDK for Go v2 dengan AWS layanan.

Klien dapat membantu Anda dengan mudah mengambil informasi tentang instance di mana aplikasi Anda berjalan, seperti AWS Wilayah atau alamat IP lokal. Biasanya, Anda harus membuat dan mengirimkan permintaan HTTP untuk mengambil metadata instance. Sebagai gantinya, buat `imds.Client` untuk mengakses Layanan Metadata EC2 Instans Amazon menggunakan klien terprogram seperti Layanan lainnya. AWS

Misalnya untuk membangun klien:

```
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)
```

Kemudian gunakan klien layanan untuk mengambil informasi dari kategori metadata seperti `local-ipv4` (alamat IP pribadi dari instance).

```
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))
```

Untuk daftar semua kategori metadata, lihat Kategori metadata instans di Panduan Pengguna Amazon. EC2

Utilitas Amazon S3

Manajer Transfer Amazon S3

Pengelola unggahan dan unduhan Amazon S3 dapat memecah objek besar, sehingga dapat ditransfer dalam beberapa bagian, secara paralel. Ini memudahkan untuk melanjutkan transfer yang terputus.

Manajer Unggahan Amazon S3

Manajer unggahan Amazon S3 menentukan apakah file dapat dibagi menjadi bagian-bagian yang lebih kecil dan diunggah secara paralel. Anda dapat menyesuaikan jumlah upload paralel dan ukuran bagian yang diunggah.

Contoh berikut menggunakan Amazon S3 `Uploader` untuk mengunggah file. `Uploader`Penggunaannya mirip dengan `s3.PutObject()` operasi.

```
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,
})
```

Opsi konfigurasi

Saat membuat instance menggunakan `Uploader` instance NewUploader, Anda dapat menentukan beberapa opsi konfigurasi untuk menyesuaikan cara objek diunggah. Opsi diganti dengan memberikan satu atau lebih argumen ke. `NewUploader` Pilihan ini meliputi:

- `PartSize`- Menentukan ukuran buffer, dalam byte, dari setiap bagian untuk meng-upload. Ukuran minimum per bagian adalah 5 MiB.
- `Concurrency`- Menentukan jumlah bagian untuk meng-upload secara parallel.
- `LeavePartsOnError`— Menunjukkan apakah akan meninggalkan bagian yang berhasil diunggah di Amazon S3.

`Concurrency`Nilai membatasi jumlah bersamaan dari bagian upload yang dapat terjadi untuk panggilan tertentu`Upload`. Ini bukan batas konkurensi klien global. Tweak nilai `PartSize` dan `Concurrency` konfigurasi untuk menemukan konfigurasi yang optimal. Misalnya, sistem dengan koneksi bandwidth tinggi dapat mengirim bagian yang lebih besar dan lebih banyak unggahan secara paralel.

Misalnya, aplikasi Anda mengonfigurasi `Uploader` dengan `Concurrency` pengaturan. `5` Jika aplikasi Anda kemudian memanggil `Upload` dari dua goroutine yang berbeda, hasilnya adalah unggahan bagian `10` bersamaan (2 goroutine* 5). `Concurrency`

> **Warning**
>
> Aplikasi Anda diharapkan membatasi panggilan bersamaan `Upload` untuk mencegah kehabisan sumber daya aplikasi.

Di bawah ini adalah contoh untuk mengatur ukuran bagian default selama `Uploader` pembuatan:

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

Untuk informasi selengkapnya tentang `Uploader` dan konfigurasinya, lihat Pengunggah di Referensi AWS SDK for Go API.

PutObjectInput Bidang Tubuh (io. ReadSeeker vs io.reader)

`Body`Bidang `s3.PutObjectInput` struct adalah `io.Reader` tipe. Namun, bidang ini dapat diisi dengan jenis yang memenuhi `io.ReaderAt` antarmuka `io.ReadSeeker` dan antarmuka untuk meningkatkan pemanfaatan sumber daya aplikasi dari lingkungan host. Contoh berikut menciptakan tipe `ReadSeekerAt` yang memenuhi kedua antarmuka:

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

Untuk `io.Reader` jenis, byte pembaca harus di-buffer dalam memori sebelum bagian dapat diunggah. Ketika Anda meningkatkan `PartSize` atau `Concurrency` nilai, memori yang diperlukan (RAM) untuk `Uploader` meningkat secara signifikan. Memori yang dibutuhkan kira-kira *`PartSize`* `Concurrency`*. Misalnya, menentukan 100 MB untuk `PartSize` dan 10 untuk`Concurrency`, membutuhkan setidaknya 1 GB.

Karena suatu `io.Reader` tipe tidak dapat menentukan ukurannya sebelum membaca byte-nya, `Uploader` tidak dapat menghitung berapa banyak bagian yang akan diunggah. Akibatnya, `Uploader` dapat mencapai batas unggah Amazon S3 10.000 bagian untuk file besar jika Anda mengatur `PartSize` terlalu rendah. Jika Anda mencoba mengunggah lebih dari 10.000 bagian, unggahan berhenti dan mengembalikan kesalahan.

Untuk body nilai yang mengimplementasikan `ReadSeekerAt` tipe, `Uploader` tidak menyangga isi isi isi dalam memori sebelum mengirimnya ke Amazon S3. `Uploader`menghitung jumlah bagian yang diharapkan sebelum mengunggah file ke Amazon S3. Jika nilai saat ini `PartSize` membutuhkan lebih dari 10.000 bagian untuk mengunggah file, `Uploader` tingkatkan nilai ukuran bagian sehingga lebih sedikit bagian yang diperlukan.

Menangani Unggahan Gagal

Jika unggahan ke Amazon S3 gagal, secara default, `Uploader` gunakan operasi Amazon `AbortMultipartUpload` S3 untuk menghapus bagian yang diunggah. Fungsionalitas ini memastikan bahwa unggahan yang gagal tidak menggunakan penyimpanan Amazon S3.

Anda dapat mengatur `LeavePartsOnError` ke true sehingga `Uploader` tidak menghapus bagian yang berhasil diunggah. Ini berguna untuk melanjutkan unggahan yang sebagian selesai. Untuk beroperasi pada bagian yang diunggah, Anda harus mendapatkan `UploadID` unggahan yang gagal. Contoh berikut menunjukkan bagaimana menggunakan jenis antarmuka `manager.MultiUploadFailure` kesalahan untuk mendapatkan. `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
}
```

Mengganti Opsi Pengunggah Per Unggahan

Anda dapat mengganti `Uploader` opsi saat memanggil `Upload` dengan memberikan satu atau beberapa argumen ke metode. Penggantian ini adalah modifikasi aman konkurensi dan tidak memengaruhi unggahan yang sedang berlangsung, atau panggilan berikutnya ke manajer. `Upload` Misalnya, untuk mengganti `PartSize` konfigurasi untuk permintaan unggahan tertentu:

```
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
})
```

Contoh

Unggah Folder ke Amazon S3

Contoh berikut menggunakan `path/filepath` paket untuk mengumpulkan daftar file secara rekursif dan mengunggahnya ke bucket Amazon S3 yang ditentukan. Kunci objek Amazon S3 diawali dengan jalur relatif file.

```
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
```

```
}
```

Pengelola Unduhan

Manajer Amazon S3 Downloader menentukan apakah file dapat dibagi menjadi bagian-bagian yang lebih kecil dan diunduh secara paralel. Anda dapat menyesuaikan jumlah unduhan paralel dan ukuran bagian yang diunduh.

Contoh: Download File

Contoh berikut menggunakan Amazon S3 `Downloader` untuk mengunduh file. Menggunakan `Downloader` mirip dengan s3. GetObjectoperasi.

```
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"),
})
```

`downloadFile`Parameternya adalah `io.WriterAt` tipe. `WriterAt`Antarmuka memungkinkan `Downloader` untuk menulis beberapa bagian file secara paralel.

Opsi konfigurasi

Saat membuat `Downloader` instance, Anda dapat menentukan opsi konfigurasi untuk menyesuaikan cara objek diunduh:

- `PartSize`- Menentukan ukuran buffer, dalam byte, dari setiap bagian untuk men-download. Ukuran minimum per bagian adalah 5 MB.
- `Concurrency`- Menentukan jumlah bagian untuk men-download secara paralel.

`Concurrency`Nilai membatasi jumlah unduhan bagian bersamaan yang dapat terjadi untuk `Download` panggilan tertentu. Ini bukan batas konkurensi klien global. Tweak nilai `PartSize` dan `Concurrency` konfigurasi untuk menemukan konfigurasi yang optimal. Misalnya, sistem dengan koneksi bandwidth tinggi dapat menerima bagian yang lebih besar dan lebih banyak unduhan secara paralel.

Misalnya, aplikasi Anda mengonfigurasi `Downloader` dengan file`Concurrency`. 5 Aplikasi Anda kemudian memanggil `Download` dari dua goroutine yang berbeda, hasilnya akan menjadi unduhan bagian `10` bersamaan (2 goroutine* 5). `Concurrency`

> **Warning**
>
> Aplikasi Anda diharapkan membatasi panggilan bersamaan `Download` untuk mencegah kehabisan sumber daya aplikasi.

Untuk informasi selengkapnya tentang `Downloader` dan opsi konfigurasi lainnya, lihat Manager.Downloader di Referensi API. AWS SDK for Go

Mengganti Opsi Pengunduh Untuk Unduhan

Anda dapat mengganti `Downloader` opsi saat memanggil `Download` dengan memberikan satu atau beberapa argumen fungsional ke metode. Penggantian ini adalah modifikasi aman konkurensi dan tidak memengaruhi unggahan yang sedang berlangsung, atau `Download` panggilan berikutnya ke manajer. Misalnya, untuk mengganti `PartSize` konfigurasi untuk permintaan unggahan tertentu:

```
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
})
```

Contoh

Unduh Semua Objek dalam Ember

Contoh berikut menggunakan pagination untuk mengumpulkan daftar objek dari bucket Amazon S3. Kemudian mengunduh setiap objek ke file lokal.

```
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

GetBucketRegionadalah fungsi utilitas untuk menentukan lokasi AWS Wilayah Bucket Amazon S3. `GetBucketRegion`mengambil klien Amazon S3 dan menggunakannya untuk menentukan lokasi Bucket yang diminta dalam AWS Partisi yang terkait dengan Wilayah yang dikonfigurasi klien.

Misalnya, untuk menemukan Wilayah untuk 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)
```

Jika `GetBucketRegion` tidak dapat menyelesaikan lokasi Bucket, fungsi mengembalikan jenis BucketNotFoundkesalahan seperti yang ditunjukkan pada contoh.

Masukan Streaming yang Tidak Dapat Dicari

Untuk operasi API seperti `PutObject` dan`UploadPart`, klien Amazon S3 mengharapkan nilai parameter `Body` input untuk mengimplementasikan antarmuka io.Seeker secara default. `io.Seeker`Antarmuka digunakan oleh klien untuk menentukan panjang nilai yang akan diunggah, dan untuk menghitung hash payload untuk tanda tangan permintaan. Jika nilai parameter `Body` input tidak diterapkan`io.Seeker`, aplikasi Anda akan menerima kesalahan.

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

Anda dapat mengubah perilaku ini dengan memodifikasi metode operasi Middleware menggunakan opsi fungsional. The With APIOptions helper mengembalikan opsi fungsional untuk nol atau lebih mutator middleware. Untuk menonaktifkan klien yang menghitung hash payload dan menggunakan tanda tangan permintaan Unsigned Payload add 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

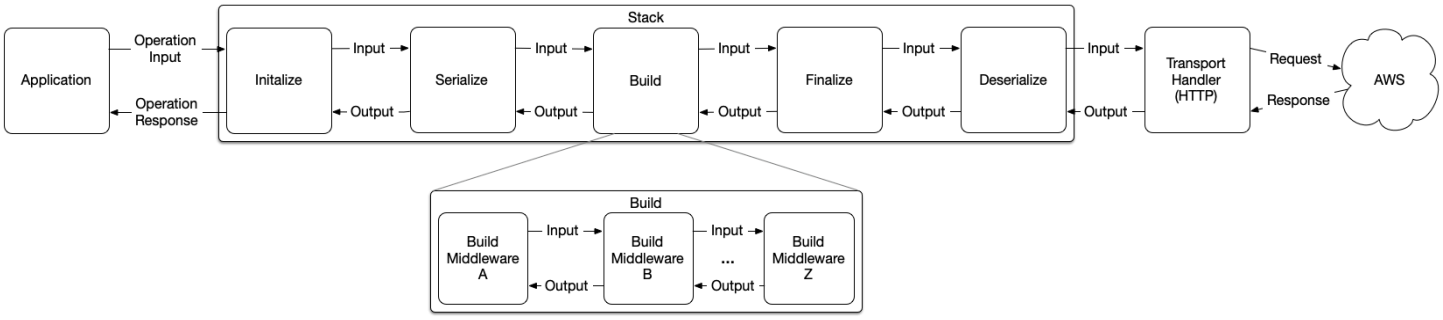
Amazon S3 mengharuskan panjang konten disediakan untuk semua objek yang diunggah ke bucket. Karena parameter `Body` input tidak mengimplementasikan `io.Seeker` antarmuka, klien tidak akan dapat menghitung `ContentLength` parameter untuk permintaan tersebut. Parameter harus disediakan oleh aplikasi. Permintaan akan gagal jika `ContentLength` parameter tidak disediakan.

Gunakan SDK Manajer Unggahan Amazon S3 untuk upload yang tidak dapat dicari dan tidak memiliki panjang yang diketahui.

Menyesuaikan Permintaan Klien AWS SDK for Go v2 dengan Middleware

> **Warning**
>
> Memodifikasi pipeline permintaan klien dapat mengakibatkan permintaan yang salah bentuk/tidak valid, atau dapat mengakibatkan kesalahan aplikasi yang tidak terduga. Fungsionalitas ini dimaksudkan untuk kasus penggunaan lanjutan yang tidak disediakan oleh antarmuka SDK secara default.

Anda dapat menyesuaikan permintaan AWS SDK for Go klien dengan mendaftarkan satu atau lebih middleware ke tumpukan operasi layanan. Tumpukan terdiri dari serangkaian langkah: Initialize, Serialize, Build, Finalize, dan Deserialize. Setiap langkah berisi nol atau lebih middleware yang beroperasi pada jenis input dan output langkah itu. Diagram dan tabel berikut memberikan gambaran umum tentang bagaimana permintaan dan respons operasi melintasi tumpukan.



Langkah Tumpukan	Deskripsi
Inisialisasi	Mempersiapkan input, dan menetapkan parameter default sesuai kebutuhan.
Serialisasi	Serialisasi input ke format protokol yang cocok untuk lapisan transport target.
Membangun	Lampirkan metadata tambahan ke input serial, seperti HTTP Content-Length.

Langkah Tumpukan	Deskripsi
Selesaikan	Persiapan pesan akhir, termasuk percobaan ulang dan otentikasi (penandatanganan SiGv4).
Deserialisasi	Deserialisasi tanggapan dari format protokol menjadi tipe atau kesalahan terstruktur.

Setiap middleware dalam langkah tertentu harus memiliki pengidentifikasi unik, yang ditentukan oleh metode middleware. `ID` Pengidentifikasi middleware memastikan bahwa hanya satu contoh dari middleware tertentu terdaftar ke langkah, dan memungkinkan middleware langkah lain dimasukkan relatif terhadap itu.

Anda melampirkan middleware langkah dengan menggunakan langkah `Insert` atau `Add` metode. Anda gunakan `Add` untuk melampirkan middleware ke awal langkah dengan menentukan middleware.Before sebagai RelativePosition, dan middleware.After untuk melampirkan ke akhir langkah. Anda gunakan `Insert` untuk melampirkan middleware ke langkah dengan memasukkan middleware relatif terhadap middleware langkah lain.

> **Warning**
>
> Anda harus menggunakan `Add` metode ini untuk memasukkan middleware langkah kustom dengan aman. Menggunakan `Insert` menciptakan ketergantungan antara middleware kustom Anda, dan middleware yang Anda masukkan relatif terhadap. Middleware dalam langkah tumpukan harus dianggap buram untuk menghindari perubahan yang melanggar yang terjadi pada aplikasi Anda.

Menulis Middleware Kustom

Setiap langkah tumpukan memiliki antarmuka yang harus Anda penuhi untuk melampirkan middleware ke langkah tertentu. Anda dapat menggunakan salah satu *Step*`MiddlewareFunc` fungsi yang disediakan untuk memenuhi antarmuka ini dengan cepat. Tabel berikut menguraikan langkah-langkah, antarmuka mereka, dan fungsi pembantu yang dapat digunakan untuk memenuhi antarmuka.

Langkah	Antarmuka	Fungsi Pembantu
Inisialisasi	InitializeMiddleware	InitializeMiddlewareFunc
Membangun	BuildMiddleware	BuildMiddlewareFunc
Serialisasi	SerializeMiddleware	SerializeMiddlewareFunc
Selesaikan	FinalizeMiddleware	FinalizeMiddlewareFunc
Deserialisasi	DeserializeMiddleware	DeserializeMiddlewareFunc

Contoh berikut menunjukkan bagaimana Anda dapat menulis middleware khusus untuk mengisi anggota Bucket dari panggilan `GetObject` API Amazon S3 jika tidak disediakan. Middleware ini akan direferensikan dalam contoh lanjutan untuk menunjukkan cara melampirkan middleware langkah ke tumpukan.

```
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)
})
```

Melampirkan Middleware ke Semua Klien

Anda dapat melampirkan middleware langkah kustom Anda ke setiap klien dengan menambahkan middleware menggunakan `APIOptions` anggota tipe AWS.config. Contoh berikut melampirkan `defaultBucket` middleware ke setiap klien yang dibangun menggunakan objek aplikasi Anda: `aws.Config`

```
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)
```

Melampirkan Middleware ke Operasi Tertentu

Anda dapat melampirkan middleware langkah kustom Anda ke operasi klien tertentu dengan memodifikasi `APIOptions` anggota klien menggunakan daftar argumen variadik untuk operasi. Contoh berikut melampirkan `defaultBucket` middleware ke pemanggilan operasi Amazon S3 tertentu: `GetObject`

```
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)
})
```

Melewati Metadata ke Tumpukan

Dalam situasi tertentu, Anda mungkin menemukan bahwa Anda memerlukan dua atau lebih middleware untuk berfungsi bersama-sama dengan berbagi informasi atau status. Anda dapat menggunakan context.Context untuk meneruskan metadata ini dengan menggunakan middleware. WithStackValue. `middleware.WithStackValue`melampirkan pasangan kunci-nilai yang diberikan ke konteks yang disediakan, dan dengan aman membatasi ruang lingkup ke tumpukan yang sedang dijalankan. Nilai stack-scoped ini dapat diambil dari konteks menggunakan middleware. GetStackValuedan memberikan kunci yang digunakan untuk menyimpan nilai yang sesuai. Kunci harus sebanding, dan Anda harus mendefinisikan tipe Anda sendiri sebagai kunci konteks untuk menghindari tabrakan. Contoh berikut menunjukkan bagaimana dua middleware dapat digunakan `context.Context` untuk meneruskan informasi ke tumpukan.

```
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 customInitalize = 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)
})
```

Metadata Disediakan oleh SDK

AWS SDK for Go Menyediakan beberapa nilai metadata yang dapat diambil dari konteks yang disediakan. Nilai-nilai ini dapat digunakan untuk mengaktifkan middleware yang lebih dinamis yang memodifikasi perilakunya berdasarkan layanan eksekusi, operasi, atau wilayah target. Beberapa kunci yang tersedia disediakan dalam tabel di bawah ini:

Kunci	Retriever	Deskripsi
ServiceID	GetServiceID	Ambil pengenal layanan untuk tumpukan eksekusi. Ini dapat dibandingkan dengan `ServiceID` konstanta paket klien layanan.
OperationName	GetOperationName	Ambil nama operasi untuk tumpukan eksekusi.
Pencatat	GetLogger	Ambil logger yang dapat digunakan untuk logging pesan dari middleware.

Melewati Metadata Ke Tumpukan

Anda dapat meneruskan metadata melalui tumpukan dengan menambahkan kunci metadata dan pasangan nilai menggunakan middleware.metadata. Setiap langkah middleware mengembalikan struktur output, metadata, dan kesalahan. Middleware kustom Anda harus mengembalikan metadata yang diterima dari memanggil handler berikutnya di langkah tersebut. Ini memastikan bahwa metadata yang ditambahkan oleh middleware hilir menyebar ke aplikasi yang menjalankan operasi layanan. Metadata yang dihasilkan dapat diakses oleh aplikasi pemanggilan baik dengan bentuk output operasi melalui anggota struktur. `ResultMetadata`

Contoh berikut menunjukkan bagaimana middleware kustom dapat menambahkan metadata yang dikembalikan sebagai bagian dari output operasi.

```
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)
```

Pertanyaan yang Sering Diajukan

Bagaimana cara mengkonfigurasi klien HTTP SDK saya? Apakah ada pedoman atau praktik terbaik?

Kami tidak dapat memberikan panduan kepada pelanggan tentang cara mengkonfigurasi alur kerja HTTP mereka dengan cara yang paling efektif untuk beban kerja khusus mereka. Jawaban untuk ini adalah produk dari persamaan multivariat, dengan faktor input termasuk tetapi tidak terbatas pada:

- jejak jaringan aplikasi (TPS, throughput, dll.)
- Layanan yang digunakan
- karakteristik komputasi dari penyebaran
- sifat geografis dari penyebaran
- perilaku aplikasi yang diinginkan atau kebutuhan aplikasi itu sendiri (SLAs, timing, dll.)

Bagaimana saya harus mengonfigurasi batas waktu operasi?

Sama seperti pertanyaan sebelumnya, itu tergantung. Elemen yang perlu dipertimbangkan di sini meliputi:

- Semua faktor di atas mengenai konfigurasi klien HTTP
- Waktu aplikasi Anda sendiri atau kendala SLA (misalnya jika Anda sendiri melayani lalu lintas ke konsumen lain)

Jawaban atas pertanyaan ini hampir TIDAK PERNAH didasarkan pada pengamatan empiris murni dari perilaku hulu - misalnya “Saya membuat 1000 panggilan ke operasi ini, butuh paling banyak 5 detik jadi saya akan mengatur batas waktu berdasarkan itu dengan faktor keamanan 2x hingga 10 detik”. Kondisi lingkungan dapat berubah, layanan dapat menurun sementara, dan jenis asumsi ini dapat menjadi salah tanpa peringatan.

Permintaan yang dibuat oleh SDK habis waktu atau terlalu lama, bagaimana cara memperbaikinya?

Kami tidak dapat membantu panggilan operasi yang diperpanjang atau habis waktu karena waktu yang lama dihabiskan untuk kabel. “Wire time” di SDK didefinisikan sebagai salah satu dari berikut ini:

- Waktu yang dihabiskan dalam metode klien SDK `HTTPClient.Do()`
- Waktu yang dihabiskan di `Read()` s pada badan respons HTTP yang telah diteruskan ke pemanggil (mis.) `GetObject`

Jika Anda mengalami masalah karena latensi operasi atau batas waktu, tindakan pertama Anda harus mendapatkan telemetri siklus hidup operasi SDK untuk menentukan gangguan waktu antara waktu yang dihabiskan pada kabel dan overhead operasi di sekitarnya. Lihat panduan tentang pengaturan waktu operasi SDK, yang berisi cuplikan kode yang dapat digunakan kembali yang dapat mencapai hal ini.

Bagaimana cara memperbaiki **`read: connection reset`** kesalahan?

SDK mencoba ulang kesalahan apa pun yang cocok dengan `connection reset` pola secara default. Ini akan mencakup penanganan kesalahan untuk sebagian besar operasi, di mana respons HTTP operasi sepenuhnya dikonsumsi dan dideserialisasi ke dalam tipe hasil modelnya.

Namun, kesalahan ini masih dapat terjadi dalam konteks di luar loop coba lagi: operasi layanan tertentu secara langsung meneruskan badan respons HTTP API ke pemanggil untuk dikonsumsi dari kabel secara langsung melalui `io.ReadCloser` (misalnya `GetObject` muatan objek). Anda mungkin mengalami kesalahan ini saat melakukan a `Read` pada badan respons.

Kesalahan ini menunjukkan bahwa host Anda, layanan, atau pihak perantara mana pun (misalnya gateway NAT, proxy, penyeimbang beban) menutup koneksi saat mencoba membaca respons.

Ini dapat terjadi karena beberapa alasan:

- Anda tidak mengkonsumsi badan respons untuk beberapa waktu setelah respons itu sendiri diterima (setelah operasi layanan dipanggil). Kami menyarankan Anda menggunakan badan respons HTTP sesegera mungkin untuk jenis operasi ini.

- Anda tidak menutup badan respons yang diterima sebelumnya. Ini dapat menyebabkan koneksi disetel ulang pada platform tertentu. Anda HARUS menutup **`io.ReadCloser`** instans apa pun yang disediakan dalam respons operasi, terlepas dari apakah Anda mengkonsumsi isinya.

Selain itu, coba jalankan `tcpdump` untuk koneksi yang terpengaruh di tepi jaringan Anda (misalnya setelah proxy apa pun yang Anda kontrol). Jika Anda melihat bahwa AWS titik akhir tampaknya mengirim RST TCP, Anda harus menggunakan konsol AWS dukungan untuk membuka kasus terhadap layanan yang melanggar. Bersiaplah untuk memberikan permintaan IDs dan stempel waktu tertentu saat masalah terjadi.

Mengapa saya mendapatkan kesalahan “tanda tangan tidak valid” saat menggunakan proxy HTTP dengan SDK?

Algoritma tanda tangan untuk AWS layanan (umumnya sigv4) terkait dengan header permintaan serial, lebih khusus lagi sebagian besar header diawali dengan. `X-` Proxy cenderung memodifikasi permintaan keluar dengan menambahkan informasi penerusan tambahan (seringkali melalui `X-Forwarded-For` header) yang secara efektif merusak tanda tangan yang dihitung SDK.

Jika Anda menggunakan proxy HTTP dan mengalami kesalahan tanda tangan, Anda harus bekerja untuk menangkap permintaan karena muncul keluar dari proxy dan menentukan apakah itu berbeda.

Pengaturan waktu operasi SDK

Saat men-debug masalah batas waktu/latensi di SDK, penting untuk mengidentifikasi komponen siklus hidup operasi yang membutuhkan lebih banyak waktu untuk dijalankan daripada yang diharapkan. Sebagai titik awal, Anda biasanya perlu memeriksa rincian waktu antara panggilan operasi keseluruhan dan panggilan HTTP itu sendiri.

Contoh program berikut mengimplementasikan probe instrumentasi dasar dalam hal `smithy-go` middleware untuk klien SQS dan menunjukkan bagaimana hal itu digunakan. Probe memancarkan informasi berikut untuk setiap panggilan operasi:

- AWS ID permintaan
- ID layanan
- nama operasi
- waktu pemanggilan operasi
- waktu panggilan http

Setiap pesan yang dipancarkan diawali dengan “ID pemanggilan” unik (untuk satu operasi) yang disetel di awal tumpukan handler.

Titik masuk untuk instrumentasi diekspos sebagai`WithOperationTiming`, yang diparameterisasi untuk menerima fungsi penanganan pesan yang akan menerima instrumentasi “peristiwa” sebagai string yang diformat. `PrintfMSGHandler`disediakan sebagai kenyamanan yang hanya akan membuang pesan ke stdout.

Layanan yang digunakan di sini dapat dipertukarkan - SEMUA opsi klien layanan menerima `APIOptions` dan konfigurasi `HTTPClient` sebagai. Misalnya, `WithOperationTiming` bisa dideklarasikan sebagai:

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

Jika Anda mengubahnya, pastikan untuk mengubah tanda tangan fungsi yang dikembalikan juga.

```
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()
}
```

Contoh output dari program ini:

```
[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
```

Unit Testing dengan AWS SDK for Go v2

Saat menggunakan SDK dalam aplikasi Anda, Anda akan ingin mengolok-olok SDK untuk pengujian unit aplikasi Anda. Mengejek SDK memungkinkan pengujian Anda difokuskan pada apa yang ingin Anda uji, bukan internal SDK.

Untuk mendukung ejekan, gunakan antarmuka Go alih-alih klien layanan konkret, paginator, dan jenis pelayan, seperti. `s3.Client` Hal ini memungkinkan aplikasi Anda untuk menggunakan pola seperti injeksi dependensi untuk menguji logika aplikasi Anda.

Mengejek Operasi Klien

Dalam contoh ini, `S3GetObjectAPI` adalah antarmuka yang mendefinisikan kumpulan operasi Amazon S3 API yang diperlukan oleh `GetObjectFromS3` fungsi. `S3GetObjectAPI`puas dengan metode klien Amazon S3. GetObject

```
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)
}
```

Untuk menguji `GetObjectFromS3` fungsi, gunakan `mockGetObjectAPI` untuk memenuhi definisi `S3GetObjectAPI` antarmuka. Kemudian gunakan `mockGetObjectAPI` tipe untuk mengejek output dan respons kesalahan yang dikembalikan dari klien layanan.

```
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)
            }
        })
    }
}
```

Mengejek Paginator

Mirip dengan klien layanan, paginator dapat diejek dengan mendefinisikan antarmuka Go untuk paginator. Antarmuka itu akan digunakan oleh kode aplikasi Anda. Ini memungkinkan implementasi SDK digunakan saat aplikasi Anda berjalan, dan implementasi tiruan untuk pengujian.

Dalam contoh berikut, `ListObjectsV2Pager` adalah antarmuka yang mendefinisikan perilaku untuk Amazon ListObjectsS3 V2Paginator yang diperlukan oleh fungsi. `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
}
```

Untuk mengujiCountObjects, buat mockListObjectsV2Pager tipe untuk memenuhi definisi ListObjectsV2Pager antarmuka. Kemudian gunakan mockListObjectsV2Pager untuk mereplikasi perilaku paging output dan respons kesalahan dari paginator operasi layanan.

```
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)
    }
}
```

Contoh kode SDK for Go V2

Contoh kode dalam topik ini menunjukkan cara menggunakan AWS SDK for Go V2 dengan AWS.

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Beberapa layanan berisi kategori contoh tambahan yang menunjukkan cara memanfaatkan pustaka atau fungsi khusus untuk layanan.

Layanan

- Contoh API Gateway menggunakan SDK for Go V2
- Contoh Aurora menggunakan SDK for Go V2
- Contoh Amazon Bedrock menggunakan SDK for Go V2
- Contoh Amazon Bedrock Runtime menggunakan SDK for Go V2
- CloudFormation contoh menggunakan SDK for Go V2
- CloudWatch Contoh log menggunakan SDK for Go V2
- Contoh Penyedia Identitas Amazon Cognito menggunakan SDK for Go V2
- Contoh Amazon DocumentDB menggunakan SDK for Go V2
- Contoh DynamoDB menggunakan SDK for Go V2
- Contoh IAM menggunakan SDK for Go V2
- Contoh Kinesis menggunakan SDK for Go V2
- Contoh Lambda menggunakan SDK for Go V2
- Contoh MSK Amazon menggunakan SDK for Go V2
- Contoh Partner Central menggunakan SDK for Go V2
- Contoh Amazon RDS menggunakan SDK for Go V2
- Contoh Amazon Redshift menggunakan SDK for Go V2

- Contoh Amazon S3 menggunakan SDK for Go V2
- Contoh Amazon SNS menggunakan SDK for Go V2
- Contoh Amazon SQS menggunakan SDK for Go V2

Contoh API Gateway menggunakan SDK for Go V2

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Go V2 dengan API Gateway.

AWS kontribusi komunitas adalah contoh yang dibuat dan dikelola oleh banyak tim AWS. Untuk memberikan umpan balik, gunakan mekanisme yang disediakan di repositori yang ditautkan.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- AWS kontribusi komunitas

AWS kontribusi komunitas

Membangun dan menguji aplikasi tanpa server

Contoh kode berikut menunjukkan cara membangun dan menguji aplikasi tanpa server menggunakan API Gateway dengan Lambda dan DynamoDB

SDK untuk Go V2

> Menunjukkan cara membuat dan menguji aplikasi tanpa server yang terdiri dari API Gateway dengan Lambda dan DynamoDB menggunakan Go SDK.
>
> Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di GitHub.
>
> Layanan yang digunakan dalam contoh ini
>
> - API Gateway
> - DynamoDB
> - Lambda

Contoh Aurora menggunakan SDK for Go V2

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Go V2 dengan Aurora.

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Memulai
- Hal-hal mendasar
- Tindakan

Memulai

Halo Aurora

Contoh kode berikut menunjukkan bagaimana memulai menggunakan Aurora.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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)
  }
 }
}
```

- Untuk detail API, lihat Menjelaskan DBClusters di Referensi AWS SDK for Go API.

Hal-hal mendasar

Pelajari dasar-dasarnya

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Membuat grup parameter klaster DB Aurora dan mengatur nilai parameter.
- Membuat klaster DB yang menggunakan grup parameter.
- Membuat instans DB yang berisi basis data.
- Mengambil snapshot klaster DB, lalu membersihkan sumber daya.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkap dan pelajari cara menyiapkan dan menjalankan di Repositori Contoh Kode AWS.

Jalankan skenario interaktif di prompt perintah.

```
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()
}
```

Tentukan fungsi-fungsi yang dipanggil oleh skenario untuk mengelola tindakan 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 paramaeters 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
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Go .
  - Buat DBCluster
  - Buat DBCluster ParameterGroup
  - Buat DBCluster Snapshot
  - Buat DBInstance
  - Hapus DBCluster
  - Hapus DBCluster ParameterGroup
  - Hapus DBInstance
  - Jelaskan DBCluster ParameterGroups
  - Jelaskan DBCluster Parameter
  - Jelaskan DBCluster Snapshots
  - Jelaskan DBClusters
  - Jelaskan DBEngine Versi
  - Jelaskan DBInstances
  - DescribeOrderableDBInstancePilihan
  - Memodifikasi DBCluster ParameterGroup

Tindakan

`CreateDBCluster`

Contoh kode berikut menunjukkan cara menggunakan`CreateDBCluster`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
 }
}
```

- Untuk detail API, lihat Membuat DBCluster di Referensi AWS SDK for Go API.

`CreateDBClusterParameterGroup`

Contoh kode berikut menunjukkan cara menggunakan`CreateDBClusterParameterGroup`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
 }
}
```

- Untuk detail API, lihat Membuat DBCluster ParameterGroup di Referensi AWS SDK for Go API.

`CreateDBClusterSnapshot`

Contoh kode berikut menunjukkan cara menggunakan`CreateDBClusterSnapshot`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
 }
}
```

- Untuk detail API, lihat Membuat DBCluster Snapshot di Referensi AWS SDK for Go API.

`CreateDBInstance`

Contoh kode berikut menunjukkan cara menggunakan`CreateDBInstance`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
 }
}
```

- Untuk detail API, lihat Membuat DBInstance di Referensi AWS SDK for Go API.

`DeleteDBCluster`

Contoh kode berikut menunjukkan cara menggunakan`DeleteDBCluster`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
 }
}
```

- Untuk detail API, lihat Menghapus DBCluster di Referensi AWS SDK for Go API.

`DeleteDBClusterParameterGroup`

Contoh kode berikut menunjukkan cara menggunakan`DeleteDBClusterParameterGroup`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
 }
}
```

- Untuk detail API, lihat Menghapus DBCluster ParameterGroup di Referensi AWS SDK for Go API.

`DeleteDBInstance`

Contoh kode berikut menunjukkan cara menggunakan`DeleteDBInstance`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
 }
}
```

- Untuk detail API, lihat Menghapus DBInstance di Referensi AWS SDK for Go API.

`DescribeDBClusterParameterGroups`

Contoh kode berikut menunjukkan cara menggunakan`DescribeDBClusterParameterGroups`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
 }
}
```

- Untuk detail API, lihat Menjelaskan DBCluster ParameterGroups di Referensi AWS SDK for Go API.

`DescribeDBClusterParameters`

Contoh kode berikut menunjukkan cara menggunakan`DescribeDBClusterParameters`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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 paramaeters for %v: %v\n", parameterGroupName, err)
   break
  } else {
   params = append(params, output.Parameters...)
  }
 }
 return params, err
}
```

- Untuk detail API, lihat Menjelaskan DBCluster Parameter di Referensi AWS SDK for Go API.

`DescribeDBClusterSnapshots`

Contoh kode berikut menunjukkan cara menggunakan`DescribeDBClusterSnapshots`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
 }
}
```

- Untuk detail API, lihat Menjelaskan DBCluster Snapshot di Referensi AWS SDK for Go API.

`DescribeDBClusters`

Contoh kode berikut menunjukkan cara menggunakan`DescribeDBClusters`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
 }
}
```

- Untuk detail API, lihat Menjelaskan DBClusters di Referensi AWS SDK for Go API.

`DescribeDBEngineVersions`

Contoh kode berikut menunjukkan cara menggunakan`DescribeDBEngineVersions`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
 }
```

```
}
```

- Untuk detail API, lihat Menjelaskan DBEngine Versi dalam Referensi AWS SDK for Go API.

`DescribeDBInstances`

Contoh kode berikut menunjukkan cara menggunakan`DescribeDBInstances`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
 }
}
```

- Untuk detail API, lihat Menjelaskan DBInstances di Referensi AWS SDK for Go API.

`DescribeOrderableDBInstanceOptions`

Contoh kode berikut menunjukkan cara menggunakan`DescribeOrderableDBInstanceOptions`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat DescribeOrderableDBInstanceOpsi di Referensi AWS SDK for Go API.

`ModifyDBClusterParameterGroup`

Contoh kode berikut menunjukkan cara menggunakan`ModifyDBClusterParameterGroup`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
 }
}
```

- Untuk detail API, lihat Memodifikasi DBCluster ParameterGroup dalam Referensi AWS SDK for Go API.

Contoh Amazon Bedrock menggunakan SDK for Go V2

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan AWS SDK for Go V2 dengan Amazon Bedrock.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Memulai
- Tindakan

Memulai

Halo Amazon Bedrock

Contoh kode berikut menunjukkan cara memulai menggunakan Amazon Bedrock.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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)
 }
}
```

- Untuk detail API, lihat ListFoundationModelsdi Referensi AWS SDK for Go API.

Tindakan

`ListFoundationModels`

Contoh kode berikut menunjukkan cara menggunakan`ListFoundationModels`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Buat daftar model foundation Bedrock yang tersedia.

```
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
}
```

- Untuk detail API, lihat ListFoundationModelsdi Referensi AWS SDK for Go API.

Contoh Amazon Bedrock Runtime menggunakan SDK for Go V2

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Go V2 dengan Amazon Bedrock Runtime.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Memulai
- Skenario
- Generator Gambar Amazon Titan
- Antropik Claude

Memulai

Halo Amazon Bedrock

Contoh kode berikut menunjukkan cara memulai menggunakan Amazon Bedrock.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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)
}
```

- Untuk detail API, lihat InvokeModeldi Referensi AWS SDK for Go API.

Skenario

Gunakan beberapa model fondasi di Amazon Bedrock

Contoh kode berikut menunjukkan cara menyiapkan dan mengirim prompt ke berbagai model berbahasa besar (LLMs) di Amazon Bedrock

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Gunakan beberapa model fondasi di 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)
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Go .
  - InvokeModel
  - InvokeModelWithResponseStream

Generator Gambar Amazon Titan

InvokeModel

Contoh kode berikut menunjukkan cara memanggil Amazon Titan Image di Amazon Bedrock untuk menghasilkan gambar.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Buat gambar dengan 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

}
```

- Untuk detail API, lihat InvokeModeldi Referensi AWS SDK for Go API.

Antropik Claude

Bercakap-cakap

Contoh kode berikut menunjukkan cara mengirim pesan teks ke Anthropic Claude, menggunakan API Converse Bedrock.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Kirim pesan teks ke Anthropic Claude, menggunakan API Converse 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

}
```

- Untuk detail API, lihat Converse di Referensi AWS SDK for Go API.

InvokeModel

Contoh kode berikut menunjukkan cara mengirim pesan teks ke Anthropic Claude, menggunakan Invoke Model API.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Gunakan model dasar Anthropic Claude 2 untuk menghasilkan teks.

```
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
}
```

- Untuk detail API, lihat InvokeModeldi Referensi AWS SDK for Go API.

InvokeModelWithResponseStream

Contoh kode berikut menunjukkan cara mengirim pesan teks ke model Anthropic Claude, menggunakan Invoke Model API, dan mencetak aliran respons.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Gunakan API Invoke Model untuk mengirim pesan teks dan memproses aliran respons secara real-time.

```
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
\". 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
}
```

- Untuk detail API, lihat InvokeModelWithResponseStreamdi Referensi AWS SDK for Go API.

CloudFormation contoh menggunakan SDK for Go V2

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Go V2 dengan CloudFormation.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Tindakan

Tindakan

`DescribeStacks`

Contoh kode berikut menunjukkan cara menggunakan`DescribeStacks`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat DescribeStacksdi Referensi AWS SDK for Go API.

CloudWatch Contoh log menggunakan SDK for Go V2

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Go V2 dengan CloudWatch Log.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Tindakan

Tindakan

`StartLiveTail`

Contoh kode berikut menunjukkan cara menggunakan`StartLiveTail`.

SDK untuk Go V2

Sertakan file-file yang diperlukan.

```
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"
)
```

Tangani acara dari sesi 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 occured during streaming: %v", err)
   } else if event == nil {
    log.Println("Stream is Closed")
    return
   } else {
    log.Fatalf("Unknown event type: %T", e)
   }
  }
 }
}
```

Mulai sesi 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)
```

Hentikan sesi Live Tail setelah periode waktu berlalu.

```
// Close the stream (which ends the session) after a timeout
time.Sleep(10 * time.Second)
```

```
    stream.Close()
    log.Println("Event stream closed")
```

- Untuk detail API, lihat StartLiveTaildi Referensi AWS SDK for Go API.

Contoh Penyedia Identitas Amazon Cognito menggunakan SDK for Go V2

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan AWS SDK for Go V2 dengan Penyedia Identitas Amazon Cognito.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Memulai
- Tindakan
- Skenario

Memulai

Halo Amazon Cognito

Contoh kode berikut menunjukkan cara memulai menggunakan Amazon Cognito.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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)
  }
 }
}
```

- Untuk detail API, lihat ListUserPoolsdi Referensi AWS SDK for Go API.

Tindakan

`AdminCreateUser`

Contoh kode berikut menunjukkan cara menggunakan`AdminCreateUser`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat AdminCreateUserdi Referensi AWS SDK for Go API.

`AdminSetUserPassword`

Contoh kode berikut menunjukkan cara menggunakan`AdminSetUserPassword`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat AdminSetUserPassworddi Referensi AWS SDK for Go API.

`ConfirmForgotPassword`

Contoh kode berikut menunjukkan cara menggunakan`ConfirmForgotPassword`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat ConfirmForgotPassworddi Referensi AWS SDK for Go API.

`DeleteUser`

Contoh kode berikut menunjukkan cara menggunakan`DeleteUser`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat DeleteUserdi Referensi AWS SDK for Go API.

`ForgotPassword`

Contoh kode berikut menunjukkan cara menggunakan`ForgotPassword`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat ForgotPassworddi Referensi AWS SDK for Go API.

`InitiateAuth`

Contoh kode berikut menunjukkan cara menggunakan`InitiateAuth`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat InitiateAuthdi Referensi AWS SDK for Go API.

`ListUserPools`

Contoh kode berikut menunjukkan cara menggunakan`ListUserPools`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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)
  }
 }
}
```

- Untuk detail API, lihat ListUserPoolsdi Referensi AWS SDK for Go API.

`SignUp`

Contoh kode berikut menunjukkan cara menggunakan`SignUp`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat SignUpdi Referensi AWS SDK for Go API.

`UpdateUserPool`

Contoh kode berikut menunjukkan cara menggunakan`UpdateUserPool`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat UpdateUserPooldi Referensi AWS SDK for Go API.

Skenario

Secara otomatis mengonfirmasi pengguna yang dikenal dengan fungsi Lambda

Contoh kode berikut menunjukkan cara mengonfirmasi pengguna Amazon Cognito yang diketahui secara otomatis dengan fungsi Lambda.

- Konfigurasikan kumpulan pengguna untuk memanggil fungsi Lambda untuk `PreSignUp` pemicunya.
- Daftarkan pengguna dengan Amazon Cognito.
- Fungsi Lambda memindai tabel DynamoDB dan secara otomatis mengonfirmasi pengguna yang dikenal.
- Masuk sebagai pengguna baru, lalu bersihkan sumber daya.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkap dan pelajari cara menyiapkan dan menjalankan di Repositori Contoh Kode AWS.

Jalankan skenario interaktif di penggugah/prompt perintah.

```
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))
}
```

Tangani `PreSignUp` pelatuk dengan fungsi 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)
}
```

Buat struct yang melakukan tugas-tugas umum.

```
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))
}
```

Buat struct yang membungkus tindakan 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
}
```

Buat struct yang membungkus tindakan 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
}
```

Buat struct yang membungkus tindakan CloudWatch Log.

```
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
}
```

Buat struct yang membungkus tindakan 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
}
```

Pembersihan sumber daya

```
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!")
 }
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Go .
  - DeleteUser
  - InitiateAuth
  - SignUp
  - UpdateUserPool

Secara otomatis memigrasikan pengguna yang dikenal dengan fungsi Lambda

Contoh kode berikut menunjukkan cara memigrasi pengguna Amazon Cognito yang dikenal secara otomatis dengan fungsi Lambda.

- Konfigurasikan kumpulan pengguna untuk memanggil fungsi Lambda untuk `MigrateUser` pemicunya.
- Masuk ke Amazon Cognito dengan nama pengguna dan email yang tidak ada di kumpulan pengguna.
- Fungsi Lambda memindai tabel DynamoDB dan secara otomatis memigrasikan pengguna yang dikenal ke kumpulan pengguna.
- Lakukan alur lupa kata sandi untuk mengatur ulang kata sandi untuk pengguna yang dimigrasi.
- Masuk sebagai pengguna baru, lalu bersihkan sumber daya.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkap dan pelajari cara menyiapkan dan menjalankan di Repositori Contoh Kode AWS.

Jalankan skenario interaktif di penggugah/prompt perintah.

```
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))
}
```

Tangani `MigrateUser` pelatuk dengan fungsi 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)
}
```

Buat struct yang melakukan tugas-tugas umum.

```
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))
}
```

Buat struct yang membungkus tindakan 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
}
```

Buat struct yang membungkus tindakan 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
}
```

Buat struct yang membungkus tindakan CloudWatch Log.

```
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
}
```

Buat struct yang membungkus tindakan 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
}
```

Pembersihan sumber daya

```
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!")
 }
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Go .
  - ConfirmForgotPassword
  - DeleteUser
  - ForgotPassword
  - InitiateAuth
  - SignUp
  - UpdateUserPool

Menulis data aktivitas khusus dengan fungsi Lambda setelah otentikasi pengguna Amazon Cognito

Contoh kode berikut menunjukkan cara menulis data aktivitas kustom dengan fungsi Lambda setelah autentikasi pengguna Amazon Cognito.

- Gunakan fungsi administrator untuk menambahkan pengguna ke kumpulan pengguna.
- Konfigurasikan kumpulan pengguna untuk memanggil fungsi Lambda untuk `PostAuthentication` pemicunya.
- Masuk pengguna baru ke Amazon Cognito.
- Fungsi Lambda menulis informasi kustom ke CloudWatch Log dan ke tabel DynamoDB.
- Dapatkan dan tampilkan data kustom dari tabel DynamoDB, lalu bersihkan sumber daya.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkap dan pelajari cara menyiapkan dan menjalankan di Repositori Contoh Kode AWS.

Jalankan skenario interaktif di penggugah/prompt perintah.

```
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))
}
```

Tangani `PostAuthentication` pelatuk dengan fungsi 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)
}
```

Buat struct yang melakukan tugas-tugas umum.

```
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))
}
```

Buat struct yang membungkus tindakan 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
}
```

Buat struct yang membungkus tindakan 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
}
```

Buat struct yang membungkus tindakan CloudWatch Log.

```
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
}
```

Buat struct yang membungkus tindakan 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
}
```

Pembersihan sumber daya

```
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!")
 }
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Go .

- AdminCreateUser
- AdminSetUserPassword
- DeleteUser
- InitiateAuth
- UpdateUserPool

Contoh Amazon DocumentDB menggunakan SDK for Go V2

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Go V2 dengan Amazon DocumentDB.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Contoh nirserver

Contoh nirserver

Memanggil fungsi Lambda dari pemicu Amazon DocumentDB

Contoh kode berikut menunjukkan cara menerapkan fungsi Lambda yang menerima peristiwa yang dipicu dengan menerima catatan dari aliran perubahan DocumentDB. Fungsi mengambil payload DocumentDB dan mencatat isi catatan.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Mengkonsumsi acara Amazon DocumentDB dengan Lambda menggunakan 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))
}
```

Contoh DynamoDB menggunakan SDK for Go V2

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Go V2 dengan DynamoDB.

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

AWS kontribusi komunitas adalah contoh yang dibuat dan dikelola oleh banyak tim AWS. Untuk memberikan umpan balik, gunakan mekanisme yang disediakan di repositori yang ditautkan.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Hal-hal mendasar
- Tindakan
- Skenario
- Contoh nirserver
- AWS kontribusi komunitas

Hal-hal mendasar

Pelajari dasar-dasarnya

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Buat tabel yang dapat menyimpan data film.

- Masukkan, dapatkan, dan perbarui satu film dalam tabel tersebut.
- Tulis data film ke tabel dari file JSON sampel.
- Kueri untuk film yang dirilis pada tahun tertentu.
- Pindai film yang dirilis dalam suatu rentang tahun.
- Hapus film dari tabel, lalu hapus tabel tersebut.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Jalankan skenario interaktif untuk membuat tabel dan melakukan tindakan pada tabel tersebut.

```
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))
}
```

Tentukan struct Movie yang digunakan dalam contoh ini.

```
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"])
}
```

Buat struct dan metode yang memanggil tindakan 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
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Go .
  - BatchWriteItem
  - CreateTable
  - DeleteItem
  - DeleteTable
  - DescribeTable
  - GetItem
  - PutItem
  - Kueri
  - Scan
  - UpdateItem

Tindakan

`BatchExecuteStatement`

Contoh kode berikut menunjukkan cara menggunakan`BatchExecuteStatement`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Tentukan struct penerima fungsi untuk contoh.

```
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
}
```

Gunakan batch pernyataan INSERT untuk menambahkan item.

```
// 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
}
```

Gunakan batch pernyataan SELECT untuk mendapatkan item.

```
// 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
}
```

Gunakan batch pernyataan UPDATE untuk memperbarui item.

```
// 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
}
```

Gunakan batch DELETE untuk menghapus item.

```
// 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
}
```

Tentukan struct Movie yang digunakan dalam contoh ini.

```
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"])
}
```

- Untuk detail API, lihat BatchExecuteStatementdi Referensi AWS SDK for Go API.

`BatchWriteItem`

Contoh kode berikut menunjukkan cara menggunakan`BatchWriteItem`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

Tentukan struct Movie yang digunakan dalam contoh ini.

```
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"])
}
```

- Untuk detail API, lihat BatchWriteItemdi Referensi AWS SDK for Go API.

`CreateTable`

Contoh kode berikut menunjukkan cara menggunakan`CreateTable`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat CreateTabledi Referensi AWS SDK for Go API.

`DeleteItem`

Contoh kode berikut menunjukkan cara menggunakan`DeleteItem`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

Tentukan struct Movie yang digunakan dalam contoh ini.

```
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"])
}
```

- Untuk detail API, lihat DeleteItemdi Referensi AWS SDK for Go API.

`DeleteTable`

Contoh kode berikut menunjukkan cara menggunakan`DeleteTable`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat DeleteTabledi Referensi AWS SDK for Go API.

`DescribeTable`

Contoh kode berikut menunjukkan cara menggunakan`DescribeTable`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat DescribeTabledi Referensi AWS SDK for Go API.

`ExecuteStatement`

Contoh kode berikut menunjukkan cara menggunakan`ExecuteStatement`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Tentukan struct penerima fungsi untuk contoh.

```
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
}
```

Gunakan pernyataan INSERT untuk menambahkan item.

```
// 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
}
```

Gunakan pernyataan SELECT untuk mendapatkan item.

```
// 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
}
```

Gunakan pernyataan SELECT untuk mendapatkan daftar item dan memproyeksikan hasilnya.

```
// 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
}
```

Gunakan pernyataan UPDATE untuk memperbarui item.

```
// 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
}
```

Gunakan pernyataan DELETE untuk menghapus sebuah item.

```
// 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
}
```

Tentukan struct Movie yang digunakan dalam contoh ini.

```
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"])
}
```

- Untuk detail API, lihat ExecuteStatementdi Referensi AWS SDK for Go API.

`GetItem`

Contoh kode berikut menunjukkan cara menggunakan`GetItem`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

Tentukan struct Movie yang digunakan dalam contoh ini.

```
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"])
}
```

- Untuk detail API, lihat GetItemdi Referensi AWS SDK for Go API.

`ListTables`

Contoh kode berikut menunjukkan cara menggunakan`ListTables`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat ListTablesdi Referensi AWS SDK for Go API.

`PutItem`

Contoh kode berikut menunjukkan cara menggunakan`PutItem`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

Tentukan struct Movie yang digunakan dalam contoh ini.

```
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"])
}
```

- Untuk detail API, lihat PutItemdi Referensi AWS SDK for Go API.

`Query`

Contoh kode berikut menunjukkan cara menggunakan`Query`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

Tentukan struct Movie yang digunakan dalam contoh ini.

```
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"])
}
```

- Untuk detail API, lihat Kueri di Referensi API AWS SDK for Go .

`Scan`

Contoh kode berikut menunjukkan cara menggunakan`Scan`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

Tentukan struct Movie yang digunakan dalam contoh ini.

```
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"])
}
```

- Untuk detail API, lihat Scan di Referensi API AWS SDK for Go .

`UpdateItem`

Contoh kode berikut menunjukkan cara menggunakan`UpdateItem`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

Tentukan struct Movie yang digunakan dalam contoh ini.

```
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"])
}
```

- Untuk detail API, lihat UpdateItemdi Referensi AWS SDK for Go API.

Skenario

Melakukan kueri pada tabel menggunakan batch pernyataan PartiQL

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Dapatkan batch item dengan menjalankan beberapa pernyataan SELECT.

- Tambahkan batch item dengan menjalankan beberapa pernyataan INSERT.
- Perbarui batch item dengan menjalankan beberapa pernyataan UPDATE.
- Hapus batch item dengan menjalankan beberapa pernyataan DELETE.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Jalankan skenario yang membuat tabel dan menjalankan batch kueri 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))
}
```

Tentukan struct Movie yang digunakan dalam contoh ini.

```
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"])
}
```

Buat struct dan metode yang menjalankan pernyataan 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
}
```

- Untuk detail API, lihat BatchExecuteStatementdi Referensi AWS SDK for Go API.

Melakukan kueri tabel menggunakan PartiQL

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Dapatkan item dengan menjalankan pernyataan SELECT.
- Tambahkan item dengan menjalankan pernyataan INSERT.
- Perbarui item dengan menjalankan pernyataan UPDATE.
- Hapus item dengan menjalankan pernyataan DELETE.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Jalankan skenario yang membuat tabel dan menjalankan kueri 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))
}
```

Tentukan struct Movie yang digunakan dalam contoh ini.

```
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"])
}
```

Buat struct dan metode yang menjalankan pernyataan 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
}
```

- Untuk detail API, lihat ExecuteStatementdi Referensi AWS SDK for Go API.

Contoh nirserver

Memanggil fungsi Lambda dari pemicu DynamoDB

Contoh kode berikut menunjukkan bagaimana menerapkan fungsi Lambda yang menerima peristiwa yang dipicu oleh menerima catatan dari aliran DynamoDB. Fungsi mengambil payload DynamoDB dan mencatat isi catatan.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Mengkonsumsi acara DynamoDB dengan Lambda menggunakan 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)
```

```
}
```

Melaporkan kegagalan item batch untuk fungsi Lambda dengan pemicu DynamoDB

Contoh kode berikut menunjukkan cara mengimplementasikan respons batch sebagian untuk fungsi Lambda yang menerima peristiwa dari aliran DynamoDB. Fungsi melaporkan kegagalan item batch dalam respons, memberi sinyal ke Lambda untuk mencoba lagi pesan tersebut nanti.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Melaporkan kegagalan item batch DynamoDB dengan Lambda menggunakan 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 kontribusi komunitas

Membangun dan menguji aplikasi tanpa server

Contoh kode berikut menunjukkan cara membangun dan menguji aplikasi tanpa server menggunakan API Gateway dengan Lambda dan DynamoDB

SDK untuk Go V2

Menunjukkan cara membuat dan menguji aplikasi tanpa server yang terdiri dari API Gateway dengan Lambda dan DynamoDB menggunakan Go SDK.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di GitHub.

Layanan yang digunakan dalam contoh ini

- API Gateway
- DynamoDB
- Lambda

Contoh IAM menggunakan SDK for Go V2

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Go V2 dengan IAM.

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Memulai
- Hal-hal mendasar
- Tindakan

Memulai

Halo IAM

Contoh kode berikut menunjukkan bagaimana memulai menggunakan IAM.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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%v\n", *policy.PolicyName)
  }
 }
}
```

- Untuk detail API, lihat ListPoliciesdi Referensi AWS SDK for Go API.

Hal-hal mendasar

Pelajari dasar-dasarnya

Contoh kode berikut menunjukkan cara membuat pengguna dan mengambil peran.

> **Warning**
>
> Untuk menghindari risiko keamanan, jangan gunakan pengguna IAM untuk otentikasi saat mengembangkan perangkat lunak yang dibuat khusus atau bekerja dengan data nyata. Sebaliknya, gunakan federasi dengan penyedia identitas seperti AWS IAM Identity Center.

- Buat pengguna tanpa izin.
- Buat peran yang memberikan izin untuk mencantumkan bucket Amazon S3 untuk akun tersebut.
- Tambahkan kebijakan agar pengguna dapat mengambil peran tersebut.
- Asumsikan peran dan daftar bucket S3 menggunakan kredenal sementara, lalu bersihkan sumber daya.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkap dan pelajari cara menyiapkan dan menjalankan di Repositori Contoh Kode AWS.

Jalankan skenario interaktif di penggugah/prompt perintah.

```
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)
}
```

Tentukan struct yang membungkus tindakan akun.

```
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
}
```

Tentukan struct yang membungkus tindakan kebijakan.

```
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
}
```

Tentukan struct yang membungkus tindakan peran.

```
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
}
```

Tentukan struct yang membungkus tindakan pengguna.

```
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
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Go .
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Tindakan

`AttachRolePolicy`

Contoh kode berikut menunjukkan cara menggunakan`AttachRolePolicy`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat AttachRolePolicydi Referensi AWS SDK for Go API.

`CreateAccessKey`

Contoh kode berikut menunjukkan cara menggunakan`CreateAccessKey`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat CreateAccessKeydi Referensi AWS SDK for Go API.

`CreatePolicy`

Contoh kode berikut menunjukkan cara menggunakan`CreatePolicy`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat CreatePolicydi Referensi AWS SDK for Go API.

`CreateRole`

Contoh kode berikut menunjukkan cara menggunakan`CreateRole`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat CreateRoledi Referensi AWS SDK for Go API.

`CreateServiceLinkedRole`

Contoh kode berikut menunjukkan cara menggunakan`CreateServiceLinkedRole`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat CreateServiceLinkedRoledi Referensi AWS SDK for Go API.

`CreateUser`

Contoh kode berikut menunjukkan cara menggunakan`CreateUser`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat CreateUserdi Referensi AWS SDK for Go API.

`DeleteAccessKey`

Contoh kode berikut menunjukkan cara menggunakan`DeleteAccessKey`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat DeleteAccessKeydi Referensi AWS SDK for Go API.

`DeletePolicy`

Contoh kode berikut menunjukkan cara menggunakan`DeletePolicy`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat DeletePolicydi Referensi AWS SDK for Go API.

`DeleteRole`

Contoh kode berikut menunjukkan cara menggunakan`DeleteRole`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat DeleteRoledi Referensi AWS SDK for Go API.

`DeleteServiceLinkedRole`

Contoh kode berikut menunjukkan cara menggunakan`DeleteServiceLinkedRole`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat DeleteServiceLinkedRoledi Referensi AWS SDK for Go API.

`DeleteUser`

Contoh kode berikut menunjukkan cara menggunakan`DeleteUser`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat DeleteUserdi Referensi AWS SDK for Go API.

`DeleteUserPolicy`

Contoh kode berikut menunjukkan cara menggunakan`DeleteUserPolicy`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat DeleteUserPolicydi Referensi AWS SDK for Go API.

`DetachRolePolicy`

Contoh kode berikut menunjukkan cara menggunakan`DetachRolePolicy`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat DetachRolePolicydi Referensi AWS SDK for Go API.

`GetAccountPasswordPolicy`

Contoh kode berikut menunjukkan cara menggunakan`GetAccountPasswordPolicy`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat GetAccountPasswordPolicydi Referensi AWS SDK for Go API.

`GetPolicy`

Contoh kode berikut menunjukkan cara menggunakan`GetPolicy`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
```

```
}
```

- Untuk detail API, lihat GetPolicydi Referensi AWS SDK for Go API.

`GetRole`

Contoh kode berikut menunjukkan cara menggunakan`GetRole`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat GetRoledi Referensi AWS SDK for Go API.

`GetUser`

Contoh kode berikut menunjukkan cara menggunakan`GetUser`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat GetUserdi Referensi AWS SDK for Go API.

`ListAccessKeys`

Contoh kode berikut menunjukkan cara menggunakan`ListAccessKeys`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat ListAccessKeysdi Referensi AWS SDK for Go API.

`ListAttachedRolePolicies`

Contoh kode berikut menunjukkan cara menggunakan`ListAttachedRolePolicies`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat ListAttachedRolePoliciesdi Referensi AWS SDK for Go API.

`ListGroups`

Contoh kode berikut menunjukkan cara menggunakan`ListGroups`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat ListGroupsdi Referensi AWS SDK for Go API.

`ListPolicies`

Contoh kode berikut menunjukkan cara menggunakan`ListPolicies`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat ListPoliciesdi Referensi AWS SDK for Go API.

`ListRolePolicies`

Contoh kode berikut menunjukkan cara menggunakan`ListRolePolicies`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat ListRolePoliciesdi Referensi AWS SDK for Go API.

`ListRoles`

Contoh kode berikut menunjukkan cara menggunakan`ListRoles`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat ListRolesdi Referensi AWS SDK for Go API.

`ListSAMLProviders`

Contoh kode berikut menunjukkan cara menggunakan`ListSAMLProviders`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat Daftar SAMLProviders di Referensi AWS SDK for Go API.

`ListUserPolicies`

Contoh kode berikut menunjukkan cara menggunakan`ListUserPolicies`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat ListUserPoliciesdi Referensi AWS SDK for Go API.

`ListUsers`

Contoh kode berikut menunjukkan cara menggunakan`ListUsers`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat ListUsersdi Referensi AWS SDK for Go API.

`PutUserPolicy`

Contoh kode berikut menunjukkan cara menggunakan`PutUserPolicy`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat PutUserPolicydi Referensi AWS SDK for Go API.

Contoh Kinesis menggunakan SDK for Go V2

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Go V2 dengan Kinesis.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Contoh nirserver

Contoh nirserver

Memanggil fungsi Lambda dari pemicu Kinesis

Contoh kode berikut menunjukkan bagaimana menerapkan fungsi Lambda yang menerima peristiwa yang dipicu oleh menerima catatan dari aliran Kinesis. Fungsi mengambil payload Kinesis, mendekode dari Base64, dan mencatat konten rekaman.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Mengkonsumsi acara Kinesis dengan Lambda menggunakan 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)
}
```

Melaporkan kegagalan item batch untuk fungsi Lambda dengan pemicu Kinesis

Contoh kode berikut menunjukkan cara mengimplementasikan respons batch sebagian untuk fungsi Lambda yang menerima peristiwa dari aliran Kinesis. Fungsi melaporkan kegagalan item batch dalam respons, memberi sinyal ke Lambda untuk mencoba lagi pesan tersebut nanti.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Melaporkan kegagalan item batch Kinesis dengan Lambda menggunakan 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)
}
```

Contoh Lambda menggunakan SDK for Go V2

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Go V2 dengan Lambda.

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

AWS kontribusi komunitas adalah contoh yang dibuat dan dikelola oleh banyak tim AWS. Untuk memberikan umpan balik, gunakan mekanisme yang disediakan di repositori yang ditautkan.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Memulai
- Hal-hal mendasar
- Tindakan
- Skenario
- Contoh nirserver
- AWS kontribusi komunitas

Memulai

Halo Lambda

Contoh kode berikut menunjukkan cara memulai menggunakan Lambda.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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%v\n", *function.FunctionName)
  }
 }
}
```

- Untuk detail API, lihat ListFunctionsdi Referensi AWS SDK for Go API.

Hal-hal mendasar

Pelajari dasar-dasarnya

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Buat peran IAM dan fungsi Lambda, lalu unggah kode handler.
- Panggil fungsi dengan satu parameter dan dapatkan hasil.
- Perbarui kode fungsi dan konfigurasikan dengan variabel lingkungan.
- Panggil fungsi dengan parameter baru dan dapatkan hasil. Tampilkan log eksekusi yang dikembalikan.
- Buat daftar fungsi untuk akun Anda, lalu bersihkan sumber daya.

Untuk informasi selengkapnya, lihat Membuat fungsi Lambda dengan konsol.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Buat skenario interaktif yang menunjukkan cara memulai fungsi 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
}
```

Buat struct yang membungkus tindakan Lambda individual.

```
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"`
}
```

Tentukan handler Lambda yang menambah angka.

```
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
```

Tentukan handler Lambda kedua yang melakukan operasi aritmatika.

```
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
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Go .
  - CreateFunction
  - DeleteFunction
  - GetFunction
  - Memohon
  - ListFunctions
  - UpdateFunctionCode
  - UpdateFunctionConfiguration

Tindakan

`CreateFunction`

Contoh kode berikut menunjukkan cara menggunakan`CreateFunction`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat CreateFunctiondi Referensi AWS SDK for Go API.

`DeleteFunction`

Contoh kode berikut menunjukkan cara menggunakan`DeleteFunction`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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)
 }
}
```

- Untuk detail API, lihat DeleteFunctiondi Referensi AWS SDK for Go API.

`GetFunction`

Contoh kode berikut menunjukkan cara menggunakan`GetFunction`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat GetFunctiondi Referensi AWS SDK for Go API.

`Invoke`

Contoh kode berikut menunjukkan cara menggunakan`Invoke`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat Memanggil di Referensi AWS SDK for Go API.

`ListFunctions`

Contoh kode berikut menunjukkan cara menggunakan`ListFunctions`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat ListFunctionsdi Referensi AWS SDK for Go API.

`UpdateFunctionCode`

Contoh kode berikut menunjukkan cara menggunakan`UpdateFunctionCode`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat UpdateFunctionCodedi Referensi AWS SDK for Go API.

`UpdateFunctionConfiguration`

Contoh kode berikut menunjukkan cara menggunakan`UpdateFunctionConfiguration`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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)
 }
}
```

- Untuk detail API, lihat UpdateFunctionConfigurationdi Referensi AWS SDK for Go API.

Skenario

Secara otomatis mengonfirmasi pengguna yang dikenal dengan fungsi Lambda

Contoh kode berikut menunjukkan cara mengonfirmasi pengguna Amazon Cognito yang diketahui secara otomatis dengan fungsi Lambda.

- Konfigurasikan kumpulan pengguna untuk memanggil fungsi Lambda untuk `PreSignUp` pemicunya.

- Daftarkan pengguna dengan Amazon Cognito.
- Fungsi Lambda memindai tabel DynamoDB dan secara otomatis mengonfirmasi pengguna yang dikenal.
- Masuk sebagai pengguna baru, lalu bersihkan sumber daya.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkap dan pelajari cara menyiapkan dan menjalankan di Repositori Contoh Kode AWS.

Jalankan skenario interaktif di penggugah/prompt perintah.

```
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))
}
```

Tangani `PreSignUp` pelatuk dengan fungsi 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)
}
```

Buat struct yang melakukan tugas-tugas umum.

```
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))
}
```

Buat struct yang membungkus tindakan 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
}
```

Buat struct yang membungkus tindakan 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
}
```

Buat struct yang membungkus tindakan CloudWatch Log.

```
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
}
```

Buat struct yang membungkus tindakan 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
}
```

Pembersihan sumber daya

```
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!")
 }
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Go .
  - DeleteUser
  - InitiateAuth
  - SignUp
  - UpdateUserPool

Secara otomatis memigrasikan pengguna yang dikenal dengan fungsi Lambda

Contoh kode berikut menunjukkan cara memigrasi pengguna Amazon Cognito yang dikenal secara otomatis dengan fungsi Lambda.

- Konfigurasikan kumpulan pengguna untuk memanggil fungsi Lambda untuk `MigrateUser` pemicunya.
- Masuk ke Amazon Cognito dengan nama pengguna dan email yang tidak ada di kumpulan pengguna.
- Fungsi Lambda memindai tabel DynamoDB dan secara otomatis memigrasikan pengguna yang dikenal ke kumpulan pengguna.
- Lakukan alur lupa kata sandi untuk mengatur ulang kata sandi untuk pengguna yang dimigrasi.
- Masuk sebagai pengguna baru, lalu bersihkan sumber daya.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkap dan pelajari cara menyiapkan dan menjalankan di Repositori Contoh Kode AWS.

Jalankan skenario interaktif di penggugah/prompt perintah.

```
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))
}
```

Tangani `MigrateUser` pelatuk dengan fungsi 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)
}
```

Buat struct yang melakukan tugas-tugas umum.

```
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))
}
```

Buat struct yang membungkus tindakan 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
}
```

Buat struct yang membungkus tindakan 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
}
```

Buat struct yang membungkus tindakan CloudWatch Log.

```
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
}
```

Buat struct yang membungkus tindakan 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
}
```

Pembersihan sumber daya

```
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!")
 }
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Go .
  - ConfirmForgotPassword
  - DeleteUser
  - ForgotPassword
  - InitiateAuth
  - SignUp
  - UpdateUserPool

Menulis data aktivitas khusus dengan fungsi Lambda setelah autentikasi pengguna Amazon Cognito

Contoh kode berikut menunjukkan cara menulis data aktivitas kustom dengan fungsi Lambda setelah autentikasi pengguna Amazon Cognito.

- Gunakan fungsi administrator untuk menambahkan pengguna ke kumpulan pengguna.
- Konfigurasikan kumpulan pengguna untuk memanggil fungsi Lambda untuk `PostAuthentication` pemicunya.
- Masuk pengguna baru ke Amazon Cognito.
- Fungsi Lambda menulis informasi kustom ke CloudWatch Log dan ke tabel DynamoDB.
- Dapatkan dan tampilkan data kustom dari tabel DynamoDB, lalu bersihkan sumber daya.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkap dan pelajari cara menyiapkan dan menjalankan di Repositori Contoh Kode AWS.

Jalankan skenario interaktif di penggugah/prompt perintah.

```
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))
}
```

Tangani `PostAuthentication` pelatuk dengan fungsi 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)
}
```

Buat struct yang melakukan tugas-tugas umum.

```
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))
}
```

Buat struct yang membungkus tindakan 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
}
```

Buat struct yang membungkus tindakan 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
}
```

Buat struct yang membungkus tindakan CloudWatch Log.

```
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
}
```

Buat struct yang membungkus tindakan 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
}
```

Pembersihan sumber daya

```
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!")
 }
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Go .
  - AdminCreateUser
  - AdminSetUserPassword
  - DeleteUser
  - InitiateAuth
  - UpdateUserPool

Contoh nirserver

Menghubungkan ke database Amazon RDS dalam fungsi Lambda

Contoh kode berikut menunjukkan bagaimana menerapkan fungsi Lambda yang menghubungkan ke database RDS. Fungsi membuat permintaan database sederhana dan mengembalikan hasilnya.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Menghubungkan ke database Amazon RDS dalam fungsi Lambda menggunakan 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)
```

```
}
```

Memanggil fungsi Lambda dari pemicu Kinesis

Contoh kode berikut menunjukkan bagaimana menerapkan fungsi Lambda yang menerima peristiwa yang dipicu oleh menerima catatan dari aliran Kinesis. Fungsi mengambil payload Kinesis, mendekode dari Base64, dan mencatat konten rekaman.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Mengkonsumsi acara Kinesis dengan Lambda menggunakan 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)
}
```

Memanggil fungsi Lambda dari pemicu DynamoDB

Contoh kode berikut menunjukkan bagaimana menerapkan fungsi Lambda yang menerima peristiwa yang dipicu oleh menerima catatan dari aliran DynamoDB. Fungsi mengambil payload DynamoDB dan mencatat isi catatan.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Mengkonsumsi acara DynamoDB dengan Lambda menggunakan 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)
}
```

Memanggil fungsi Lambda dari pemicu Amazon DocumentDB

Contoh kode berikut menunjukkan cara menerapkan fungsi Lambda yang menerima peristiwa yang dipicu dengan menerima catatan dari aliran perubahan DocumentDB. Fungsi mengambil payload DocumentDB dan mencatat isi catatan.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Mengkonsumsi acara Amazon DocumentDB dengan Lambda menggunakan 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))
}
```

Memanggil fungsi Lambda dari pemicu MSK Amazon

Contoh kode berikut menunjukkan cara menerapkan fungsi Lambda yang menerima peristiwa yang dipicu dengan menerima catatan dari kluster MSK Amazon. Fungsi mengambil muatan MSK dan mencatat konten catatan.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Mengkonsumsi acara MSK Amazon dengan Lambda menggunakan 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)
}
```

Menginvokasi fungsi Lambda dari pemicu Amazon S3

Contoh kode berikut menunjukkan cara mengimplementasikan fungsi Lambda yang menerima peristiwa yang dipicu dengan mengunggah objek ke bucket S3. Fungsi ini mengambil nama bucket S3 dan kunci objek dari parameter peristiwa dan memanggil Amazon S3 API untuk mengambil dan mencatat jenis konten objek.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Menggunakan peristiwa S3 dengan Lambda menggunakan 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)
}
```

Memanggil fungsi Lambda dari pemicu Amazon SNS

Contoh kode berikut menunjukkan cara menerapkan fungsi Lambda yang menerima peristiwa yang dipicu dengan menerima pesan dari topik SNS. Fungsi mengambil pesan dari parameter peristiwa dan mencatat konten setiap pesan.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Mengkonsumsi acara SNS dengan Lambda menggunakan 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)
}
```

Memanggil fungsi Lambda dari pemicu Amazon SQS

Contoh kode berikut menunjukkan bagaimana menerapkan fungsi Lambda yang menerima peristiwa yang dipicu oleh menerima pesan dari antrian SQS. Fungsi mengambil pesan dari parameter peristiwa dan mencatat konten setiap pesan.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Mengkonsumsi acara SQS dengan Lambda menggunakan 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)
}
```

Melaporkan kegagalan item batch untuk fungsi Lambda dengan pemicu Kinesis

Contoh kode berikut menunjukkan cara mengimplementasikan respons batch sebagian untuk fungsi Lambda yang menerima peristiwa dari aliran Kinesis. Fungsi melaporkan kegagalan item batch dalam respons, memberi sinyal ke Lambda untuk mencoba lagi pesan tersebut nanti.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Melaporkan kegagalan item batch Kinesis dengan Lambda menggunakan 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)
}
```

Melaporkan kegagalan item batch untuk fungsi Lambda dengan pemicu DynamoDB

Contoh kode berikut menunjukkan cara mengimplementasikan respons batch sebagian untuk fungsi Lambda yang menerima peristiwa dari aliran DynamoDB. Fungsi melaporkan kegagalan item batch dalam respons, memberi sinyal ke Lambda untuk mencoba lagi pesan tersebut nanti.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Melaporkan kegagalan item batch DynamoDB dengan Lambda menggunakan 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)
}
```

Melaporkan kegagalan item batch untuk fungsi Lambda dengan pemicu Amazon SQS

Contoh kode berikut menunjukkan cara mengimplementasikan respons batch sebagian untuk fungsi Lambda yang menerima peristiwa dari antrian SQS. Fungsi melaporkan kegagalan item batch dalam respons, memberi sinyal ke Lambda untuk mencoba lagi pesan tersebut nanti.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Melaporkan kegagalan item batch SQS dengan Lambda menggunakan 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)
}
```

AWS kontribusi komunitas

Membangun dan menguji aplikasi tanpa server

Contoh kode berikut menunjukkan cara membangun dan menguji aplikasi tanpa server menggunakan API Gateway dengan Lambda dan DynamoDB

SDK untuk Go V2

Menunjukkan cara membuat dan menguji aplikasi tanpa server yang terdiri dari API Gateway dengan Lambda dan DynamoDB menggunakan Go SDK.

Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di GitHub.

Layanan yang digunakan dalam contoh ini

- API Gateway
- DynamoDB
- Lambda

Contoh MSK Amazon menggunakan SDK for Go V2

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan AWS SDK for Go V2 dengan Amazon MSK.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Contoh nirserver

Contoh nirserver

Memanggil fungsi Lambda dari pemicu MSK Amazon

Contoh kode berikut menunjukkan cara menerapkan fungsi Lambda yang menerima peristiwa yang dipicu dengan menerima catatan dari kluster MSK Amazon. Fungsi mengambil muatan MSK dan mencatat konten catatan.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Mengkonsumsi acara MSK Amazon dengan Lambda menggunakan 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)
}
```

Contoh Partner Central menggunakan SDK for Go V2

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan AWS SDK for Go V2 dengan Partner Central.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Tindakan

Tindakan

`GetOpportunity`

Contoh kode berikut menunjukkan cara menggunakan`GetOpportunity`.

SDK untuk Go V2

Dapatkan kesempatan.

```
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/partnercentralselling"
)

func main() {
 config, err := config.LoadDefaultConfig(context.TODO())

 if err != nil {
  log.Fatal(err)
 }

 config.Region = "us-east-1"

 client := partnercentralselling.NewFromConfig(config)

 output, err := client.GetOpportunity(context.TODO(),
 &partnercentralselling.GetOpportunityInput{
  Identifier: aws.String("O1111111"),
  Catalog:    aws.String("AWS"),
 })

 if err != nil {
  log.Fatal(err)
 }
 log.Println("printing opportuniy...\n")

 jsonOutput, err := json.MarshalIndent(output, "", "    ")

 fmt.Println(string(jsonOutput))
}
```

- Untuk detail API, lihat GetOpportunitydi Referensi AWS SDK for Go API.

`ListOpportunities`

Contoh kode berikut menunjukkan cara menggunakan`ListOpportunities`.

SDK untuk Go V2

Daftar peluang.

```
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/partnercentralselling"
)

func main() {
 config, err := config.LoadDefaultConfig(context.TODO())

 if err != nil {
  log.Fatal(err)
 }

 config.Region = "us-east-1"

 client := partnercentralselling.NewFromConfig(config)

 output, err := client.ListOpportunities(context.TODO(),
 &partnercentralselling.ListOpportunitiesInput{
  MaxResults: aws.Int32(2),
  Catalog:    aws.String("AWS"),
 })

 if err != nil {
  log.Fatal(err)
 }

 jsonOutput, err := json.MarshalIndent(output, "", "    ")
 fmt.Println(string(jsonOutput))
}
```

- Untuk detail API, lihat ListOpportunitiesdi Referensi AWS SDK for Go API.

Contoh Amazon RDS menggunakan SDK for Go V2

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Go V2 dengan Amazon RDS.

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Memulai
- Hal-hal mendasar
- Tindakan
- Contoh nirserver

Memulai

Halo Amazon RDS

Contoh kode berikut menunjukkan cara memulai menggunakan Amazon RDS.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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)
  }
 }
}
```

- Untuk detail API, lihat Menjelaskan DBInstances di Referensi AWS SDK for Go API.

Hal-hal mendasar

Pelajari dasar-dasarnya

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Membuat grup parameter basis data kustom dan mengatur nilai parameter.
- Membuat instans basis data yang dikonfigurasikan untuk menggunakan grup parameter. Instans basis data juga berisi basis data.
- Mengambil cuplikan instans.
- Menghapus instans dan grup parameter.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkap dan pelajari cara menyiapkan dan menjalankan di Repositori Contoh Kode AWS.

Jalankan skenario interaktif di penggugah/prompt perintah.

```
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()
}
```

Tentukan fungsi-fungsi yang dipanggil oleh skenario untuk mengelola tindakan 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
 } 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
}
```

- Lihat detail API di topik-topik berikut dalam Referensi API AWS SDK for Go .
  - Buat DBInstance
  - Buat DBParameter Grup
  - Buat DBSnapshot
  - Hapus DBInstance
  - Hapus DBParameter Grup
  - Jelaskan DBEngine Versi
  - Jelaskan DBInstances
  - Jelaskan DBParameter Grup
  - Jelaskan DBParameters
  - Jelaskan DBSnapshots
  - DescribeOrderableDBInstancePilihan
  - Ubah DBParameter Grup

Tindakan

`CreateDBInstance`

Contoh kode berikut menunjukkan cara menggunakan`CreateDBInstance`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
 }
}
```

- Untuk detail API, lihat Membuat DBInstance di Referensi AWS SDK for Go API.

`CreateDBParameterGroup`

Contoh kode berikut menunjukkan cara menggunakan`CreateDBParameterGroup`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
 }
}
```

- Untuk detail API, lihat Membuat DBParameter Grup di Referensi AWS SDK for Go API.

`CreateDBSnapshot`

Contoh kode berikut menunjukkan cara menggunakan`CreateDBSnapshot`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
 }
}
```

- Untuk detail API, lihat Membuat DBSnapshot di Referensi AWS SDK for Go API.

`DeleteDBInstance`

Contoh kode berikut menunjukkan cara menggunakan`DeleteDBInstance`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
 }
}
```

- Untuk detail API, lihat Menghapus DBInstance di Referensi AWS SDK for Go API.

`DeleteDBParameterGroup`

Contoh kode berikut menunjukkan cara menggunakan`DeleteDBParameterGroup`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
 }
}
```

- Untuk detail API, lihat Menghapus DBParameter Grup di Referensi AWS SDK for Go API.

`DescribeDBEngineVersions`

Contoh kode berikut menunjukkan cara menggunakan`DescribeDBEngineVersions`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
 }
}
```

- Untuk detail API, lihat Menjelaskan DBEngine Versi dalam Referensi AWS SDK for Go API.

`DescribeDBInstances`

Contoh kode berikut menunjukkan cara menggunakan`DescribeDBInstances`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
 }
}
```

- Untuk detail API, lihat Menjelaskan DBInstances di Referensi AWS SDK for Go API.

`DescribeDBParameterGroups`

Contoh kode berikut menunjukkan cara menggunakan`DescribeDBParameterGroups`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
 }
}
```

- Untuk detail API, lihat Menjelaskan DBParameter Grup dalam Referensi AWS SDK for Go API.

`DescribeDBParameters`

Contoh kode berikut menunjukkan cara menggunakan`DescribeDBParameters`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat Menjelaskan DBParameters di Referensi AWS SDK for Go API.

`DescribeDBSnapshots`

Contoh kode berikut menunjukkan cara menggunakan`DescribeDBSnapshots`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
 }
}
```

- Untuk detail API, lihat Menjelaskan DBSnapshots di Referensi AWS SDK for Go API.

`DescribeOrderableDBInstanceOptions`

Contoh kode berikut menunjukkan cara menggunakan`DescribeOrderableDBInstanceOptions`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat DescribeOrderableDBInstanceOpsi di Referensi AWS SDK for Go API.

`ModifyDBParameterGroup`

Contoh kode berikut menunjukkan cara menggunakan`ModifyDBParameterGroup`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
 }
}
```

- Untuk detail API, lihat Memodifikasi DBParameter Grup di Referensi AWS SDK for Go API.

Contoh nirserver

Menghubungkan ke database Amazon RDS dalam fungsi Lambda

Contoh kode berikut menunjukkan bagaimana menerapkan fungsi Lambda yang menghubungkan ke database RDS. Fungsi membuat permintaan database sederhana dan mengembalikan hasilnya.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Menghubungkan ke database Amazon RDS dalam fungsi Lambda menggunakan 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)
}
```

Contoh Amazon Redshift menggunakan SDK for Go V2

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan AWS SDK for Go V2 dengan Amazon Redshift.

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Memulai
- Hal-hal mendasar
- Tindakan

Memulai

Halo Amazon Redshift

Contoh kode berikut menunjukkan cara memulai menggunakan Amazon Redshift.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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%v : %v\n", *cluster.ClusterIdentifier, *cluster.ClusterStatus)
 }
}
```

- Untuk detail API, lihat DescribeClustersdi Referensi AWS SDK for Go API.

Hal-hal mendasar

Pelajari dasar-dasarnya

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Buat cluster Redshift.

- Daftar database di cluster.
- Buat tabel bernama Movies.
- Isi tabel Film.
- Kueri tabel Film berdasarkan tahun.
- Ubah cluster Redshift.
- Hapus cluster Amazon Redshift.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkap dan pelajari cara menyiapkan dan menjalankan di Repositori Contoh Kode 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)
   }
  }
 }
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Go .
  - CreateCluster
  - DescribeClusters
  - DescribeStatement
  - ExecuteStatement
  - GetStatementResult
  - ListDatabasesPaginator
  - ModifyCluster

Tindakan

`CreateCluster`

Contoh kode berikut menunjukkan cara menggunakan`CreateCluster`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat CreateClusterdi Referensi AWS SDK for Go API.

`DeleteCluster`

Contoh kode berikut menunjukkan cara menggunakan`DeleteCluster`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat DeleteClusterdi Referensi AWS SDK for Go API.

`DescribeClusters`

Contoh kode berikut menunjukkan cara menggunakan`DescribeClusters`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat DescribeClustersdi Referensi AWS SDK for Go API.

`ModifyCluster`

Contoh kode berikut menunjukkan cara menggunakan`ModifyCluster`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat ModifyClusterdi Referensi AWS SDK for Go API.

Contoh Amazon S3 menggunakan SDK for Go V2

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan AWS SDK for Go V2 dengan Amazon S3.

Dasar-dasar adalah contoh kode yang menunjukkan kepada Anda bagaimana melakukan operasi penting dalam suatu layanan.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Memulai
- Hal-hal mendasar
- Tindakan
- Skenario
- Contoh nirserver

Memulai

Halo Amazon S3

Contoh kode berikut menunjukkan cara memulai menggunakan Amazon S3.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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%v\n", *bucket.Name)
  }
 }
}
```

- Untuk detail API, lihat ListBucketsdi Referensi AWS SDK for Go API.

Hal-hal mendasar

Pelajari dasar-dasarnya

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Membuat bucket dan mengunggah file ke dalamnya.
- Mengunduh objek dari bucket.
- Menyalin objek ke subfolder di bucket.
- Membuat daftar objek dalam bucket.
- Menghapus objek bucket dan bucket tersebut.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Tentukan struktur yang membungkus tindakan bucket dan objek yang digunakan oleh skenario.

```
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
   }
  } else 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",
 *delObjs.Key)
   } else {
    log.Printf("Deleted %s.\n", *delObjs.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
}
```

Jalankan skenario interaktif yang menunjukkan cara bekerja dengan bucket dan objek 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%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))
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Go .
  - CopyObject
  - CreateBucket
  - DeleteBucket
  - DeleteObjects
  - GetObject
  - ListObjectsV2
  - PutObject

Tindakan

`CopyObject`

Contoh kode berikut menunjukkan cara menggunakan`CopyObject`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat CopyObjectdi Referensi AWS SDK for Go API.

`CreateBucket`

Contoh kode berikut menunjukkan cara menggunakan`CreateBucket`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Buat bucket dengan konfigurasi default.

```
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
}
```

Buat ember dengan penguncian objek dan tunggu sampai ada.

```
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
}
```

- Untuk detail API, lihat CreateBucketdi Referensi AWS SDK for Go API.

`DeleteBucket`

Contoh kode berikut menunjukkan cara menggunakan`DeleteBucket`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat DeleteBucketdi Referensi AWS SDK for Go API.

`DeleteObject`

Contoh kode berikut menunjukkan cara menggunakan`DeleteObject`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat DeleteObjectdi Referensi AWS SDK for Go API.

`DeleteObjects`

Contoh kode berikut menunjukkan cara menggunakan`DeleteObjects`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat DeleteObjectsdi Referensi AWS SDK for Go API.

`GetObject`

Contoh kode berikut menunjukkan cara menggunakan`GetObject`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat GetObjectdi Referensi AWS SDK for Go API.

`GetObjectLegalHold`

Contoh kode berikut menunjukkan cara menggunakan`GetObjectLegalHold`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat GetObjectLegalHolddi Referensi AWS SDK for Go API.

`GetObjectLockConfiguration`

Contoh kode berikut menunjukkan cara menggunakan`GetObjectLockConfiguration`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat GetObjectLockConfigurationdi Referensi AWS SDK for Go API.

`GetObjectRetention`

Contoh kode berikut menunjukkan cara menggunakan`GetObjectRetention`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat GetObjectRetentiondi Referensi AWS SDK for Go API.

`HeadBucket`

Contoh kode berikut menunjukkan cara menggunakan`HeadBucket`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat HeadBucketdi Referensi AWS SDK for Go API.

`ListBuckets`

Contoh kode berikut menunjukkan cara menggunakan`ListBuckets`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat ListBucketsdi Referensi AWS SDK for Go API.

`ListObjectVersions`

Contoh kode berikut menunjukkan cara menggunakan`ListObjectVersions`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat ListObjectVersionsdi Referensi AWS SDK for Go API.

`ListObjectsV2`

Contoh kode berikut menunjukkan cara menggunakan`ListObjectsV2`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat ListObjectsV2 di Referensi AWS SDK for Go API.

`PutObject`

Contoh kode berikut menunjukkan cara menggunakan`PutObject`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Masukkan objek ke dalam ember dengan menggunakan API tingkat rendah.

```
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
}
```

Unggah objek ke bucket dengan menggunakan manajer transfer.

```
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
}
```

- Untuk detail API, lihat PutObjectdi Referensi AWS SDK for Go API.

`PutObjectLegalHold`

Contoh kode berikut menunjukkan cara menggunakan`PutObjectLegalHold`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat PutObjectLegalHolddi Referensi AWS SDK for Go API.

`PutObjectLockConfiguration`

Contoh kode berikut menunjukkan cara menggunakan`PutObjectLockConfiguration`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Atur konfigurasi kunci objek ember.

```
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
}
```

Setel periode retensi default 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
}
```

- Untuk detail API, lihat PutObjectLockConfigurationdi Referensi AWS SDK for Go API.

`PutObjectRetention`

Contoh kode berikut menunjukkan cara menggunakan`PutObjectRetention`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat PutObjectRetentiondi Referensi AWS SDK for Go API.

Skenario

Membuat URL yang telah ditetapkan sebelumnya

Contoh kode berikut menunjukkan cara membuat URL presigned untuk Amazon S3 dan mengunggah objek.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Buat fungsi yang membungkus tindakan S3 yang telah ditetapkan sebelumnya.

```
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
}
```

Jalankan contoh interaktif yang menghasilkan dan menggunakan presigned URLs untuk mengunggah, mengunduh, dan menghapus objek 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))
}
```

Tentukan pembungkus permintaan HTTP yang digunakan oleh contoh untuk membuat permintaan 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)
}
```

Kunci objek Amazon S3

Contoh kode berikut menunjukkan cara bekerja dengan fitur kunci objek S3.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Jalankan skenario interaktif yang mendemonstrasikan fitur kunci objek 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
 (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))
}
```

Tentukan struct yang membungkus tindakan S3 yang digunakan dalam contoh ini.

```
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 _, 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
}
```

Pembersihan sumber daya

```
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)
    }
   }
  }
 }
 delObjects[i] = types.ObjectIdentifier{Key: version.Key, VersionId:
 version.VersionId}
 }
 err = resources.s3Actions.DeleteObjects(ctx, bucket.name, delObjects,
 bucket.retentionEnabled)
 if err != nil {
  switch err.(type) {
  case *types.NoSuchBucket:
   log.Println("Nothing to delete.")
  default:
   panic(err)
  }
 }
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Go .
  - GetObjectLegalHold
  - GetObjectLockConfiguration
  - GetObjectRetention
  - PutObjectLegalHold

- PutObjectLockConfiguration
- PutObjectRetention

Mengunggah atau mengunduh file besar

Contoh kode berikut menunjukkan cara mengunggah atau mengunduh file besar ke dan dari Amazon S3.

Untuk informasi selengkapnya, lihat Pengunggahan objek menggunakan unggahan multibagian.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Buat fungsi yang menggunakan pengelola unggah dan unduhan untuk memecah data menjadi beberapa bagian dan mentransfernya secara bersamaan.

```
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
}
```

Jalankan skenario interaktif yang menunjukkan cara menggunakan pengelola unggahan dan unduhan dalam konteks.

```
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))
}
```

Contoh nirserver

Menginvokasi fungsi Lambda dari pemicu Amazon S3

Contoh kode berikut menunjukkan cara mengimplementasikan fungsi Lambda yang menerima peristiwa yang dipicu dengan mengunggah objek ke bucket S3. Fungsi ini mengambil nama bucket S3 dan kunci objek dari parameter peristiwa dan memanggil Amazon S3 API untuk mengambil dan mencatat jenis konten objek.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Menggunakan peristiwa S3 dengan Lambda menggunakan 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)
}
```

Contoh Amazon SNS menggunakan SDK for Go V2

Contoh kode berikut menunjukkan cara melakukan tindakan dan menerapkan skenario umum dengan menggunakan AWS SDK for Go V2 dengan Amazon SNS.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Memulai
- Tindakan
- Skenario
- Contoh nirserver

Memulai

Halo Amazon SNS

Contoh kode berikut menunjukkan cara memulai menggunakan Amazon SNS.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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)
  }
 }
}
```

- Untuk detail API, lihat ListTopicsdi Referensi AWS SDK for Go API.

Tindakan

`CreateTopic`

Contoh kode berikut menunjukkan cara menggunakan`CreateTopic`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat CreateTopicdi Referensi AWS SDK for Go API.

`DeleteTopic`

Contoh kode berikut menunjukkan cara menggunakan`DeleteTopic`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat DeleteTopicdi Referensi AWS SDK for Go API.

`ListTopics`

Contoh kode berikut menunjukkan cara menggunakan`ListTopics`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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)
  }
 }
```

```
}
```

- Untuk detail API, lihat ListTopicsdi Referensi AWS SDK for Go API.

`Publish`

Contoh kode berikut menunjukkan cara menggunakan`Publish`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat Publikasikan di Referensi AWS SDK for Go API.

`Subscribe`

Contoh kode berikut menunjukkan cara menggunakan`Subscribe`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode AWS.

Berlangganan antrian ke topik dengan filter opsional.

```
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 susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}
```

- Untuk detail API, lihat Berlangganan di Referensi AWS SDK for Go API.

Skenario

Publikasikan pesan ke antrian

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Buat topik (FIFO atau non-FIFO).
- Berlangganan beberapa antrian ke topik dengan opsi untuk menerapkan filter.
- Publikasikan pesan ke topik.
- Polling antrian untuk pesan yang diterima.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkap dan pelajari cara menyiapkan dan menjalankan di Repositori Contoh Kode AWS.

Jalankan skenario interaktif di penggugah/prompt perintah.

```
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 messsage?
 (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))
}
```

Tentukan struct yang membungkus tindakan Amazon SNS yang digunakan dalam contoh ini.

```
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 susbscribe 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
}
```

Tentukan struct yang membungkus tindakan Amazon SQS yang digunakan dalam contoh ini.

```
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
}
```

Pembersihan sumber daya

```
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)
  }
 }
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Go .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publikasikan
  - ReceiveMessage
  - SetQueueAttributes
  - Berlangganan
  - Berhenti berlangganan

Contoh nirserver

Memanggil fungsi Lambda dari pemicu Amazon SNS

Contoh kode berikut menunjukkan cara menerapkan fungsi Lambda yang menerima peristiwa yang dipicu dengan menerima pesan dari topik SNS. Fungsi mengambil pesan dari parameter peristiwa dan mencatat konten setiap pesan.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Mengkonsumsi acara SNS dengan Lambda menggunakan 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)
}
```

Contoh Amazon SQS menggunakan SDK for Go V2

Contoh kode berikut menunjukkan cara melakukan tindakan dan mengimplementasikan skenario umum dengan menggunakan AWS SDK for Go V2 dengan Amazon SQS.

Tindakan merupakan kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Sementara tindakan menunjukkan cara memanggil fungsi layanan individual, Anda dapat melihat tindakan dalam konteks dalam skenario terkait.

Skenario adalah contoh kode yang menunjukkan kepada Anda bagaimana menyelesaikan tugas tertentu dengan memanggil beberapa fungsi dalam layanan atau dikombinasikan dengan yang lain Layanan AWS.

Setiap contoh menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode dalam konteks.

Topik

- Memulai
- Tindakan
- Skenario
- Contoh nirserver

Memulai

Halo Amazon SQS

Contoh kode berikut menunjukkan cara memulai menggunakan Amazon SQS.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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)
  }
 }
}
```

- Untuk detail API, lihat ListQueuesdi Referensi AWS SDK for Go API.

Tindakan

`CreateQueue`

Contoh kode berikut menunjukkan cara menggunakan`CreateQueue`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat CreateQueuedi Referensi AWS SDK for Go API.

`DeleteMessageBatch`

Contoh kode berikut menunjukkan cara menggunakan`DeleteMessageBatch`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat DeleteMessageBatchdi Referensi AWS SDK for Go API.

`DeleteQueue`

Contoh kode berikut menunjukkan cara menggunakan`DeleteQueue`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat DeleteQueuedi Referensi AWS SDK for Go API.

`GetQueueAttributes`

Contoh kode berikut menunjukkan cara menggunakan`GetQueueAttributes`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat GetQueueAttributesdi Referensi AWS SDK for Go API.

`ListQueues`

Contoh kode berikut menunjukkan cara menggunakan`ListQueues`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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)
  }
 }
}
```

- Untuk detail API, lihat ListQueuesdi Referensi AWS SDK for Go API.

`ReceiveMessage`

Contoh kode berikut menunjukkan cara menggunakan`ReceiveMessage`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
}
```

- Untuk detail API, lihat ReceiveMessagedi Referensi AWS SDK for Go API.

`SetQueueAttributes`

Contoh kode berikut menunjukkan cara menggunakan`SetQueueAttributes`.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di Repositori Contoh Kode 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
```

- Untuk detail API, lihat SetQueueAttributesdi Referensi AWS SDK for Go API.

Skenario

Publikasikan pesan ke antrian

Contoh kode berikut ini menunjukkan cara untuk melakukan:

- Buat topik (FIFO atau non-FIFO).
- Berlangganan beberapa antrian ke topik dengan opsi untuk menerapkan filter.
- Publikasikan pesan ke topik.
- Polling antrian untuk pesan yang diterima.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkap dan pelajari cara menyiapkan dan menjalankan di Repositori Contoh Kode AWS.

Jalankan skenario interaktif di penggugah/prompt perintah.

```
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 messsage?
 (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))
}
```

Tentukan struct yang membungkus tindakan Amazon SNS yang digunakan dalam contoh ini.

```
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 susbscribe 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
}
```

Tentukan struct yang membungkus tindakan Amazon SQS yang digunakan dalam contoh ini.

```
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
}
```

Pembersihan sumber daya

```
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)
  }
 }
}
```

- Untuk detail API, lihat topik berikut di Referensi API AWS SDK for Go .
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publikasikan
  - ReceiveMessage
  - SetQueueAttributes
  - Berlangganan
  - Berhenti berlangganan

Contoh nirserver

Memanggil fungsi Lambda dari pemicu Amazon SQS

Contoh kode berikut menunjukkan bagaimana menerapkan fungsi Lambda yang menerima peristiwa yang dipicu oleh menerima pesan dari antrian SQS. Fungsi mengambil pesan dari parameter peristiwa dan mencatat konten setiap pesan.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Mengkonsumsi acara SQS dengan Lambda menggunakan 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)
}
```

Melaporkan kegagalan item batch untuk fungsi Lambda dengan pemicu Amazon SQS

Contoh kode berikut menunjukkan cara mengimplementasikan respons batch sebagian untuk fungsi Lambda yang menerima peristiwa dari antrian SQS. Fungsi melaporkan kegagalan item batch dalam respons, memberi sinyal ke Lambda untuk mencoba lagi pesan tersebut nanti.

SDK untuk Go V2

> **Note**
>
> Ada lebih banyak tentang GitHub. Temukan contoh lengkapnya dan pelajari cara mengatur dan menjalankannya di repositori contoh Nirserver.

Melaporkan kegagalan item batch SQS dengan Lambda menggunakan 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)
}
```

Migrasi ke v2 AWS SDK for Go

Versi Go Minimum

Ini AWS SDK for Go membutuhkan versi minimum Go 1.23. Versi terbaru Go dapat diunduh di halaman Unduhan. Lihat Riwayat Rilis untuk informasi selengkapnya tentang setiap rilis versi Go, dan informasi relevan yang diperlukan untuk meningkatkan.

Modularisasi

AWS SDK for Go Telah diperbarui untuk memanfaatkan modul Go yang menjadi mode pengembangan default di Go 1.13. Sejumlah paket yang disediakan oleh SDK telah dimodulasi dan masing-masing diversi dan dirilis secara independen. Perubahan ini memungkinkan pemodelan ketergantungan aplikasi yang ditingkatkan, dan memungkinkan SDK menyediakan fitur dan fungsionalitas baru yang mengikuti strategi pembuatan versi modul Go.

Daftar berikut adalah beberapa modul Go yang disediakan oleh SDK:

Modul	Deskripsi
`github.com/aws/aws-sdk-go-v2`	Inti SDK
`github.com/aws/aws-sdk-go-v2/config`	Pemuatan Konfigurasi Bersama
`github.com/aws/aws-sdk-go-v2/credentials`	AWS Penyedia Kredensi
`github.com/aws/aws-sdk-go-v2/feature/ec2/imds`	Klien EC2 Layanan Metadata Instans Amazon

Klien layanan SDK dan modul utilitas tingkat yang lebih tinggi bersarang di bawah jalur impor berikut:

Impor Root	Deskripsi
`github.com/aws/aws-sdk-go-v2/service/`	Modul Layanan Klien
`github.com/aws/aws-sdk-go-v2/feature/`	Utilitas Tingkat Tinggi untuk layanan seperti Amazon S3 Transfer Manager

Pemuatan Konfigurasi

Paket sesi dan fungsionalitas terkait diganti dengan sistem konfigurasi yang disederhanakan yang disediakan oleh paket konfigurasi. `config`Paket ini adalah modul Go terpisah, dan dapat disertakan dalam dependensi aplikasi Anda dengan with. `go get`

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

Session.new, sesi. NewSession, NewSessionWithOptions, dan session.must harus dimigrasikan ke config. LoadDefaultConfig.

`config`Paket ini menyediakan beberapa fungsi pembantu yang membantu mengganti pemuatan konfigurasi bersama secara terprogram. Nama-nama fungsi ini diawali dengan `With` diikuti oleh opsi yang mereka timpa. Mari kita lihat beberapa contoh cara memigrasikan penggunaan `session` paket.

Untuk informasi selengkapnya tentang memuat konfigurasi bersama, lihatKonfigurasikan SDK.

Contoh

Migrasi dari ke NewSession LoadDefaultConfig

Contoh berikut menunjukkan bagaimana penggunaan `session.NewSession` tanpa parameter argumen tambahan dimigrasikan ke`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
}
```

Bermigrasi dari NewSession dengan opsi AWS.config

Contoh ini menunjukkan cara memigrasikan penggantian `aws.Config` nilai selama pemuatan konfigurasi. Satu atau lebih fungsi `config.With*` pembantu dapat disediakan `config.LoadDefaultConfig` untuk mengganti nilai konfigurasi yang dimuat. Dalam contoh ini AWS Region diganti untuk `us-west-2` menggunakan konfigurasi. WithRegionfungsi pembantu.

```
// 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
}
```

Migrasi dari NewSessionWithOptions

Contoh ini menunjukkan cara memigrasikan nilai utama selama pemuatan konfigurasi. Nol atau lebih fungsi `config.With*` pembantu dapat disediakan `config.LoadDefaultConfig` untuk mengganti nilai konfigurasi yang dimuat. Dalam contoh ini kami menunjukkan cara mengganti profil target yang digunakan saat memuat konfigurasi bersama 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
}
```

Mengejek dan `*iface`

`*iface`Paket dan antarmuka di dalamnya (misalnya S3iface.s3API) telah dihapus. Definisi antarmuka ini tidak stabil karena rusak setiap kali layanan menambahkan operasi baru.

Penggunaan `*iface` harus diganti dengan antarmuka yang ditentukan pemanggil cakupan untuk operasi layanan yang digunakan:

```
// 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)
}
```

Untuk informasi lebih lanjut, lihat Unit Testing dengan AWS SDK for Go v2.

Penyedia Kredensyal dan Kredensyal

Paket aws/credentials dan penyedia kredensi terkait telah dipindahkan ke lokasi paket kredensyal. `credentials`Paket ini adalah modul Go yang Anda ambil dengan menggunakan`go get`.

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

Rilis AWS SDK for Go v2 memperbarui Penyedia AWS Kredensyal untuk menyediakan antarmuka yang konsisten untuk mengambil AWS Kredensyal. Setiap penyedia mengimplementasikan aws. CredentialsProviderantarmuka, yang mendefinisikan `Retrieve` metode yang mengembalikan `a(aws.Credentials, error)`. AWS.credentials yang analog dengan tipe v1 credentials.Value. AWS SDK for Go

Anda harus membungkus `aws.CredentialsProvider` objek dengan aws. CredentialsCacheuntuk memungkinkan caching kredenal terjadi. Anda gunakan NewCredentialsCacheuntuk membangun `aws.CredentialsCache` objek. Secara default, kredensyal yang dikonfigurasi oleh `config.LoadDefaultConfig` dibungkus dengan. `aws.CredentialsCache`

Tabel berikut mencantumkan perubahan lokasi penyedia AWS kredensyal dari AWS SDK for Go v1 ke v2.

Nama	V1 Impor	V2 Impor
Kredensyal Peran Amazon EC2 IAM	`github.com/aws/aws-sdk-go/aws/credentials/ec2rolecreds`	`github.com/aws/aws-sdk-go-v2/credentials/ec2rolecreds`
Kredensyal Endpoint	`github.com/aws/aws-sdk-go/aws/credentials/endpointcreds`	`github.com/aws/aws-sdk-go-v2/credentials/endpointcreds`
Kredensyal Proses	`github.com/aws/aws-sdk-go/aws/credentials/processcreds`	`github.com/aws/aws-sdk-go-v2/credentials/processcreds`
AWS Security Token Service	`github.com/aws/aws-sdk-go/aws/credentials/stscreds`	`github.com/aws/aws-sdk-go-v2/credentials/stscreds`

Kredensyal Statis

Aplikasi yang menggunakan kredensyal. NewStaticCredentialsuntuk membangun kredensyal statis secara terprogram harus menggunakan kredensyal. NewStaticCredentialsProvider.

Contoh

```
// 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
}
```

Kredensyal Peran Amazon EC2 IAM

Anda harus memigrasikan penggunaan NewCredentialsdan NewCredentialsWithClientuntuk menggunakan New.

`ec2rolecreds`Paket menggunakan opsi fungsional `ec2rolecreds.New` EC2RoleCreds.Options sebagai input, memungkinkan Anda mengganti klien Layanan Metadata EC2 Instans Amazon tertentu untuk digunakan, atau untuk mengganti jendela kedaluwarsa kredensyal.

Contoh

```
// 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
}
```

Kredensyal Endpoint

Anda harus memigrasikan penggunaan NewCredentialsClientdan NewProviderClientuntuk menggunakan New.

NewFungsi `endpointcreds` paket mengambil argumen string yang berisi URL titik akhir HTTP atau HTTPS untuk mengambil kredensyal dari, dan opsi fungsional EndpointCreds.Options untuk mengubah penyedia kredensyal dan mengganti pengaturan konfigurasi tertentu.

Kredensyal Proses

Anda harus memigrasikan penggunaan NewCredentials, NewCredentialsCommand, dan NewCredentialsTimeoutuntuk menggunakan NewProvideratau NewProviderCommand.

`NewProvider`Fungsi `processcreds` paket mengambil argumen string yang merupakan perintah yang akan dieksekusi di shell lingkungan host, dan opsi fungsional Options untuk mengubah penyedia kredensyal dan mengganti pengaturan konfigurasi tertentu.

`NewProviderCommand`mengambil implementasi NewCommandBilderantarmuka yang mendefinisikan perintah proses yang lebih kompleks yang mungkin mengambil satu atau lebih argumen baris perintah, atau memiliki persyaratan lingkungan eksekusi tertentu. DefaultNewCommandBuildermengimplementasikan antarmuka ini, dan mendefinisikan pembuat perintah untuk proses yang memerlukan beberapa argumen baris perintah.

Contoh

```
// 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 Kredensyal

AssumeRole

Anda harus memigrasikan penggunaan NewCredentials, dan NewCredentialsWithClientuntuk menggunakan NewAssumeRoleProvider.

`NewAssumeRoleProvider`Fungsi `stscreds` paket harus dipanggil dengan STS.Client, dan AWS Identity and Access Management ARN Peran diasumsikan dari kredenal yang dikonfigurasi yang disediakan`sts.Client`. Anda juga dapat menyediakan serangkaian opsi fungsional AssumeRoleOptionsuntuk memodifikasi pengaturan opsional lain dari penyedia.

Contoh

```
// 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

Anda harus memigrasikan penggunaan NewWebIdentityCredentials, NewWebIdentityRoleProvider, dan NewWebIdentityRoleProviderWithTokenuntuk digunakan NewWebIdentityRoleProvider.

`NewWebIdentityRoleProvider`Fungsi `stscreds` paket harus dipanggil dengan STS.Client, dan AWS Identity and Access Management ARN Peran diasumsikan menggunakan kredensyal yang dikonfigurasi yang disediakan`sts.Client`, dan implementasi untuk IdentityTokenRetrievermenyediakan token ID 2.0 atau OAuth OpenID Connect. IdentityTokenFileadalah `IdentityTokenRetriever` yang dapat digunakan untuk memberikan token identitas web dari file yang terletak di sistem file host aplikasi. Anda juga dapat menyediakan serangkaian opsi fungsional WebIdentityRoleOptionsuntuk memodifikasi pengaturan opsional lainnya untuk penyedia.

Contoh

```
// 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
}
```

Layanan Klien

AWS SDK for Go menyediakan modul klien layanan yang bersarang di bawah jalur `github.com/aws/aws-sdk-go-v2/service` impor. Setiap klien layanan terkandung dalam paket Go menggunakan pengenal unik setiap layanan. Tabel berikut memberikan beberapa contoh jalur impor layanan di AWS SDK for Go.

Nama Layanan	Jalur Impor V1	Jalur Impor 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`
CloudWatch Log Amazon	`github.com/aws/aws-sdk-go/service/cloudwatchlogs`	`github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs`

Setiap paket klien layanan adalah modul Go berversi independen. Untuk menambahkan klien layanan sebagai dependensi aplikasi Anda, gunakan `go get` perintah dengan jalur impor layanan. Misalnya, untuk menambahkan klien Amazon S3 ke dependensi Anda gunakan

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

Konstruksi Klien

Anda dapat membangun klien dalam AWS SDK for Go menggunakan fungsi `New` atau `NewFromConfig` konstruktor dalam paket klien. Saat bermigrasi dari AWS SDK for Go v1, kami sarankan Anda menggunakan `NewFromConfig` varian, yang akan mengembalikan klien layanan baru menggunakan nilai dari file. `aws.Config` `aws.Config`Nilai akan dibuat saat memuat konfigurasi bersama SDK menggunakan`config.LoadDefaultConfig`. Untuk detail tentang membuat klien layanan, lihatGunakan AWS SDK for Go v2 dengan AWS layanan.

Contoh 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)
```

Contoh 2: Mengganti Pengaturan Klien

```
// 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"
})
```

Titik akhir

Paket titik akhir tidak ada lagi di. AWS SDK for Go Setiap klien layanan sekarang menyematkan metadata AWS titik akhir yang diperlukan dalam paket klien. Ini mengurangi ukuran biner keseluruhan aplikasi yang dikompilasi dengan tidak lagi menyertakan metadata titik akhir untuk layanan yang tidak digunakan oleh aplikasi Anda.

Selain itu, setiap layanan sekarang memperlihatkan antarmuka sendiri untuk resolusi titik akhir di. `EndpointResolverV2` Setiap API mengambil satu set parameter unik untuk layanan`EndpointParameters`, yang nilainya bersumber dari SDK dari berbagai lokasi saat operasi dipanggil.

Secara default, klien layanan menggunakan AWS Wilayah yang dikonfigurasi untuk menyelesaikan titik akhir layanan untuk Wilayah target. Jika aplikasi Anda memerlukan titik akhir kustom, Anda dapat menentukan perilaku kustom pada `EndpointResolverV2` bidang pada `aws.Config` struktur. Jika aplikasi Anda mengimplementasikan Endpoints.Resolver kustom, Anda harus memigrasikannya agar sesuai dengan antarmuka per-layanan baru ini.

Untuk informasi selengkapnya tentang titik akhir dan menerapkan resolver kustom, lihat. Konfigurasikan Titik Akhir Klien

Autentikasi

AWS SDK for Go Mendukung perilaku otentikasi yang lebih canggih, yang memungkinkan penggunaan fitur AWS layanan yang lebih baru seperti codecatalyst dan S3 Express One Zone. Selain itu, perilaku ini dapat disesuaikan berdasarkan per klien.

Memanggil Operasi API

Jumlah metode operasi klien layanan telah berkurang secara signifikan. `<OperationName>`Metode`<OperationName>Request`,`<OperationName>WithContext`, dan semuanya telah dikonsolidasikan ke dalam metode operasi tunggal,`<OperationName>`.

Contoh

Contoh berikut menunjukkan bagaimana panggilan ke PutObject operasi Amazon S3 akan dimigrasikan dari AWS SDK for Go v1 ke v2.

```
// 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
})
```

Jenis Data Layanan

Jenis input dan output tingkat atas dari suatu operasi ditemukan dalam paket klien layanan. Jenis input dan output untuk operasi tertentu mengikuti pola `<OperationName>Input` dan`<OperationName>Output`, di `OperationName` mana nama operasi yang Anda panggil. Misalnya, bentuk input dan output untuk PutObject operasi Amazon S3 adalah PutObjectInputdan PutObjectOutputmasing-masing.

Semua tipe data layanan lainnya, selain jenis input dan output, telah dimigrasikan ke `types` paket yang terletak di bawah hirarki jalur impor paket klien layanan. Misalnya, s3. AccessControlPolicytipe sekarang terletak di tipe. AccessControlPolicy.

Nilai Pencacahan

SDK sekarang menyediakan pengalaman yang diketik untuk semua bidang enumerasi API. Daripada menggunakan nilai literal string yang disalin dari dokumentasi referensi API layanan, Anda sekarang dapat menggunakan salah satu tipe konkret yang ditemukan dalam `types` paket klien layanan. Misalnya, Anda dapat memberikan PutObjectInput operasi Amazon S3 dengan ACL untuk diterapkan pada objek. Di AWS SDK for Go v1, parameter ini adalah `*string` tipe. Dalam AWS SDK for Go, parameter ini sekarang menjadi tipe. ObjectCannedACL. `types`Paket ini menyediakan konstanta yang dihasilkan untuk nilai enumerasi valid yang dapat ditetapkan ke bidang ini. Misalnya jenis. ObjectCannedACLPrivateadalah konstanta untuk nilai ACL kalengan “pribadi”. Nilai ini dapat digunakan sebagai pengganti mengelola konstanta string dalam aplikasi Anda.

Parameter Pointer

Referensi pointer AWS SDK for Go v1 diperlukan untuk diteruskan untuk semua parameter input ke operasi layanan. AWS SDK for Go V2 telah menyederhanakan pengalaman dengan sebagian besar layanan dengan menghilangkan kebutuhan untuk meneruskan nilai input sebagai pointer jika memungkinkan. Perubahan ini berarti bahwa banyak operasi klien layanan tidak lagi memerlukan aplikasi Anda untuk meneruskan referensi pointer untuk jenis berikut:`uint8`,,`uint16`,`uint32`,`int8`,`int16`,`int32`, `float32``float64`,`bool`. Demikian pula, tipe

elemen slice dan map telah diperbarui sesuai untuk mencerminkan apakah elemen mereka harus diteruskan sebagai referensi pointer.

Paket aws berisi fungsi pembantu untuk membuat pointer untuk tipe bawaan Go, pembantu ini harus digunakan untuk lebih mudah menangani pembuatan tipe penunjuk untuk tipe Go ini. Demikian pula, metode pembantu disediakan untuk menghilangkan referensi nilai pointer dengan aman untuk jenis ini. Misalnya, fungsi AWS.String mengkonversi dari ⇒. `string` `*string` Sebaliknya, aws. ToStringmengkonversi dari `*string` ⇒`string`. Saat memutakhirkan aplikasi dari AWS SDK for Go v1 ke v2, Anda harus memigrasikan penggunaan helper untuk mengonversi dari tipe pointer ke varian non-pointer. Misalnya, aws. StringValueharus diperbarui ke`aws.ToString`.

Jenis Kesalahan

Ini AWS SDK for Go mengambil keuntungan penuh dari fungsionalitas pembungkus kesalahan yang diperkenalkan di Go 1.13. Layanan yang memodelkan respons kesalahan telah menghasilkan jenis yang tersedia dalam `types` paket klien mereka yang dapat digunakan untuk menguji apakah kesalahan operasi klien disebabkan oleh salah satu jenis ini. Misalnya, `GetObject` operasi Amazon S3 dapat mengembalikan `NoSuchKey` kesalahan jika mencoba mengambil kunci objek yang tidak ada. Anda dapat menggunakan errors.as untuk menguji apakah kesalahan operasi yang dikembalikan adalah tipe. NoSuchKeykesalahan. Jika layanan tidak memodelkan jenis tertentu untuk kesalahan, Anda dapat menggunakan bengkel. APIErrorjenis antarmuka untuk memeriksa kode kesalahan yang dikembalikan dan pesan dari layanan. Fungsionalitas ini menggantikan awserr.Error dan fungsionalitas awserr lainnya dari v1. AWS SDK for Go Untuk informasi detail selengkapnya tentang penanganan kesalahan, lihatMenangani Kesalahan di AWS SDK for Go V2.

Contoh

```
// 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 nsk *types.NoSuchKey
    if errors.As(err, &nsk) {
        // 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
}
```

Paginator

Paginator operasi layanan tidak lagi dipanggil sebagai metode pada klien layanan. Untuk menggunakan paginator untuk operasi Anda harus membangun paginator untuk operasi menggunakan salah satu metode konstruktor paginator. Misalnya, untuk menggunakan paginate di atas operasi Amazon `ListObjectsV2` S3, Anda harus membuat paginatornya menggunakan s3. NewListObjectsV2Paginator. Konstruktor ini mengembalikan ListObjectsV2Paginator yang menyediakan metode`HasMorePages`, dan `NextPage` untuk menentukan apakah ada lebih banyak halaman untuk mengambil dan menjalankan operasi untuk mengambil halaman berikutnya masing-masing. Rincian lebih lanjut tentang penggunaan paginator SDK dapat ditemukan di. Menggunakan Operation Paginators

Mari kita lihat contoh cara bermigrasi dari paginator AWS SDK for Go v1 ke setara v2. AWS SDK for Go

Contoh

```
// 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)
```

Pelayan

Pelayan operasi layanan tidak lagi dipanggil sebagai metode pada klien layanan. Untuk menggunakan pelayan, pertama-tama Anda membangun jenis pelayan yang diinginkan, dan kemudian memanggil metode tunggu. Misalnya, untuk menunggu Bucket Amazon S3 ada, Anda harus membuat pelayan. `BucketExists` Gunakan s3. NewBucketExistsWaiterkonstruktor untuk membuat s3. BucketExistsWaiter. `s3.BucketExistsWaiter`Ini menyediakan `Wait` metode yang dapat digunakan untuk menunggu ember tersedia.

Permintaan yang Ditandatangani

V1 SDK secara teknis mendukung presigning operasi AWS SDK apa pun, namun, ini tidak secara akurat mewakili apa yang sebenarnya didukung di tingkat layanan (dan pada kenyataannya sebagian besar operasi AWS layanan tidak mendukung presigning).

AWS SDK for Go menyelesaikan ini dengan mengekspos `PresignClient` implementasi spesifik dalam paket layanan dengan spesifik APIs untuk operasi presignable yang didukung.

Catatan: Jika layanan tidak memiliki dukungan pra-penandatanganan untuk operasi yang berhasil Anda gunakan di SDK v1, beri tahu kami dengan mengajukan masalah. GitHub

Penggunaan Presign dan PresignRequestharus dikonversi untuk menggunakan klien presigning khusus layanan.

Contoh berikut menunjukkan cara memigrasikan presigning permintaan GetObject S3:

```
// 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)
}
```

Minta kustomisasi

API request.request monolitik telah dikotak-kotak-kotak ulang.

Input/output operasi

`Request`Bidang buram `Params` dan`Data`, yang masing-masing memegang struktur input dan output operasi, sekarang dapat diakses dalam fase middleware tertentu sebagai input/output:

Minta penangan yang mereferensikan `Request.Params` dan `Request.Data` harus dimigrasikan ke middleware.

bermigrasi **`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.ObjectCannedACLBucketOwnerFullContr

    // ...
}
```

```
// 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))
    // ...
}
```

bermigrasi **`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)
    // ...
}
```

Permintaan/tanggapan HTTP

HTTPResponseBidang HTTPRequest dan dari sekarang Request diekspos dalam fase middleware tertentu. Karena middleware adalah transport-agnostik, Anda harus melakukan pernyataan tipe pada input atau output middleware untuk mengungkapkan permintaan atau respons HTTP yang mendasarinya.

Minta penangan yang mereferensikan Request.HTTPRequest dan Request.HTTPResponse harus dimigrasikan ke middleware.

bermigrasi **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", "..."))
    // ...
}
```

Fase handler

Fase middleware SDK v2 adalah penerus fase handler v1.

Tabel berikut menyediakan pemetaan kasar fase handler v1 ke lokasi yang setara dalam tumpukan middleware V2:

nama handler v1	fase middleware v2
Validasi	Inisialisasi
Membangun	Serialisasi
Sign	Selesaikan
Kirim	n/a (1)
ValidateResponse	Deserialisasi
Unmarshal	Deserialisasi
UnmarshalMetadata	Deserialisasi
UnmarshalError	Deserialisasi

nama handler v1	fase middleware v2
Coba lagi	Selesaikan, setelah `"Retry"` middleware (2)
AfterRetry	Selesaikan, sebelum `"Retry"` middleware, post- `next.HandleFinalize()` (2,3)
CompleteAttempt	Selesaikan, akhir langkah
Selesai	Inisialisasi, mulai langkah, posting- `next.HandleInitialize()` (3)

(1) `Send` Fase di v1 secara efektif adalah pulang-pergi klien HTTP yang dibungkus di v2. Perilaku ini dikendalikan oleh `HTTPClient` bidang pada opsi klien.

(2) Setiap middleware setelah `"Retry"` middleware di langkah Finalize akan menjadi bagian dari loop coba lagi.

(3) “Tumpukan” middleware pada waktu operasi dibangun ke dalam fungsi handler yang didekorasi berulang kali. Setiap pawang bertanggung jawab untuk memanggil yang berikutnya dalam rantai. Ini secara implisit berarti bahwa langkah middleware juga dapat mengambil tindakan SETELAH langkah selanjutnya dipanggil.

Misalnya, untuk langkah Inisialisasi, yang berada di bagian atas tumpukan, ini berarti Inisialisasi middlewares yang mengambil tindakan setelah memanggil handler berikutnya beroperasi secara efektif di akhir permintaan:

```
// 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
}
```

Fitur

Layanan EC2 Metadata Instans Amazon

AWS SDK for Go Ini menyediakan klien Amazon EC2 Instance Metadata Service (IMDS) yang dapat Anda gunakan untuk menanyakan IMDS lokal saat menjalankan aplikasi Anda di instans Amazon. EC2 Klien IMDS adalah modul Go terpisah yang dapat ditambahkan ke aplikasi Anda dengan menggunakan

```
go get github.com/aws/aws-sdk-go-v2/feature/ec2/imds
```

Konstruktor klien dan operasi metode telah diperbarui agar sesuai dengan desain klien layanan SDK lainnya.

Contoh

```
// 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
}
```

Manajer Transfer Amazon S3

Manajer transfer Amazon S3 tersedia untuk mengelola unggahan dan unduhan objek secara bersamaan. Paket ini terletak di modul Go di luar jalur impor klien layanan. Modul ini dapat diambil dengan menggunakango `get github.com/aws/aws-sdk-go-v2/feature/s3/manager`.

s3. NewUploaderdan s3. NewUploaderWithClienttelah diganti dengan manajer metode konstruktor. NewUploaderuntuk membuat klien pengelola Unggah.

s3. NewDownloaderdan s3. NewDownloaderWithClienttelah diganti dengan manajer metode konstruktor tunggal. NewDownloaderuntuk membuat klien Download manager.

Utilitas CloudFront Penandatanganan Amazon

AWS SDK for Go Ini menyediakan utilitas CloudFront penandatanganan Amazon dalam modul Go di luar jalur impor klien layanan. Modul ini dapat diambil dengan menggunakango `get`.

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

Klien Enkripsi Amazon S3

Mulai tahun AWS SDK for Go, klien enkripsi Amazon S3 adalah modul terpisah di bawah AWS Crypto Tools. Versi terbaru dari klien enkripsi S3 untuk Go, 3.x, sekarang tersedia di. GitHub Modul ini dapat diambil dengan menggunakango `get`:

```
go get github.com/aws/amazon-s3-encryption-client-go/v3
```

Yang terpisah `EncryptionClient` (v1, v2) dan `DecryptionClient` (v1, v2) APIs telah diganti dengan satu klien, S3 EncryptionClient V3, yang memperlihatkan fungsionalitas enkripsi dan dekripsi.

Seperti klien layanan lainnya AWS SDK for Go, operasi APIs telah diringkas:

- The`GetObject`,`GetObjectRequest`, dan `GetObjectWithContext` dekripsi APIs diganti dengan. GetObject
- `PutObjectWithContext`Enkripsi `PutObjectPutObjectRequest`,, dan APIs digantikan oleh PutObject.

Untuk mempelajari cara bermigrasi ke versi utama 3.x klien enkripsi, lihat panduan ini.

Perubahan Kustomisasi Layanan

Amazon S3

Saat bermigrasi dari AWS SDK for Go v1 ke v2, perubahan penting yang harus diperhatikan melibatkan penanganan yang `SSECustomerKey` digunakan untuk enkripsi sisi server dengan kunci yang disediakan pelanggan (SSE-C). Di AWS SDK for Go v1, pengkodean `SSECustomerKey` to Base64 ditangani secara internal oleh SDK. Di SDK v2, pengkodean otomatis ini telah dihapus, dan sekarang diperlukan untuk menyandikan secara manual ke Base64 sebelum meneruskannya `SSECustomerKey` ke SDK.

Contoh Penyesuaian:

```
// 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
```

Keamanan di AWS SDK for Go

Keamanan cloud di AWS adalah prioritas tertinggi. Sebagai AWS pelanggan, Anda mendapat manfaat dari pusat data dan arsitektur jaringan yang dibangun untuk memenuhi persyaratan organisasi yang paling sensitif terhadap keamanan.

Keamanan adalah tanggung jawab bersama antara Anda AWS dan Anda. Model tanggung jawab bersama menjelaskan hal ini sebagai keamanan cloud dan keamanan dalam cloud:

- Keamanan cloud — AWS bertanggung jawab untuk melindungi infrastruktur yang menjalankan AWS layanan di AWS Cloud. AWS juga memberi Anda layanan yang dapat Anda gunakan dengan aman. Auditor pihak ketiga secara teratur menguji dan memverifikasi efektivitas keamanan kami sebagai bagian dari Program AWS Kepatuhan Program AWS Kepatuhan . Untuk mempelajari tentang program kepatuhan yang berlaku AWS SDK for Go, lihat AWS Layanan dalam Lingkup oleh AWS Layanan Program Kepatuhan .
- Keamanan di cloud — Tanggung jawab Anda ditentukan oleh AWS layanan yang Anda gunakan. Anda juga bertanggung jawab atas faktor lain, yang mencakup sensitivitas data Anda, persyaratan perusahaan Anda, serta undang-undang dan peraturan yang berlaku.

Dokumentasi ini membantu Anda memahami cara menerapkan model tanggung jawab bersama saat menggunakan AWS SDK for Go. Topik berikut menunjukkan cara mengonfigurasi AWS SDK for Go untuk memenuhi tujuan keamanan dan kepatuhan Anda. Anda juga belajar cara menggunakan AWS layanan lain yang membantu Anda memantau dan mengamankan AWS SDK for Go sumber daya Anda.

Topik

- Perlindungan data di AWS SDK for Go
- Validasi kepatuhan untuk AWS SDK for Go
- Ketahanan di AWS SDK for Go

Perlindungan data di AWS SDK for Go

Model tanggung jawab AWS bersama model berlaku untuk perlindungan data di AWS SDK for Go. Seperti yang dijelaskan dalam model AWS ini, bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk mempertahankan

kendali atas konten yang di-host pada infrastruktur ini. Anda juga bertanggung jawab atas tugas-tugas konfigurasi dan manajemen keamanan untuk Layanan AWS yang Anda gunakan. Lihat informasi yang lebih lengkap tentang privasi data dalam Pertanyaan Umum Privasi Data. Lihat informasi tentang perlindungan data di Eropa di pos blog Model Tanggung Jawab Bersama dan GDPR AWS di Blog Keamanan AWS .

Untuk tujuan perlindungan data, kami menyarankan Anda melindungi Akun AWS kredensyal dan mengatur pengguna individu dengan AWS IAM Identity Center atau AWS Identity and Access Management (IAM). Dengan cara itu, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugasnya. Kami juga menyarankan supaya Anda mengamankan data dengan cara-cara berikut:

- Gunakan autentikasi multi-faktor (MFA) pada setiap akun.
- Gunakan SSL/TLS untuk berkomunikasi dengan sumber daya. AWS Kami mensyaratkan TLS 1.2 dan menganjurkan TLS 1.3.
- Siapkan API dan pencatatan aktivitas pengguna dengan AWS CloudTrail. Untuk informasi tentang penggunaan CloudTrail jejak untuk menangkap AWS aktivitas, lihat Bekerja dengan CloudTrail jejak di AWS CloudTrail Panduan Pengguna.
- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default di dalamnya Layanan AWS.
- Gunakan layanan keamanan terkelola tingkat lanjut seperti Amazon Macie, yang membantu menemukan dan mengamankan data sensitif yang disimpan di Amazon S3.
- Jika Anda memerlukan modul kriptografi tervalidasi FIPS 140-3 saat mengakses AWS melalui antarmuka baris perintah atau API, gunakan titik akhir FIPS. Lihat informasi selengkapnya tentang titik akhir FIPS yang tersedia di Standar Pemrosesan Informasi Federal (FIPS) 140-3.

Kami sangat merekomendasikan agar Anda tidak pernah memasukkan informasi identifikasi yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam tanda atau bidang isian bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan AWS SDK for Go atau lainnya Layanan AWS menggunakan konsol, API AWS CLI, atau AWS SDKs. Data apa pun yang Anda masukkan ke dalam tanda atau bidang isian bebas yang digunakan untuk nama dapat digunakan untuk log penagihan atau log diagnostik. Saat Anda memberikan URL ke server eksternal, kami sangat menganjurkan supaya Anda tidak menyertakan informasi kredensial di dalam URL untuk memvalidasi permintaan Anda ke server itu.

Validasi kepatuhan untuk AWS SDK for Go

Untuk mempelajari apakah an Layanan AWS berada dalam lingkup program kepatuhan tertentu, lihat Layanan AWS di Lingkup oleh Program Kepatuhan Layanan AWS dan pilih program kepatuhan yang Anda minati. Untuk informasi umum, lihat Program AWS Kepatuhan Program AWS .

Anda dapat mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat Mengunduh Laporan di AWS Artifact .

Tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, dan hukum dan peraturan yang berlaku. Untuk informasi selengkapnya tentang tanggung jawab kepatuhan Anda saat menggunakan Layanan AWS, lihat Dokumentasi AWS Keamanan.

Ketahanan di AWS SDK for Go

Infrastruktur AWS global dibangun di sekitar Wilayah AWS dan Availability Zones. Wilayah AWS menyediakan beberapa Availability Zone yang terpisah secara fisik dan terisolasi, yang terhubung dengan latensi rendah, throughput tinggi, dan jaringan yang sangat redundan. Dengan Zona Ketersediaan, Anda dapat merancang serta mengoperasikan aplikasi dan basis data yang secara otomatis melakukan fail over di antara zona tanpa gangguan. Zona Ketersediaan memiliki ketersediaan dan toleransi kesalahan yang lebih baik, dan dapat diskalakan dibandingkan infrastruktur pusat data tunggal atau multi tradisional.

Untuk informasi selengkapnya tentang Wilayah AWS dan Availability Zone, lihat Infrastruktur AWS Global.

Riwayat dokumen untuk Panduan Pengembang AWS SDK for Go v2

Tabel berikut menjelaskan rilis dokumentasi untuk AWS SDK for Go v2.

Perubahan	Deskripsi	Tanggal
Pencegat HTTP	Bagian baru yang mencakup pencegat HTTP, keunggulannya dibandingkan middleware, kait yang tersedia, dan metode pendaftaran.	September 4, 2025
Perlindungan integritas data dengan checksum	Konten diperbarui dengan rincian tentang perhitungan checksum otomatis.	Januari 16, 2025
Rilis awal	Rilis awal Panduan Pengembang AWS SDK for Go v2	Oktober 31, 2024

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.