



개발자 가이드

AWS SDK for Go v2



AWS SDK for Go v2: 개발자 가이드

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

Amazon의 상표 및 트레이드 드레스는 Amazon 외 제품 또는 서비스와 함께, Amazon 브랜드 이미지를 떨어뜨리거나 고객에게 혼동을 일으킬 수 있는 방식으로 사용할 수 없습니다. Amazon이 소유하지 않은 기타 모든 상표는 Amazon과 제휴 관계이거나 관련이 있거나 후원 관계와 관계없이 해당 소유자의 자산입니다.

Table of Contents

AWS SDK for Go v2란 무엇입니까?	1
SDK 메이저 버전에 대한 유지 관리 및 지원	1
시작하기	2
Amazon 계정 가져오기	2
AWS SDK for Go v2 설치	2
AWS 액세스 키 받기	3
계정 액세스 키 ID와 시크릿 액세스 키를 얻는 방법은 다음과 같습니다.	3
관련 주제	3
작업 간접 호출	4
SDK 구성	5
AWS 공유 구성 파일 로드	5
AWS 리전 지정	6
환경 변수를 사용하여 리전 구성	6
프로그래밍 방식으로 리전 지정	6
자격 증명 지정	6
태스크에 대한 IAM 역할	8
Amazon EC2 인스턴스의 IAM 역할	8
공유 자격 증명 및 구성	8
환경 변수	11
프로그래밍 방식으로 자격 증명 지정	12
인증	15
정의	16
인증 체계 해결 워크플로	18
기본 지원 AuthScheme	19
클라이언트 엔드포인트	21
사용자 지정	21
V2: EndpointResolverV2 + BaseEndpoint	22
V1: EndpointResolver	26
마이그레이션	28
HTTP 클라이언트	32
구성 로드 중 재정의	32
제한 시간	33
다이얼러	33
운송	34

인터셉터	37
인터셉터와 미들웨어 비교	37
사용 가능한 인터셉터 후크	38
인터셉터 등록	38
글로벌 인터셉터 구성	39
로깅	40
Logger	40
ClientLogMode	40
재시도 및 제한 시간	41
표준 재시도기	41
NopRetryer	42
동작 사용자 지정	42
시간 초과	46
SDK 사용	48
서비스 클라이언트 구성	48
NewFromConfig	48
New	49
서비스 작업 직접 호출	51
파라미터를 서비스 작업에 전달	51
작업 직접 호출을 위한 클라이언트 옵션 재정의	52
작업 응답 처리	53
서비스 클라이언트 동시 사용	55
작업 페이지네이터 사용	56
웨이터 사용	58
웨이터 구성 재정의	59
고급 웨이터 구성 재정의	61
SDK에서 오류 처리	62
오류 로깅	62
서비스 클라이언트 오류	63
요청 식별자 검색	64
AWS 서비스 사용	66
Amazon S3 체크섬	66
객체 업로드	67
객체 다운로드	70
SDK 유틸리티	71
Amazon RDS 유틸리티	71

[IAM Authentication]	71
Amazon CloudFront 유틸리티	72
Amazon CloudFront URL 서명자	72
Amazon EC2 인스턴스 메타데이터 서비스	73
Amazon S3 유틸리티	74
Amazon S3 Transfer Managers	74
확인할 수 없는 스트리밍 입력	84
미들웨어	85
사용자 지정 미들웨어 작성	86
모든 클라이언트에 미들웨어 연결	88
특정 작업에 미들웨어 연결	88
스택으로 메타데이터 전달	89
SDK에서 제공하는 메타데이터	90
스택에 메타데이터 전달	91
FAQ	93
SDK의 HTTP 클라이언트를 구성하려면 어떻게 해야 합니까? 지침이나 모범 사례가 있습니 까?	93
작업 제한 시간을 구성하려면 어떻게 해야 합니까?	93
SDK의 요청이 시간 초과되거나 너무 오래 걸리는데, 이 문제를 해결하려면 어떻게 해야 합니 까?	93
read: connection reset 오류를 해결하려면 어떻게 해야 합니까?	94
SDK에서 HTTP 프록시를 사용할 때 "잘못된 서명" 오류가 발생하는 이유는 무엇입니까?	95
SDK 작업 타이밍	95
유닛 테스트	102
클라이언트 작업 모의 테스트	102
모의 테스트 페이지네이터	104
코드 예제	107
API Gateway	108
AWS 커뮤니티 기여	108
Aurora	109
시작하기	109
기본 사항	111
작업	129
Amazon Bedrock	149
시작하기	109
작업	129

Amazon Bedrock 런타임	152
시작하기	109
시나리오	155
Amazon Titan Image Generator	159
Anthropic Claude	161
CloudFormation	168
작업	129
CloudWatch Logs	170
작업	129
Amazon Cognito 자격 증명 공급자	172
시작하기	109
작업	129
시나리오	155
Amazon DocumentDB	254
서버리스 예제	255
DynamoDB	256
기본 사항	111
작업	129
시나리오	155
서버리스 예제	255
AWS 커뮤니티 기여	108
IAM	329
시작하기	109
기본 사항	111
작업	129
Kinesis	389
서버리스 예제	255
Lambda	392
시작하기	109
기본 사항	111
작업	129
시나리오	155
서버리스 예제	255
AWS 커뮤니티 기여	108
Amazon MSK	504
서버리스 예제	255

Partner Central	506
작업	129
Amazon RDS	508
시작하기	109
기본 사항	111
작업	129
서버리스 예제	255
Amazon Redshift	544
시작하기	109
기본 사항	111
작업	129
Amazon S3	563
시작하기	109
기본 사항	111
작업	129
시나리오	155
서버리스 예제	255
Amazon SNS	652
시작하기	109
작업	129
시나리오	155
서버리스 예제	255
Amazon SQS	680
시작하기	109
작업	129
시나리오	155
서버리스 예제	255
v2로 마이그레이션	713
Go 최소 버전	713
모듈화	713
구성 로드	714
예제	23
모의 테스트 및 *iface	717
자격 증명 및 자격 증명 공급자	718
정적 자격 증명	12
Amazon EC2 IAM 역할 자격 증명	720

엔드포인트 자격 증명	721
프로세스 자격 증명	721
AWS Security Token Service 자격 증명	722
서비스 클라이언트	724
클라이언트 구성	725
엔드포인트	727
Authentication	727
API 작업 간접 호출	727
서비스 데이터 형식	728
포인터 파라미터	729
오류 유형	729
페이지네이터	731
Waiters	733
미리 서명된 요청	733
요청 사용자 지정	735
작업 입력/출력	735
HTTP 요청/응답	739
핸들러 단계	741
특성	743
Amazon EC2 인스턴스 메타데이터 서비스	743
Amazon S3 Transfer Manager	744
Amazon CloudFront 서명 유틸리티	745
Amazon S3 암호화 클라이언트	745
서비스 사용자 지정 변경 사항	745
Amazon S3	745
보안	748
데이터 보호	748
규정 준수 확인	749
복원성	750
문서 기록	751
.....	dcclii

AWS SDK for Go v2란 무엇입니까?

AWS SDK for Go v2는 개발자가 Amazon Elastic Compute Cloud(Amazon EC2) 및 Amazon Simple Storage Service(Amazon S3)와 같은 AWS 서비스를 사용하는 Go 애플리케이션을 구축하는 데 사용할 수 있는 API와 유틸리티를 제공합니다.

SDK는 웹 서비스 인터페이스에 대한 직접 코딩의 복잡성을 제거합니다. 인증, 요청 재시도 및 오류 처리와 같은 많은 하위 수준 배관을 숨깁니다.

SDK에는 유용한 유틸리티도 포함되어 있습니다. 예를 들어 Amazon S3 다운로드 및 업로드 관리자는 대용량 객체를 자동으로 여러 파트로 나누어 병렬로 전송할 수 있습니다.

AWS SDK for Go 개발자 안내서를 사용하여 SDK를 설치, 구성 및 사용할 수 있습니다. 이 가이드에서는 구성 정보, 샘플 코드 및 SDK 유틸리티를 소개합니다.

SDK 메이저 버전에 대한 유지 관리 및 지원

SDK 메이저 버전 및 기본 종속성의 유지 관리 및 지원에 대한 자세한 내용은 [AWS SDK 및 도구 참조 안내서](#)에서 다음 내용을 참조하세요.

- [AWS SDK 및 도구 유지 관리 정책](#)
- [AWS SDK 및 도구 버전 지원 매트릭스](#)

AWS SDK for Go 시작하기

AWS SDK for Go에는 최소한 Go 1.23 버전이 필요합니다. 다음 명령을 실행하여 Go의 최신 버전을 확인할 수 있습니다.

```
go version
```

Go 버전 설치 또는 업그레이드에 대한 자세한 내용은 Go 설명서의 [다운로드 및 설치](#)를 참조하세요.

Amazon 계정 가져오기

AWS SDK for Go v2를 사용하려면 먼저 Amazon 계정이 있어야 합니다. 자세한 내용은 [AWS 계정 생성](#)을 참조하세요.

AWS SDK for Go v2 설치

AWS SDK for Go v2는 Go 1.11에 도입된 기능인 Go 모듈을 사용합니다. 로컬 프로젝트를 초기화하려면 다음 Go 명령을 실행합니다.

```
go mod init example
```

Go 모듈 프로젝트를 초기화한 후 `go get` 명령을 사용하여 SDK와 필요한 종속성을 검색할 수 있습니다. 이러한 종속성은 이전 명령으로 생성된 `go.mod` 파일에 기록됩니다.

다음 명령은 애플리케이션에서 사용할 표준 SDK 모듈 세트를 검색하는 방법을 보여줍니다.

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

그러면 코어 SDK 모듈과 AWS 공유 구성을 로드하는 데 사용되는 구성 모듈이 검색됩니다.

그런 다음 애플리케이션에 필요한 AWS 서비스 API 클라이언트를 하나 이상 설치할 수 있습니다. 모든 API 클라이언트는 `github.com/aws/aws-sdk-go-v2/service` 가져오기 계층 구조 아래에 있습니다. 현재 지원되는 API 클라이언트의 전체 세트는 [여기](#)에서 찾을 수 있습니다. 서비스 클라이언트를 설치하려면 다음 명령을 실행하여 모듈을 검색하고 `go.mod` 파일에 종속성을 기록합니다. 이 예제에서는 Amazon S3 API 클라이언트를 검색합니다.

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

AWS 액세스 키 받기

액세스 키는 액세스 키 ID 및 비밀 액세스 키로 이루어져 있는데, 이를 사용하여 에 보내는 프로그래밍 방식의 요청에 서명할 수 있습니다. AWS 액세스 키가 없는 경우 [AWS Management Console](#)을 사용하여 액세스 키를 생성할 수 있습니다. AWS 루트 계정 액세스 키 대신에 IAM 액세스 키를 사용하는 것이 좋습니다. IAM으로 AWS 서비스와 AWS 계정의 리소스에 대한 액세스 권한을 안전하게 제어할 수 있습니다.

Note

액세스 키를 생성하려면 필요한 IAM 작업을 수행할 수 있는 권한이 있어야 합니다. 자세한 내용은 IAM 사용 설명서의 [IAM 리소스에 액세스하는 데 필요한 권한](#)을 참조하세요.

계정 액세스 키 ID와 시크릿 액세스 키를 얻는 방법은 다음과 같습니다.

1. [IAM 콘솔](#)을 엽니다.
2. 탐색 메뉴에서 사용자를 선택합니다.
3. IAM 사용자 이름(확인란이 아님)을 선택합니다.
4. Security credentials(보안 자격 증명) 탭을 연 다음 Create access key(액세스 키 생성)를 선택합니다.
5. 새 액세스 키를 보려면 [Show]를 선택합니다. 자격 증명은 다음과 같을 것입니다.
 - 액세스 키 ID: AKIAIOSFODNN7EXAMPLE
 - 시크릿 액세스 키: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
6. 키 페어 파일을 다운로드하려면 [Download .csv file]을 선택합니다. 안전한 위치에 키를 저장합니다.

Warning

키를 기밀로 유지하여 AWS 계정을 보호하고 조직 외부의 누구와도 공유하지 마세요.

관련 주제

- IAM 사용 설명서의 [IAM이란 무엇입니까?](#)

- Amazon Web Services 일반 참조의 [AWS 보안 자격 증명](#).

작업 간접 호출

SDK를 설치한 후 AWS, Config 및 Amazon S3 라이브러리를 가져오는 다음 예제와 같이 패키지를 AWS Go 애플리케이션으로 가져와 SDK를 사용합니다. SDK 패키지를 가져온 후 AWS SDK 공유 구성이 로드되고 클라이언트가 구성되며 API 작업이 간접적으로 호출됩니다.

```
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)
    }
}
```

SDK 구성

AWS SDK for Go V2에서는 로거, 로그 수준 및 재시도 구성과 같은 서비스 클라이언트에 대한 공통 설정을 구성할 수 있습니다. 대부분의 설정은 선택 사항입니다. 그러나 각 서비스 클라이언트에 대해 AWS 리전과 자격 증명을 지정해야 합니다. SDK는 이러한 값을 사용하여 올바른 리전으로 요청을 보내고 올바른 자격 증명으로 요청에 서명합니다. 이러한 값은 코드 또는 실행 환경을 통해 프로그래밍 방식으로 지정할 수 있습니다.

AWS 공유 구성 파일 로드

서비스 API 클라이언트를 초기화하는 방법에는 여러 가지가 있지만 사용자에게 권장되는 가장 일반적인 패턴은 다음과 같습니다.

AWS 공유 구성 파일을 사용하도록 SDK를 구성하려면 다음 코드를 사용합니다.

```
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())`는 AWS 공유 구성 소스를 사용하여 [aws.Config](#)를 구성합니다. 여기에는 자격 증명 공급자 구성, AWS 리전 구성, 서비스별 구성 로드가 포함됩니다. 로드된 `aws.Config`를 사용하여 서비스 클라이언트를 구성할 수 있으므로 클라이언트를 구성하기 위한 일관된 패턴을 제공합니다.

AWS 공유 구성 파일에 대한 자세한 내용은 SDK 및 도구 참조 안내서의 [구성을](#) 참조하세요. AWS SDKs

AWS 리전 지정

리전을 지정할 때 us-west-2 또는 us-east-2와 같은 요청을 보낼 위치를 지정합니다. 각 서비스의 리전 목록을 보려면, Amazon Web Services 일반 참조의 [서비스 엔드포인트 및 할당량](#)을 참조하세요.

SDK에는 기본 리전이 없습니다. 리전을 지정하려면 다음을 수행합니다.

- 기본 리전에 AWS_REGION 환경 변수를 설정합니다.
- 구성을 로드할 때 config.LoadDefaultConfig에 [config.WithRegion](#)을 인수로 사용하여 리전을 명시적으로 설정합니다.

검토: 이러한 모든 기술을 사용하여 리전을 설정하는 경우 SDK는 명시적으로 지정한 리전을 사용합니다.

환경 변수를 사용하여 리전 구성

Linux, macOS 또는 Unix

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

Windows

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

프로그래밍 방식으로 리전 지정

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

자격 증명 지정

에는 요청에 서명하기 위한 자격 증명(액세스 키 및 보안 액세스 키)이 AWS SDK for Go 필요합니다. AWS. 특정 사용 사례에 따라 여러 위치에서 자격 증명을 지정할 수 있습니다. 자격 증명을 받는 방법에 대한 자세한 내용은 [AWS SDK for Go 시작하기](#) 섹션을 참조하세요.

를 사용하여 aws.Config 인스턴스를 초기화하면 SDKconfig.LoadDefaultConfig는 기본 자격 증명 체인을 사용하여 AWS 자격 증명을 찾습니다. 이 기본 자격 증명 체인은 다음 순서대로 자격 증명을 찾습니다.

1. 환경 변수.

1. 정적 자격 증명(AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY, AWS_SESSION_TOKEN)
2. 웹 자격 증명 토큰(AWS_WEB_IDENTITY_TOKEN_FILE)

2. 공유 구성 파일.

1. SDK는 기본적으로 컴퓨터의 홈 폴더에 있는 .aws 폴더 아래의 credentials 파일로 설정됩니다.
2. SDK는 기본적으로 컴퓨터의 홈 폴더에 있는 .aws 폴더 아래의 config 파일로 설정됩니다.
3. 애플리케이션이 Amazon ECS 태스크 정의 또는 RunTask API 작업을 사용하는 경우 태스크에 대한 IAM 역할입니다.
4. 애플리케이션이 Amazon EC2 인스턴스에서 실행 중인 경우 Amazon EC2에 대한 IAM 역할입니다.

SDK는 수동 구성 없이 기본 제공 공급자를 자동으로 감지하고 사용합니다. 예를 들어 Amazon EC2 인스턴스에 IAM 역할을 사용하는 경우 애플리케이션은 인스턴스의 자격 증명을 자동으로 사용합니다. 애플리케이션에서 자격 증명을 수동으로 구성할 필요가 없습니다.

가장 좋은 방법은 자격 증명을 다음 순서로 AWS 지정하는 것입니다.

1. 애플리케이션이 Amazon ECS 태스크 정의 또는 RunTask API 작업을 사용하는 경우 태스크에 IAM 역할을 사용합니다.
2. 애플리케이션이 Amazon EC2 인스턴스에서 실행 중인 경우 Amazon EC2에 IAM 역할을 사용합니다.

IAM 역할은 인스턴스의 애플리케이션을 임시 보안 자격 증명으로 제공하여 AWS 호출합니다. IAM 역할은 여러 Amazon EC2 인스턴스에서 보안 인증을 배포 및 관리할 수 있는 쉬운 방법을 제공합니다.

3. 공유 자격 증명 또는 구성 파일을 사용합니다.

자격 증명 및 구성 파일은 다른 AWS SDK 및 AWS CLI에서 공유됩니다. 보안 모범 사례로 액세스 키 ID 및 보안 키와 같은 민감한 값을 설정하려면 자격 증명 파일을 사용하는 것이 좋습니다. 다음은 이러한 각 파일의 [형식 지정 요구 사항](#)입니다.

4. 환경 변수를 사용합니다.

환경 변수 설정은 Amazon EC2 인스턴스 이외의 시스템에서 개발 작업을 수행 중인 경우에 유용합니다.

태스크에 대한 IAM 역할

애플리케이션이 Amazon ECS 작업 정의 또는 RunTask 작업을 사용하는 경우 [작업에 대한 IAM 역할](#)을 사용하여 작업의 컨테이너에서 사용할 수 있는 IAM 역할을 지정합니다.

Amazon EC2 인스턴스의 IAM 역할

Amazon EC2 인스턴스에서 애플리케이션을 실행하는 경우 인스턴스의 [IAM 역할](#)을 사용하여 AWS를 직접 호출하기 위한 임시 보안 자격 증명을 가져옵니다.

IAM 역할을 사용하도록 인스턴스를 구성한 경우 SDK는 애플리케이션에 이러한 자격 증명을 자동으로 사용합니다. 이러한 자격 증명을 수동으로 지정할 필요가 없습니다.

공유 자격 증명 및 구성

공유 자격 증명 및 구성 파일을 사용하여 AWS SDKs 및 기타 도구 간에 공통 구성을 공유할 수 있습니다. 다양한 도구나 애플리케이션에 대해 서로 다른 자격 증명을 사용하는 경우 프로파일을 사용하여 동일한 구성 파일에서 여러 액세스 키를 구성할 수 있습니다.

`config.LoadOptions`를 사용하여 여러 자격 증명 또는 구성 파일 위치를 제공할 수 있습니다. 기본적으로 SDK는 [자격 증명 지정](#)에 언급된 기본 위치에 저장된 파일을 로드합니다.

```
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"},
    )
)
```

공유 자격 증명 및 구성 파일을 사용할 때 중복 프로필이 지정되면 프로필을 확인하기 위해 병합됩니다. 병합 충돌이 발생하는 경우

1. 중복 프로필이 동일한 자격 증명/구성 파일 내에서 지정된 경우 후자 프로필에 지정된 프로필 속성이 우선합니다.
2. 중복 프로필이 여러 자격 증명 파일 또는 여러 구성 파일에 지정된 경우 프로필 속성은 `config.LoadOptions`에 대한 파일 입력 순서에 따라 확인됩니다. 후자의 파일에서 프로필 속성이 우선합니다.
3. 프로필이 자격 증명 파일과 구성 파일 모두에 있는 경우 자격 증명 파일 속성이 우선합니다.

필요한 경우 `config.LoadOptions`에서 `LogConfigurationWarnings`를 활성화하고 프로필 확인 단계를 기록할 수 있습니다.

자격 증명 파일 생성

공유 자격 증명 파일(`.aws/credentials`)이 없는 경우 텍스트 편집기를 사용하여 홈 디렉터리에 파일을 생성할 수 있습니다. 자격 증명 파일에 다음 콘텐츠를 추가하여 `<YOUR_ACCESS_KEY_ID>` 및 `<YOUR_SECRET_ACCESS_KEY>`를 자격 증명으로 바꿉니다.

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

[default] 제목은 다른 프로필을 사용하도록 구성하지 않는 한 SDK가 사용할 기본 프로필의 자격 증명을 정의합니다.

다음 예제와 같이 프로필에 세션 토큰을 추가하여 임시 보안 자격 증명을 사용할 수도 있습니다.

```
[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>
```

자격 증명 파일 내의 기본이 아닌 프로필의 섹션 이름은 단어 `profile`로 시작해서는 안 됩니다. [AWS SDK 및 도구 참조 안내서](#)에서 자세한 내용을 읽을 수 있습니다.

구성 파일 생성

공유 자격 증명 파일(`.aws/config`)이 없는 경우 텍스트 편집기를 사용하여 홈 디렉터리에 파일을 생성할 수 있습니다. 구성 파일에 다음 콘텐츠를 추가하여 `<REGION>`을 원하는 리전으로 바꿉니다.

```
[default]
```

```
region = <REGION>
```

[default] 제목은 다른 프로필을 사용하도록 구성하지 않는 한 SDK가 사용할 기본 프로필의 구성을 정의합니다.

다음 예시와 같이 이름이 지정된 프로필을 사용할 수 있습니다.

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

구성 파일 내의 기본이 아닌 프로필의 섹션 이름은 항상 단어 `profile`로 시작하고 의도한 프로필 이름을 따라야 합니다. [AWS SDK 및 도구 참조 안내서](#)에서 자세한 내용을 읽을 수 있습니다.

프로필 지정

각 액세스 키 세트를 프로필과 연결하여 동일한 구성 파일에 여러 액세스 키를 포함할 수 있습니다. 예를 들어 자격 증명 파일에서 다음과 같이 여러 프로필을 선언할 수 있습니다.

```
[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>
```

기본적으로 SDK는 `AWS_PROFILE` 환경 변수를 확인하여 사용할 프로필을 결정합니다. `AWS_PROFILE` 변수가 설정되지 않은 경우 SDK는 `default` 프로필을 사용합니다.

경우에 따라 애플리케이션에 다른 프로필을 사용할 수 있습니다. 예를 들어 `myapp` 애플리케이션에 `test-account` 자격 증명을 사용하려고 합니다. 다음 명령을 사용하여 이 프로필을 사용할 수 있습니다.

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

`config.LoadDefaultConfig`를 직접 호출하기 전에 `os.Setenv("AWS_PROFILE", "test-account")`를 직접 호출하거나 다음 예제와 같이 명시적 프로필을 인수로 전달하여 SDK에 프로필을 선택하도록 지시할 수도 있습니다.

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

Note

환경 변수에서 자격 증명을 지정하는 경우 SDK는 지정한 프로필에 관계없이 항상 해당 자격 증명을 사용합니다.

환경 변수

기본적으로 SDK는 사용자 환경에서 설정된 AWS 자격 증명을 감지하고 이를 사용하여 요청에 서명합니다 AWS. 이렇게 하면 애플리케이션에서 자격 증명을 관리할 필요가 없습니다.

SDK는 다음 환경 변수에서 자격 증명을 찾습니다.

- `AWS_ACCESS_KEY_ID`
- `AWS_SECRET_ACCESS_KEY`
- `AWS_SESSION_TOKEN` (선택 사항)

다음 예제는 환경 변수를 구성하는 방법을 보여줍니다.

Linux, OS X 또는 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
```

프로그래밍 방식으로 자격 증명 지정

`config.LoadDefaultConfig`를 사용하면 공유 구성 소스를 로드할 때 명시적 [aws.CredentialProvider](#)를 제공할 수 있습니다. 공유 구성을 로드할 때 명시적 자격 증명 공급자를 전달하려면 [config.WithCredentialsProvider](#)를 사용합니다. 예를 들어 `customProvider`가 `aws.CredentialProvider` 구현 인스턴스를 참조하는 경우 다음과 같이 구성 로드 중에 전달할 수 있습니다.

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

이 예제와 같이 자격 증명을 명시적으로 제공하는 경우 SDK는 해당 자격 증명만 사용합니다.

Note

`LoadDefaultConfig`가 전달받거나 반환하는 모든 자격 증명 공급자는 [CredentialsCache](#)에 자동으로 래핑됩니다. 이렇게 하면 동시성 면에서 안전한 캐싱 및 자격 증명 교체가 가능합니다. `aws.Config`에서 직접 공급자를 명시적으로 구성하는 경우 [NewCredentialsCache](#)를 사용하여 이 유형으로 공급자를 명시적으로 래핑해야 합니다.

정적 자격 증명

[credentials.NewStaticCredentialsProvider](#) 자격 증명 공급자를 사용하여 애플리케이션에서 자격 증명을 하드 코딩하여 사용할 액세스 키를 명시적으로 설정할 수 있습니다. 예:

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

Warning

애플리케이션 내에 자격 증명을 포함하지 마세요. 이 메서드는 테스트 목적으로만 사용하세요.

Single Sign-On 자격 증명

SDK를 사용하여 임시 자격 증명을 검색하기 위한 AWS 자격 증명 공급자를 제공합니다 AWS IAM Identity Center. 를 사용하여 AWS 액세스 포털로 AWS CLI인증하고 임시 AWS 자격 증명에 대한 액세스를 승인합니다. 그런 다음 Single Sign-On(SSO) 프로파일을 로드하도록 애플리케이션을 구성하고 SDK는 SSO 자격 증명을 사용하여 만료될 경우 자동으로 갱신되는 임시 AWS 자격 증명을 검색합니다. SSO 자격 증명 만료되면를 사용하여 IAM Identity Center 계정에 다시 로그인하여 명시적으로 갱신해야 합니다 AWS CLI.

예를 들어 프로필을 생성하고,를 사용하여 해당 프로필을 dev-profile인증 및 승인하고 AWS CLI, 아래와 같이 애플리케이션을 구성할 수 있습니다.

1. 먼저 profile 및 sso-session을 생성합니다.

```
[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. AWS CLI 를 사용하여 로그인하여 SSO 프로파일을 인증하고 승인합니다.

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

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

Then enter the code:

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

3. 그런 다음 SSO 프로필을 사용하도록 애플리케이션을 구성합니다.

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

// ...

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

SSO 프로필 구성 및를 사용한 인증에 대한 자세한 내용은 사용 설명서의 [AWS CLI 를 사용하도록 구성을 AWS IAM Identity Center](#) AWS CLI 참조하세요 AWS CLI . SSO 자격 증명 공급자를 프로그래밍 방식으로 구성하는 방법에 대한 자세한 내용은 [ssocreds](#) API 참조 설명서를 참조하세요.

로그인 자격 증명

기존 AWS Management Console 로그인 자격 증명을 사용하여 AWS 서비스에 프로그래밍 방식으로 액세스할 수 있습니다. 브라우저 기반 인증 흐름 후는 AWS CLI 및 AWS SDKs AWS Tools for PowerShell 와 같은 로컬 개발 도구에서 작동하는 임시 자격 증명을 AWS 생성합니다. 이 기능은 특히 장기 액세스 키를 관리하는 것보다 대화형 인증을 선호하는 경우 AWS CLI 자격 증명을 구성하고 관리하는 프로세스를 간소화합니다.

1. AWS CLI를 사용하여 로그인 흐름을 시작하고 브라우저 프롬프트를 따릅니다. 이 예제에서는 로그인 세션을 새 프로파일에 저장dev-profile하지만 이는 선택 사항입니다.

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

2. (선택 사항) AWS 공유 구성 파일을 검사하여 세션이 설정되었는지 확인합니다.

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

3. 그런 다음 로그인 프로필을 사용하도록 애플리케이션을 구성합니다.

```
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
}
```

로그인 프로필 구성 및 AWS CLI를 사용한 인증에 대한 자세한 내용은 [콘솔 자격 증명을 사용한 AWS 로컬 개발을 위한 로그인](#)을 참조하세요.

다른 자격 증명 공급자

SDK는 [자격 증명](#) 모듈에서 자격 증명을 검색하는 다른 방법을 제공합니다. 예를 들어에서 임시 보안 자격 증명을 검색 AWS Security Token Service 하거나 암호화된 스토리지에서 자격 증명을 검색할 수 있습니다.

사용 가능한 자격 증명 공급자:

- [ec2rolecreds](#) - Amazon EC2 IMDS를 통해 Amazon EC2 인스턴스 역할에서 자격 증명을 검색합니다.
- [endpointcreds](#) - 임의의 HTTP 엔드포인트에서 자격 증명을 검색합니다.
- [processcreds](#) - 호스트 환경의 셸에서 간접적으로 호출할 외부 프로세스에서 자격 증명을 검색합니다.
- [stscreds](#) -에서 자격 증명 검색 AWS STS

인증 구성

AWS SDK for Go는 인증 동작 서비스를 구성하는 기능을 제공합니다. 대부분의 경우 기본 구성으로도 충분하지만 사용자 지정 인증을 구성하면 시험판 서비스 기능 작업과 같은 추가 동작을 수행할 수 있습니다.

정의

이 섹션에서는 AWS SDK for Go의 인증 구성 요소에 대한 개략적인 설명을 제공합니다.

AuthScheme

[AuthScheme](#)은 SDK가 호출자 자격 증명을 검색하여 작업 요청에 연결하는 워크플로를 정의하는 인터페이스입니다.

인증 체계는 아래에 자세히 설명된 다음 구성 요소를 사용합니다.

- 체계를 식별하는 고유 ID
- 서명 프로세스에 사용되는 호출자 자격 증명(예: AWS 자격 증명)을 반환하는 자격 증명 해석기
- 작업의 전송 요청에 호출자 자격 증명을 실제로 삽입하는 서명자(예: Authorization HTTP 헤더)

각 서비스 클라이언트 옵션에는 AuthSchemes 필드가 포함되며, 기본적으로 이 필드는 해당 서비스에서 지원하는 인증 체계 목록으로 채워집니다.

AuthSchemeResolver

각 서비스 클라이언트 옵션에는 AuthSchemeResolver 필드가 포함됩니다. 서비스별로 정의된 이 인터페이스는 각 작업에 대해 가능한 인증 옵션을 결정하기 위해 SDK에서 호출하는 API입니다.

Important

인증 체계 해석기는 사용되는 인증 체계를 지정하지 않습니다. 사용할 수 있는 체계 목록('옵션')을 반환하고 [여기](#)에 설명된 고정 알고리즘을 통해 최종 체계를 선택합니다.

옵션

ResolverAuthSchemes에 대한 직접 호출에서 반환되는 [옵션](#)은 가능한 인증 옵션을 나타냅니다.

옵션은 세 가지 정보 세트로 구성됩니다.

- 가능한 체계를 나타내는 ID
- 체계의 자격 증명 해석기에 제공할 불투명한 속성 세트
- 체계의 서명자에 제공할 불투명한 속성 세트

속성에 대한 참고 사항

사용 사례의 99%에서 호출자는 자격 증명 확인 및 서명을 위한 불투명한 속성에 대해 걱정할 필요가 없습니다. SDK는 각 체계에 필요한 속성을 가져와 SDK에 노출된 강력한 유형의 인터페이스에 전달합니다. 예를 들어 서비스의 기본 인증 해석기는 서명 이름 및 리전에 대한 서명자 속성을 갖도록 SigV4 옵션을 인코딩합니다. 서명 이름 및 리전의 값은 SigV4를 선택할 때 클라이언트의 구성된 [v4.HTTPSigner](#) 구현으로 전달됩니다.

자격 증명

[자격 증명](#)은 SDK 호출자를 추상적으로 표현한 것입니다.

SDK에 사용되는 가장 일반적인 유형의 자격 증명은 `aws.Credentials` 집합입니다. 대부분의 사용 사례에서 호출자는 Identity를 추상화로 간주할 필요가 없으며 구체적인 유형으로 직접 작업할 수 있습니다.

Note

이전 버전과의 호환성을 유지하고 API 혼동을 방지하기 위해 AWS SDK별 자격 증명 유형 `aws.Credentials`는 Identity 인터페이스를 직접 충족하지 않습니다. 이 매핑은 내부적으로 처리됩니다.

IdentityResolver

[IdentityResolver](#)는 Identity가 검색되는 인터페이스입니다.

IdentityResolver의 구체적인 버전은 강력한 형식(예: [aws.CredentialsProvider](#))으로 SDK에 존재하며, SDK는 이 매핑을 내부적으로 처리합니다.

호출자는 외부 인증 체계를 정의할 때 IdentityResolver 인터페이스를 직접 구현하기만 하면 됩니다.

Signer

[서명자](#)는 검색된 호출자 Identity로 요청이 보완되는 인터페이스입니다.

Signer의 구체적인 버전은 강력한 형식(예: [v4.HTTPSigner](#))으로 SDK에 존재하며, SDK는 이 매핑을 내부적으로 처리합니다.

호출자는 외부 인증 체계를 정의할 때 Signer 인터페이스를 직접 구현하기만 하면 됩니다.

AuthResolverParameters

각 서비스는 각 서비스 패키지에 AuthResolverParameters로 정의된 확인 함수로 전달되는 특정 입력 세트를 가져옵니다.

기본 해석기 파라미터는 다음과 같습니다.

name	유형	설명
Operation	string	간접적으로 호출 중인 작업의 이름입니다.
Region	string	클라이언트의 AWS 리전입니다. SigV4[A]를 사용하는 서비스에만 표시됩니다.

자체 해석기를 구현하는 경우 파라미터의 자체 인스턴스를 구성할 필요가 없습니다. SDK는 요청당 이러한 값을 소싱하여 구현에 전달합니다.

인증 체계 해결 워크플로

SDK를 통해 AWS 서비스 작업을 직접 호출하면 요청이 직렬화된 후 다음과 같은 일련의 작업이 수행됩니다.

1. SDK는 필요에 따라 입력 파라미터를 소싱하여 클라이언트의 `AuthSchemeResolver.ResolveAuthSchemes()` API를 직접 호출하여 작업에 사용할 수 있는 [옵션](#) 목록을 가져옵니다.
2. SDK는 해당 목록을 반복하고 다음 조건을 충족하는 첫 번째 체계를 선택합니다.
 - ID가 일치하는 체계가 클라이언트 자체 AuthSchemes 목록에 있습니다.
 - 체계의 자격 증명 해석기가 클라이언트의 옵션에 있음(`NotNil`)(체계의 `GetIdentityResolver` 메서드를 통해 확인됨, 위에서 설명한 구체적인 자격 증명 해석기 유형에 대한 매핑은 내부적으로 처리됨)(1)
3. 실행 가능한 체계가 선택되었다고 가정하면 SDK는 `GetIdentityResolver()` API를 간접적으로 호출하여 호출자의 자격 증명을 검색합니다. 예를 들어 내장 SigV4 인증 체계는 내부적으로 클라이언트의 `Credentials` 공급자에 매핑됩니다.
4. SDK는 자격 증명 해석기의 `GetIdentity()`를 호출합니다(예: SigV4용 `aws.CredentialProvider.Retrieve()`).

5. SDK는 엔드포인트 해석기의 `ResolveEndpoint()`를 직접 호출하여 요청에 대한 엔드포인트를 찾습니다. 엔드포인트에는 서명 프로세스에 영향을 미치는 추가 메타데이터(예: S3 Object Lambda의 고유한 서명 이름)가 포함될 수 있습니다.
6. SDK는 인증 체계의 `Signer()` API를 호출하여 서명자를 검색하고 `SignRequest()` API를 사용하여 이전에 검색한 호출자 자격 증명으로 요청에 서명합니다.

(1) SDK가 목록에서 익명 옵션(ID smithy.api#noAuth)을 발견하면 해당하는 자격 증명 해석기가 없으므로 자동으로 선택됩니다.

기본 지원 AuthScheme

다음 인증 체계는 기본적으로 AWS SDK for Go에서 지원됩니다.

명칭	체계 ID	자격 증명 해석기	Signer	Notes
SigV4	<code>aws.auth#sigv4</code>	aws.CredentialsProvider	v4.HTTPSigner	대부분의 AWS 서비스 작업에 대한 현재 기본값입니다.
SigV4A	<code>aws.auth#sigv4a</code>	<code>aws.CredentialsProvider</code>	해당 사항 없음	SigV4A 사용은 현재 제한되어 있으며 서명자 구현은 내부적입니다. HTTP 요청에 서명하기 위한 범용 API를 제공하는 새로운 옵트인 모듈 <code>aws-http-auth</code> 에 대한 이 공지 를 참조하세요.
SigV4Express	<code>com.amazonaws.s3#sigv4express</code>	s3.ExpressCredentialsProvider	<code>v4.HTTPSigner</code>	Express One Zone 에 사용됩니다.

명칭	체계 ID	자격 증명 해석기	Signer	Notes
HTTP 보유자	smithy.ap i#httpBearerAuth	smithybearer.Token Provider	smithybearer.Signer	codecatalyst 에서 사용합니다.
익명	smithy.ap i#noAuth	해당 사항 없음	해당 사항 없음	인증 없음 - 자격 증명이 필요하지 않으며 요청에 서 명되거나 인증되 지 않습니다.

자격 증명 구성

AWS SDK for Go에서 인증 체계의 자격 증명 구성 요소는 SDK 클라이언트 Options에서 구성됩니다. SDK는 작업이 호출될 때 선택한 체계에 대해 이러한 구성 요소의 값을 자동으로 선택하고 사용합니다.

Note

이전 버전과의 호환성을 위해 SDK는 자격 증명 해석기가 구성되지 않은 경우 익명 인증 체계의 사용을 암시적으로 허용합니다. 이는 클라이언트의 Options의 모든 자격 증명 해석기를 nil로 설정하여 수동으로 수행할 수 있습니다(sigv4 자격 증명 해석기는 `aws.AnonymousCredentials{}`로 설정할 수도 있음).

서명자 구성

AWS SDK for Go에서 인증 체계의 서명자 구성 요소는 SDK 클라이언트 Options에서 구성됩니다. SDK는 작업이 호출될 때 선택한 체계에 대해 이러한 구성 요소의 값을 자동으로 선택하고 사용합니다. 추가 구성은 필요하지 않습니다.

사용자 지정 인증 체계

사용자 지정 인증 체계를 정의하고 사용하도록 구성하려면 호출자가 다음을 수행해야 합니다.

1. [AuthScheme](#) 구현 정의
2. SDK 클라이언트 AuthSchemes 목록에 체계 등록

3. 해당하는 경우 SDK 클라이언트의 `AuthSchemeResolver`를 계측하여 체계의 ID로 인증 `Option`을 반환합니다.

⚠ Warning

다음 서비스에는 고유하거나 사용자 지정된 인증 동작이 있습니다. 사용자 지정 인증 동작이 필요한 경우 기본 구현에 위임하고 그에 따라 래핑하는 것이 좋습니다.

Service	Notes
S3	작업 입력에 따라 SigV4A 및 SigV4Express의 조건부 사용.
EventBridge	작업 입력에 따라 SigV4A의 조건부 사용.
Cognito	특정 작업은 익명 전용입니다.
SSO	특정 작업은 익명 전용입니다.
STS	특정 작업은 익명 전용입니다.

클라이언트 엔드포인트 구성

⚠ Warning

엔드포인트 확인은 고급 SDK 주제입니다. 이러한 설정을 변경하면 코드가 손상될 위험이 있습니다. 기본 설정은 프로덕션 환경 내 대부분의 사용자에게 적용 가능해야 합니다.

는 서비스에 사용할 사용자 지정 엔드포인트를 구성하는 기능을 AWS SDK for Go 제공합니다. 대부분의 경우는 기본 구성으로도 충분합니다. 사용자 지정 엔드포인트를 구성하면 서비스의 프리릴리스 버전 작업과 같은 추가 동작을 수행할 수 있습니다.

사용자 지정

SDK 내에는 엔드포인트 확인 구성의 두 가지 '버전'이 있습니다.

- 2023년 Q3에 릴리스되었으며 다음을 통해 구성된 v2:
 - EndpointResolverV2
 - BaseEndpoint
- SDK와 함께 릴리스되었으며 다음을 통해 구성된 v1:
 - EndpointResolver

v1 엔드포인트 확인을 사용 중인 경우, 최신 엔드포인트 관련 서비스 기능에 액세스하려면 v2로 마이그레이션하는 것이 좋습니다.

V2: EndpointResolverV2 + BaseEndpoint

확인 v2에서 EndpointResolverV2는 엔드포인트 확인이 발생하는 최종 메커니즘입니다. 해석기의 ResolveEndpoint 메서드는 SDK에서 수행하는 모든 요청에서 워크플로의 일부로 간접 호출됩니다. 해석기가 반환한 Endpoint의 호스트 이름은 요청 시 그대로 사용됩니다(단, 작업 직렬화기는 여전히 HTTP 경로에 추가할 수 있음).

확인 v2에는 서비스 인스턴스의 '기본' 호스트 이름을 지정하는 데 사용되는 추가 클라이언트 수준 구성인 BaseEndpoint가 포함되어 있습니다. 여기에 설정된 값은 확정적이지 않습니다. 최종 확인이 발생하면 궁극적으로 클라이언트의 EndpointResolverV2에 파라미터로 전달됩니다 (EndpointResolverV2 파라미터에 대한 자세한 내용 참조). 그러면 해석기 구현을 통해 해당 값을 검사하고 잠재적으로 수정하여 최종 엔드포인트를 결정할 수 있습니다.

예를 들어 BaseEndpoint를 지정한 클라이언트를 사용하여 주어진 버킷에 대해 S3 GetObject 요청을 수행하는 경우, 가상 호스트와 호환되면(클라이언트 구성에서 가상 호스팅을 비활성화하지 않은 경우) 기본 해석기가 버킷을 호스트 이름에 주입합니다.

실제로 BaseEndpoint는 클라이언트가 서비스의 개발 또는 미리 보기 인스턴스를 가리키는 데 사용될 가능성이 높습니다.

EndpointResolverV2 파라미터

각 서비스는 각 서비스 패키지에 EndpointParameters로 정의된 확인 함수로 전달되는 특정 입력 세트를 가져옵니다.

모든 서비스에는 AWS내에서 일반적인 엔드포인트 확인을 용이하게 하는 데 사용되는 다음의 기본 파라미터가 포함되어 있습니다.

이름	type	description
Region	string	클라이언트의 AWS 리전
Endpoint	string	클라이언트 구성에서 BaseEndpoint 에 대해 설정된 값
UseFips	bool	클라이언트 구성에서 FIPS 엔드포인트가 활성화되었는지 여부
UseDualStack	bool	클라이언트 구성에서 듀얼 스택 엔드포인트가 활성화되었는지 여부

서비스는 해결에 필요한 추가 파라미터를 지정할 수 있습니다. 예를 들어 S3의 EndpointParameters에는 버킷 이름 외에도 가상 호스트 주소 지정 활성화 여부와 같은 여러 S3 전용 기능 설정이 포함됩니다.

자체 EndpointResolverV2를 구현하는 경우 자체 EndpointParameters 인스턴스를 구성할 필요가 없습니다. SDK는 요청당 값을 소싱하여 구현에 전달합니다.

Amazon S3에 대한 참고 사항

Amazon S3는 버킷 가상 호스팅, S3 MRAP 등과 같은 복잡한 엔드포인트 사용자 지정을 통해 모델링된 많은 기능을 갖춘 복잡한 서비스입니다.

따라서 S3 클라이언트에서 EndpointResolverV2 구현을 교체하지 않는 것이 좋습니다. 추가적인 엔드포인트 고려 사항과 함께 로컬 개발 스택에 요청을 보내는 등 해결 동작을 확장해야 하는 경우, 기본 구현을 래핑하여 폴백의 형태로 기본값으로 위임하는 것이 좋습니다(아래 예시 참조).

예제

BaseEndpoint 포함

다음 코드 조각은 S3 클라이언트가 서비스의 로컬 인스턴스를 가리키도록 하는 방법을 보여줍니다. 이 예제에서는 포트 8080의 루프백 디바이스에서 호스팅됩니다.

```
client := s3.NewFromConfig(cfg, func (o *svc.Options) {
```

```
o.BaseEndpoint = aws.String("https://localhost:8080/")
})
```

EndpointResolverV2 포함

다음 코드 조각은 EndpointResolverV2를 사용하여 S3의 엔드포인트 확인에 사용자 지정 동작을 주입하는 방법을 보여줍니다.

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



```
    }
  })
}
```

둘 다 포함

다음 샘플 프로그램은 BaseEndpoint와 EndpointResolverV2 간의 상호 작용을 보여줍니다. 다음은 고급 사용 사례입니다.

```
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)
}
```

실행 시 위 프로그램은 다음을 출력합니다.

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

V1: EndpointResolver

Warning

엔드포인트 확인 v1은 이전 버전과의 호환성을 위해 유지되며 엔드포인트 확인 v2의 최신 동작과 분리됩니다. 호출자가 `EndpointResolver` 필드를 설정한 경우에만 사용됩니다.

v1을 사용하면 v2 확인 릴리스 시 또는 릴리스 후에 도입된 엔드포인트 관련 서비스 기능에 액세스하지 못할 가능성이 높습니다. 업그레이드 방법에 대한 지침은 '마이그레이션'을 참조하세요.

서비스 클라이언트에 대한 사용자 지정 엔드포인트 확인 로직을 제공하도록 [EndpointResolver](#)를 구성할 수 있습니다. 사용자 지정 엔드포인트 해석기를 사용하여 모든 엔드포인트 또는 특정 리전 엔드포인트에 대한 서비스의 엔드포인트 해석 로직을 재정의할 수 있습니다. 사용자 지정 엔드포인트 해석기는 사용자 지정 해석기가 요청된 엔드포인트를 확인하지 않으려는 경우 서비스의 엔드포인트 확인 로직을 대체하도록 트리거할 수 있습니다. [EndpointResolverWithOptionsFunc](#)를 사용하면 `EndpointResolverWithOptions` 인터페이스를 충족하는 데 필요한 함수를 쉽게 래핑할 수 있습니다.

[WithEndpointResolverWithOptions](#)로 래핑된 해석기를 [LoadDefaultConfig](#)에 전달하여 `EndpointResolver`를 쉽게 구성할 수 있으므로 자격 증명을 로드할 때 엔드포인트를 재정의하고 사용자 지정 엔드포인트 해석기로 결과 `aws.Config`를 구성할 수 있습니다.

엔드포인트 해석기에는 서비스와 리전이 문자열로 제공되므로 해석기가 동적으로 동작을 구동할 수 있습니다. 각 서비스 클라이언트 패키지에는 엔드포인트 해석기를 간접적으로 호출하는 서비스 클라이언트를 결정하는 데 사용할 수 있는 내보낸 `ServiceID` 상수가 있습니다.

엔드포인트 해석기는 [EndpointNotFoundError](#) 감시 오류 값을 사용하여 서비스 클라이언트 기본 해결 로직에 대한 폴백 확인을 트리거할 수 있습니다. 이렇게 하면 폴백 로직을 처리할 필요 없이 하나 이상의 엔드포인트를 원활하게 선택적으로 재정의할 수 있습니다.

엔드포인트 해석기 구현에서 `EndpointNotFoundError` 이외의 오류가 반환되면 엔드포인트 확인이 중지되고 서비스 작업은 애플리케이션에 오류를 반환합니다.

예제

폴백 포함

다음 코드 조각은 다른 엔드포인트에 대한 폴백 동작을 사용하여 DynamoDB에 단일 서비스 엔드포인트를 재정의하는 방법을 보여줍니다.

```
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))
```

폴백 없음

다음 코드 조각은 다른 엔드포인트에 대한 폴백 동작을 사용하지 않고 DynamoDB에 단일 서비스 엔드포인트를 재정의하는 방법을 보여줍니다.

```
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))
```

변경 불가능한 엔드포인트

Warning

엔드포인트를 변경 불가능으로 설정하면 일부 서비스 클라이언트 기능이 올바르게 작동하지 않게 되어 정의되지 않은 동작이 발생할 수 있습니다. 엔드포인트를 변경 불가능으로 정의할 때는 주의해야 합니다.

Amazon S3와 같은 일부 서비스 클라이언트는 특정 서비스 작업을 위해 해석기가 반환한 엔드포인트를 수정할 수 있습니다. 예를 들어 Amazon S3는 확인된 엔드포인트를 변경하여 [가상 버킷 주소 지정](#)을 자동으로 처리합니다. [HostnameImmutable](#)을 true로 설정하여 SDK가 사용자 지정 엔드포인트를 변경하지 못하도록 할 수 있습니다. 예제:

```
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))
```

마이그레이션

엔드포인트 확인의 v1에서 v2로 마이그레이션할 때 다음과 같은 일반 원칙이 적용됩니다.

- [HostnameImmutable](#)이 false로 설정된 [엔드포인트](#)를 반환하는 것은 BaseEndpoint를 v1에서 원래 반환된 URL로 설정하고 EndpointResolverV2를 기본값으로 두는 것과 거의 동일합니다.

- HostnameImmutable이 true로 설정된 엔드포인트를 반환하는 것은 v1에서 원래 반환된 URL을 반환하는 EndpointResolverV2를 구현하는 것과 거의 동일합니다.
- 기본 예외는 모델링된 엔드포인트 접두사가 있는 작업을 위한 것입니다. 이에 대한 자세한 내용은 아래에 나와 있습니다.

이러한 사례에 대한 예제는 아래에 제시되어 있습니다.

Warning

V1 변경 불가능 엔드포인트와 V2 확인은 동작이 동일하지 않습니다. 예를 들어 S3 객체 Lambda와 같은 사용자 지정 기능에 대한 서명 재정의는 v1 코드를 통해 반환되는 변경 불가능 엔드포인트에 여전히 설정되지만 v2에 대해서는 동일하게 수행되지 않습니다.

호스트 접두사에 대한 참고 사항

일부 작업은 확인된 엔드포인트 앞에 추가할 호스트 접두사로 모델링됩니다. 이 동작은 ResolveEndpointV2의 출력과 함께 작동해야 하므로 호스트 접두사는 해당 결과에 계속 적용됩니다.

미들웨어를 적용하여 엔드포인트 호스트 접두사를 수동으로 비활성화할 수 있습니다. 예제 섹션을 참조하세요.

예제

변경 가능한 엔드포인트

다음 코드 샘플은 수정 가능한 엔드포인트를 반환하는 기본 v1 엔드포인트 해석기를 마이그레이션하는 방법을 보여줍니다.

```
// 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/")
})
```

```
})
```

변경 불가능한 엔드포인트

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

호스트 접두사 비활성화

```
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)
    }
}
```

HTTP 클라이언트 사용자 지정

AWS SDK for Go는 기본 구성 값과 함께 기본 HTTP 클라이언트를 사용합니다. 이러한 구성 값 중 일부를 변경할 수 있지만 기본 HTTP 클라이언트 및 전송은 처리량이 높고 지연 시간이 짧은 환경에서 AWS SDK for Go를 사용하는 고객을 위해 충분히 구성되지 않습니다. 구성 권장 사항은 특정 워크로드에 따라 다르므로 자세한 내용은 [FAQ](#) 섹션을 참조하세요. 이 섹션에서는 사용자 지정 HTTP 클라이언트를 구성하고 해당 클라이언트를 사용하여 AWS SDK for Go 직접 호출을 생성하는 방법을 설명합니다.

사용자 지정 HTTP 클라이언트를 생성하는 데 도움이 되도록 이 섹션에서는 [NewBuildableClient](#)가 사용자 지정 설정을 구성하고 해당 클라이언트를 AWS SDK for Go 서비스 클라이언트와 함께 사용하는 방법을 설명합니다.

사용자 지정하려는 항목을 정의해 보겠습니다.

구성 로드 중 재정의

[LoadDefaultConfig](#)를 직접적으로 호출할 때 [WithHTTPClient](#)를 사용하여 클라이언트를 래핑하고 결과 값을 [LoadDefaultConfig](#)에 전달하여 사용자 지정 HTTP 클라이언트를 제공할 수 있습니다. 예를 들어 `customClient`를 클라이언트로 전달하려면 다음을 수행합니다.


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

제한 시간

요청 제한 시간으로 `BuildableHTTPClient`를 구성할 수 있습니다. 이 제한 시간에는 연결, 리디렉션 처리 및 전체 응답 본문 읽기 시간이 포함됩니다. 예를 들어 클라이언트 제한 시간을 수정하려면 다음을 수행합니다.

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

// ...

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

다이얼러

`BuildableHTTPClient`는 수정된 [다이얼러](#) 옵션으로 클라이언트를 구성하기 위한 빌더 메커니즘을 제공합니다. 다음 예제에서는 클라이언트 `Dialer` 설정을 구성하는 방법을 보여줍니다.

```
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
})
```

설정

Dialer.KeepAlive

이 설정은 활성 네트워크 연결의 연결 유지 기간을 나타냅니다.

연결 유지를 비활성화하려면 음수 값으로 설정합니다.

프로토콜 및 운영 체제에서 지원하는 경우 연결 유지를 활성화하려면 0으로 설정합니다.

연결 유지를 지원하지 않는 네트워크 프로토콜 또는 운영 체제는 이 필드를 무시합니다. 기본적으로 TCP는 연결 유지를 활성화합니다.

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

KeepAlive를 `time.Duration`으로 설정합니다.

`Dialer.Timeout`

이 설정은 다이얼이 연결이 생성되기를 기다리는 최대 시간을 나타냅니다.

기본값은 30초입니다.

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

Timeout를 `time.Duration`으로 설정합니다.

운송

`BuildableHTTPClient`는 수정된 [운송](#) 옵션으로 클라이언트를 구성하기 위한 빌더 메커니즘을 제공합니다.

프록시 구성

인터넷에 직접 연결할 수 없는 경우 Go 지원 환경 변수(`HTTP_PROXY/HTTPS_PROXY`)를 사용하거나 사용자 지정 HTTP 클라이언트를 생성하여 프록시를 구성할 수 있습니다. 다음 예제에서는 `PROXY_URL`을 프록시 엔드포인트로 사용하도록 클라이언트를 구성합니다.

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

// ...

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

기타 설정

다음은 HTTP 클라이언트를 조정하기 위해 수정할 수 있는 몇 가지 다른 Transport 설정입니다. 여기에 설명되지 않은 추가 설정은 [전송](#) 유형 설명서에서 확인할 수 있습니다. 다음 예제와 같이 이러한 설정을 적용할 수 있습니다.

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

// ...

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

Transport.ExpectContinueTimeout

요청에 "Expect: 100-continue" 헤더가 있는 경우 이 설정은 요청 헤더를 완전히 작성한 후 서버의 첫 번째 응답 헤더를 기다리는 최대 시간을 나타냅니다. 이 시간에는 요청 헤더를 보내는 시간이 포함되지 않습니다. HTTP 클라이언트는 이 제한 시간이 소진된 후 페이로드를 전송합니다.

기본값은 1초입니다.

제한 시간이 없으면 0으로 설정하고 대기 없이 요청 페이로드를 전송합니다. 한 가지 사용 사례는 나중에 표시된 함수에서 Amazon S3를 사용하는 것과 유사한 세션을 수행하는 프록시 또는 타사 서비스에 문제가 발생하는 경우입니다.

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

ExpectContinue를 time.Duration으로 설정합니다.

Transport.IdleConnTimeout

이 설정은 유휴 네트워크 연결을 HTTP 요청 간에 유지하는 최대 시간을 나타냅니다.

제한이 없는 경우 0으로 설정합니다.

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

IdleConnTimeout를 time.Duration으로 설정합니다.

Transport.MaxIdleConns

이 설정은 모든 호스트에서 유휴(keep-alive) 연결의 최대 수를 나타냅니다. 이 값을 늘리는 한 가지 사용 사례는 동일한 클라이언트에서 단기간에 많은 연결이 표시되는 경우입니다.

0은 제한이 없음을 의미합니다.

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

MaxIdleConns를 int로 설정합니다.

Transport.MaxIdleConnsPerHost

이 설정은 호스트당 유지할 유휴(keep-alive) 연결의 최대 수를 나타냅니다. 이 값을 늘리는 한 가지 사용 사례는 동일한 클라이언트에서 단기간에 많은 연결이 표시되는 경우입니다.

기본값은 호스트당 두 개의 유휴 연결입니다.

DefaultMaxIdleConnsPerHost를 사용하려면 0으로 설정합니다. (2)

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

MaxIdleConnsPerHost를 int로 설정합니다.

Transport.ResponseHeaderTimeout

이 설정은 클라이언트가 응답 헤더를 읽을 때까지 기다리는 최대 시간을 나타냅니다.

클라이언트가 이 기간 내에 응답의 헤더를 읽을 수 없는 경우 제한 시간 오류와 함께 요청이 실패합니다.

장기 실행 Lambda 함수를 사용할 때는 Lambda 함수가 완료되거나 시간 초과될 때까지 작업이 응답 헤더를 반환하지 않으므로 이 값을 신중하게 설정하세요. 그러나 **** InvokeAsync**** API 작업에서 이 옵션을 계속 사용할 수 있습니다.

기본값은 제한 시간 없음이며 영구적으로 기다립니다.

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

ResponseHeaderTimeout를 time.Duration으로 설정합니다.

Transport.TLSHandshakeTimeout

이 설정은 TLS 핸드셰이크가 완료될 때까지 대기하는 최대 시간을 나타냅니다.

기본값은 10초입니다.

0은 제한 시간이 없음을 의미합니다.

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

TLSHandshakeTimeout를 time.Duration으로 설정합니다.

HTTP 인터셉터

인터셉터를 사용하여 API 요청 및 응답 실행에 연결할 수 있습니다. 인터셉터는 SDK가 요청/응답 수명 주기에 동작을 주입하기 위해 작성하는 코드를 호출하는 개방형 메커니즘입니다. 이 방법으로 진행 중인 요청 수정, 요청 처리 디버그, 예외 보기 등의 작업을 수행할 수 있습니다.

인터셉터와 미들웨어 비교

AWS SDK for Go v2는 요청 처리를 사용자 지정하기 위한 인터셉터와 미들웨어를 모두 제공합니다. 둘 다 목적이 비슷하지만 다양한 대상과 사용 사례에 맞게 설계되었습니다.

- 인터셉터는 간단한 HTTP 중심 API를 사용하여 요청/응답 처리를 사용자 지정하려는 SDK 사용자를 위해 설계되었습니다. 요청 수명 주기에서 특정 후크 포인트를 제공하고 HTTP 요청 및 응답과 직접 작동합니다.
- 미들웨어는 전송에 구매받지 않는 고급 시스템으로, 주로 SDK에서 내부적으로 사용됩니다. 미들웨어는 강력하지만 SDK 내부 기능에 대한 심층적인 지식이 요구되며 더 복잡한 인터페이스가 필요합니다.

일반적인 사용 사례에서 미들웨어 대비 인터셉터의 주요 이점은 다음과 같습니다.

- HTTP 중심: 인터셉터는 HTTP 요청 및 응답에서 직접 작동하므로 미들웨어에 필요한 전송 유형을 확인할 필요가 없습니다.
- 더 간단한 인터페이스: 각 인터셉터 후크에는 일반 미들웨어 패턴이 아닌 특정 집중 인터페이스가 있습니다.
- 더 명확한 실행 모델: 인터셉터는 미들웨어 스택 주문에 대한 지식 없이 요청 수명 주기의 잘 정의된 지점에서 실행됩니다.

Note

인터셉터는 기존 미들웨어 시스템을 기반으로 구축되므로 둘 다 동일한 애플리케이션에서 공존할 수 있습니다. 미들웨어는 전송에 구매받지 않는 동작이나 복잡한 스택 조작이 필요한 고급 사용 사례에 계속 사용할 수 있습니다.

사용 가능한 인터셉터 후크

AWS SDK for Go v2는 요청 수명 주기의 다양한 단계에서 인터셉터 후크를 제공합니다. 각 후크는 구현할 수 있는 특정 인터페이스에 해당합니다.

- BeforeExecution - 작업 실행 중에 직접 호출된 첫 번째 후크
- BeforeSerialization - 입력 메시지가 전송 요청으로 직렬화되기 전
- AfterSerialization - 입력 메시지가 전송 요청으로 직렬화된 후
- BeforeRetryLoop - 재시도 루프를 입력하기 전
- BeforeAttempt - 재시도 루프 내에서 직접 호출된 첫 번째 후크
- BeforeSigning - 전송 요청에 서명하기 전
- AfterSigning - 전송 요청에 서명한 후
- BeforeTransmit - 전송 요청이 전송되기 전
- AfterTransmit - 전송 응답을 수신한 후
- BeforeDeserialization - 전송 응답이 역직렬화되기 전
- AfterDeserialization - 전송 응답의 마샬링 해제 후
- AfterAttempt - 재시도 루프 내에서 직접 호출된 마지막 후크
- AfterExecution - 작업 실행 중에 직접 호출된 마지막 후크

단일 인터셉터에서 여러 인터페이스를 구현하여 요청 수명 주기의 여러 단계에 연결할 수 있습니다.

인터셉터 등록

서비스 클라이언트를 구성하거나 특정 작업에 대한 구성을 재정의할 때 인터셉터를 등록합니다. 등록은 인터셉터를 클라이언트의 모든 작업에 적용할지 아니면 특정 작업에만 적용할지에 따라 달라집니다.

인터셉터는 인터셉터를 추가 및 제거하는 방법을 제공하는 인터셉터 레지스트리를 통해 관리됩니다. 다음 예제는 서명 프로세스 전에 발신 요청에 AWS X-Ray 트레이스 ID 헤더를 추가하는 간단한 인터셉터를 보여줍니다.

```
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{})
})
```

인터셉터 레지스트리는 클라이언트 옵션에 추가되어 작업별 인터셉터 구성을 활성화합니다.

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

글로벌 인터셉터 구성

각 인터셉터 유형에 적합한 `With*` 옵션과 함께 `config.LoadDefaultConfig` 함수를 사용하여 인터셉터를 전역적으로 등록할 수도 있습니다. 이렇게 하면 해당 구성에서 생성된 모든 AWS 서비스 클라이언트에 인터셉터가 적용됩니다.

```
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)

```

로깅

AWS SDK for Go에는 애플리케이션에서 요청 문제 또는 실패를 디버깅하고 진단하기 위한 디버깅 정보를 활성화할 수 있는 로깅 기능이 있습니다. [Logger](#) 인터페이스와 [ClientLogMode](#)는 클라이언트가 로깅하는 방법과 대상을 결정하는 데 사용할 수 있는 주요 구성 요소입니다.

Logger

[LoadDefaultConfig](#)를 사용하여 [Config](#)를 구성할 때 기본값 Logger는 프로세스의 표준 오류(stderr)로 로그 메시지를 보내도록 구성됩니다. [Logger](#) 인터페이스를 충족하는 사용자 지정 로거는 [config.WithLogger](#)로 래핑하여 LoadDefaultConfig에 인수로 전달할 수 있습니다.

예를 들어 applicationLogger를 사용하도록 클라이언트를 구성하려면 다음을 수행합니다.

```

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

```

이제 구성된 aws.Config를 사용하여 구성된 클라이언트가 applicationLogger에 로그 메시지를 보냅니다.

컨텍스트 인식 로거

Logger 구현은 선택적 [ContextLogger](#) 인터페이스를 구현할 수 있습니다. 이 인터페이스를 구현하는 로거는 현재 컨텍스트로 WithContext 메서드를 간접 호출합니다. 이를 통해 로깅 구현은 컨텍스트에 있는 값을 기반으로 추가 로깅 메타데이터를 작성할 수 있는 새 Logger를 반환할 수 있습니다.

ClientLogMode

기본적으로 서비스 클라이언트는 로그 메시지를 생성하지 않습니다. 디버깅을 위해 로그 메시지를 전송하도록 클라이언트를 구성하려면 Config의 [ClientLogMode](#) 멤버를 사용합니다. ClientLogMode는 다음에 대한 디버깅 메시지를 활성화하도록 설정할 수 있습니다.

- Signature Version 4(SigV4) 서명
- 요청 재시도 횟수
- HTTP 요청
- HTTP 응답

예를 들어 HTTP 요청 및 재시도 로깅을 활성화하려면 다음을 수행합니다.

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

사용 가능한 다양한 클라이언트 로그 모드는 [ClientLogMode](#)를 참조하세요.

재시도 및 제한 시간

를 AWS SDK for Go 사용하면 HTTP 서비스에 대한 요청의 재시도 동작을 구성할 수 있습니다. 기본적으로 서비스 클라이언트는 [retry.Standard](#)를 기본 재시도기로 사용합니다. 기본 구성 또는 동작이 애플리케이션 요구 사항을 충족하지 않는 경우 재시도기 구성을 조정하거나 자체 재시도기 구현을 제공할 수 있습니다.

는 구현을 위한 [재시도 구현에 필요한 메서드 세트를 정의하는 aws.Retryer](#) 인터페이스를 AWS SDK for Go 제공합니다. SDK는 재시도에 대해 [retry.Standard](#)와 [aws.NoOpRetryer](#)라는 두 가지 구현을 제공합니다.

표준 재시도기

[retry.Standard](#) 재시도기는 SDK 클라이언트에서 사용하는 기본 [aws.Retryer](#) 구현입니다. 표준 재시도기는 속도 제한 재시도기로 구성 가능한 최대 시도 횟수와 요청 백오프 정책을 조정할 수 있습니다.

다음 표에서는 이 재시도기의 기본값을 정의합니다.

속성	기본값
최대 시도 횟수	3
최대 백오프 지연	20초

요청을 간접 호출하는 동안 재시도 가능한 오류가 발생하면 표준 재시도자는 제공된 구성을 사용하여 요청을 지연시킨 다음 재시도합니다. 재시도는 요청의 전체 지연 시간에 추가되며 기본 구성이 애플리케이션 요구 사항을 충족하지 않는 경우 재시도를 구성해야 합니다.

표준 [재시도기](#) 구현에서 재시도 가능한 것으로 간주되는 오류에 대한 자세한 내용은 재시도 패키지 설명서를 참조하세요.

NopRetryer

[aws.NopRetryer](#)는 모든 재시도를 비활성화하려는 경우 제공되는 `aws.Retryer` 구현입니다. 서비스 클라이언트 작업을 간접적으로 호출할 때 이 재시도자는 요청을 한 번만 시도하도록 허용하고 결과로 발생하는 모든 오류는 호출 애플리케이션에 반환됩니다.

동작 사용자 지정

SDK는 `aws.Retryer` 구현을 래핑하는 헬퍼 유틸리티 세트를 제공하고 원하는 재시도 동작으로 래핑된 제공된 재시도를 반환합니다. 애플리케이션 요구 사항에 따라 모든 클라이언트, 클라이언트당 또는 작업당 기본 재시도를 재정의할 수 있습니다. 이를 수행하는 방법을 보여주는 추가 예제를 보려면 [재시도](#) 패키지 설명서 예제를 참조하세요.

Warning

`config.WithRetryer`를 사용하여 전역 `aws.Retryer` 구현을 지정하는 경우 `aws.Retryer` 각 간접 호출의 새 인스턴스를 반환해야 합니다. 이렇게 하면 모든 서비스 클라이언트에서 글로벌 재시도 토큰 버킷을 생성하지 않습니다.

최대 시도 횟수 제한

[retry.AddWithMaxAttempts](#)를 사용해 `aws.Retryer` 구현을 래핑하여 최대 시도 횟수를 원하는 값으로 설정합니다.

Warning

값이 0인 `retry.AddWithMaxAttempts`를 사용하면 요청이 성공하거나 재시도할 수 없는 오류가 반환될 때까지 SDK가 모든 재시도 가능한 오류를 재시도할 수 있습니다. SDK가 무한으로 재시도하도록 허용하면 런어웨이 워크로드와 팅창된 결제 주기가 발생할 수 있습니다.

```
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)
```

MaxAttempts를 직접 설정하는 기능 옵션을 사용하여 재시도를 인스턴스화하면 동작이 약간 다릅니다. 보다 구체적으로, 값을 0보다 작거나 같게 설정하면 재시도자가 무한히 재시도하지 않고 기본 최대 3회 시도를 사용합니다.

```
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)
```

최대 백오프 지연 제한

[retry.AddWithMaxBackoffDelay](#)를 사용하여 `aws.Retryer` 구현을 래핑하고 실패한 요청을 재시도할 때 발생할 수 있는 최대 백오프 지연을 제한합니다.

예를 들어 다음 코드를 사용하여 표준 클라이언트 재시도를 원하는 최대 지연 시간인 5초로 래핑할 수 있습니다.

```
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)
```

추가 API 오류 코드 재시도

[retry.AddWithErrorCodes](#)를 사용하여 `aws.Retryer` 구현을 래핑하고 재시도 가능한 것으로 간주되어야 하는 추가 API 오류 코드를 포함합니다.

예를 들어 다음 코드를 사용하여 표준 클라이언트 재시도를 래핑하여 Amazon S3 `NoSuchBucketException` 예외를 재시도 가능으로 포함할 수 있습니다.

```
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)

```

클라이언트 측 속도 제한

는 최신 SDK의 동작에 맞게 표준 재시도 정책에 새로운 클라이언트 측 속도 제한 메커니즘을 AWS SDK for Go 도입합니다. SDKs 이 동작은 재시도자의 [옵션](#)에 있는 [RateLimiter](#) 필드에 의해 제어됩니다.

RateLimiter는 용량이 설정된 토큰 버킷으로 작동하며, 작업 시도 실패 시 토큰이 사용됩니다. 사용 가능한 것보다 많은 토큰을 사용하려고 시도하면 [QuotaExceededError](#)에서 작업이 실패합니다.

기본 구현은 다음과 같이 파라미터화됩니다(각 설정을 수정하는 방법).

- 용량 500([NewTokenRateLimit](#)을 사용하여 StandardOptions에서 RateLimiter 값 설정)
- 제한 시간 비용으로 인한 재시도는 토큰 10개(StandardOptions에서 RetryTimeoutCost 설정)
- 다른 오류로 인한 재시도 비용은 토큰 5개(StandardOptions에서 RetryCost 설정)
- 첫 번째 시도에서 성공하는 작업은 토큰 1개 추가(StandardOptions에서 NoRetryIncrement 설정)
 - 두 번째 이상 시도에서 성공한 작업은 토큰을 다시 추가하지 않음

기본 동작이 애플리케이션의 요구 사항에 맞지 않는 경우 [ratelimit.None](#)으로 비활성화할 수 있습니다.

예: 수정된 속도 제한기

```

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
    })
}))
```

예: `ratelimit.None`을 사용한 속도 제한 없음

```
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
    })
}))
```

시간 초과

[컨텍스트](#) 패키지를 사용하여 서비스 클라이언트 작업을 간접적으로 호출할 때 제한 시간 또는 기한을 설정합니다. [context.WithDeadline](#)을 사용하여 애플리케이션 컨텍스트를 래핑하고 간접 호출된 작업을 완료해야 하는 특정 시간으로 기한을 설정합니다. 특정 `time.Duration` 사용 [context.WithTimeout](#) 이후에 제한 시간을 설정하려면 다음을 수행합니다. SDK는 서비스 API를 간접적으로 호출할 때 제공된

`context.Context`를 HTTP 전송 클라이언트에 전달합니다. 작업을 간접 호출하는 동안 SDK에 전달된 컨텍스트가 취소되었거나 취소될 예정이면 SDK는 요청을 더 이상 재시도하지 않고 호출 애플리케이션으로 돌아갑니다. SDK에 제공된 컨텍스트가 취소된 경우 애플리케이션에서 컨텍스트 취소를 적절하게 처리해야 합니다.

제한 시간 설정

다음 예제에서는 서비스 클라이언트 작업에 대한 제한 시간을 설정하는 방법을 보여줍니다.

```
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
}
```

AWS SDK for Go 사용

애플리케이션에서 AWS SDK for Go를 사용하여 일반적으로 권장되는 프로그래밍 방법을 알아봅니다.

주제

- [서비스 클라이언트 구성](#)
- [서비스 작업 직접 호출](#)
- [서비스 클라이언트 동시 사용](#)
- [작업 페이지네이터 사용](#)
- [웨이터 사용](#)
- [AWS SDK for Go V2에서 오류 처리](#)

서비스 클라이언트 구성

서비스 클라이언트는 서비스 클라이언트의 Go 패키지에서 사용할 수 있는 `New` 또는 `NewFromConfig` 함수를 사용하여 구성할 수 있습니다. 각 함수는 서비스 API를 간접 호출하는 메서드가 포함된 `Client` 구조체 유형을 반환합니다. `New` 및 `NewFromConfig` 각각은 서비스 클라이언트를 구성하기 위해 동일한 구성 가능한 옵션 세트를 제공하지만, 다음 섹션에서 살펴볼 약간 다른 구성 패턴을 제공합니다.

NewFromConfig

`NewFromConfig` 함수는 [aws.Config](#)를 사용하여 서비스 클라이언트를 구성하기 위한 일관된 인터페이스를 제공합니다. [config.LoadDefaultConfig](#)를 사용하여 `aws.Config`를 로드할 수 있습니다. `aws.Config` 구성에 대한 자세한 내용은 [SDK 구성](#) 섹션을 참조하세요. 다음 예제에서는 `aws.Config` 및 `NewFromConfig` 함수를 사용하여 Amazon S3 서비스 클라이언트를 구성하는 방법을 보여줍니다.

```
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)

```

구성 재정의

`NewFromConfig`는 클라이언트의 구성 `Options` 구조를 변경할 수 있는 하나 이상의 기능적 인수를 취할 수 있습니다. 이렇게 하면 리전 변경 또는 Amazon S3 UseAccelerate 옵션과 같은 서비스별 옵션 수정과 같은 특정 재정의의 수행할 수 있습니다. 예:

```

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
})

```

클라이언트 `Options` 값에 대한 재정의는 함수 인수가 `NewFromConfig`에 부여되는 순서에 따라 결정됩니다.

New

Note

`New`는 클라이언트 구성의 고급 형태로 간주됩니다. `aws.Config` 구조체를 사용하여 구성할 수 있으므로 클라이언트 구성에 `NewFromConfig`를 사용하는 것이 좋습니다. 이렇게 하면 애플리케이션에 필요한 각 서비스 클라이언트에 대한 `Options` 구조체 인스턴스를 구성할 필요가 없습니다.

New 함수는 클라이언트의 구성 옵션을 정의하기 위한 클라이언트 패키지 Options 구조만 사용하여 클라이언트를 구성하기 위한 인터페이스를 제공하는 클라이언트 생성자입니다. 예를 들어 New를 사용하여 Amazon S3 클라이언트를 구성하려면 다음을 수행합니다.

```
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,
"")),
})
```

구성 재정의

New는 클라이언트의 구성 Options 구조를 변경할 수 있는 하나 이상의 기능적 인수를 취할 수 있습니다. 이렇게 하면 리전 변경 또는 Amazon S3 UseAccelerate 옵션과 같은 서비스별 옵션 수정과 같은 특정 재정의의 수행할 수 있습니다. 예:

```
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
})
```

클라이언트 Options 값에 대한 재정의는 함수 인수가 New에 부여되는 순서에 따라 결정됩니다.

서비스 작업 직접 호출

서비스 클라이언트 인스턴스가 있으면 이를 사용하여 서비스 작업을 직접적으로 호출할 수 있습니다. 예를 들어 Amazon S3 GetObject 작업을 직접 호출하려면 다음을 수행합니다.

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

서비스 작업을 직접 호출하면 SDK는 입력을 동기적으로 검증하고, 요청을 직렬화하고, 자격 증명으로 서명하여 AWS로 전송한 다음 응답 또는 오류를 역직렬화합니다. 대부분의 경우 서비스 작업을 직접 호출할 수 있습니다. 각 서비스 작업 클라이언트 메서드는 작업 응답 구조와 오류 인터페이스 유형을 반환합니다. 서비스 작업의 응답 구조에 액세스하기 전에 항상 error 유형을 확인하여 오류가 발생했는지 확인해야 합니다.

파라미터를 서비스 작업에 전달

각 서비스 작업 메서드는 SDK에서 준수할 요청 기한을 설정하는 데 사용할 수 있는 [context.Context](#) 값을 사용합니다. 또한 각 서비스 작업은 서비스의 해당 Go 패키지에 있는 <OperationName>Input 구조체를 사용합니다. 작업 입력 구조를 사용하여 API 입력 파라미터를 전달합니다.

작업 입력 구조에는 표준 Go 숫자, 부울, 문자열, 맵 및 목록 유형과 같은 입력 파라미터가 있을 수 있습니다. 더 복잡한 API 작업에서는 서비스에 입력 파라미터 모델링이 더 복잡할 수 있습니다. 서비스별 구조 및 열거형 값과 같은 다른 유형은 서비스의 types Go 패키지에서 찾을 수 있습니다.

또한 서비스는 Go 유형의 기본값과 사용자가 값을 설정했는지 여부를 구별할 수 있습니다. 이러한 경우 입력 파라미터를 사용하려면 해당 유형에 대한 포인터 참조를 전달해야 할 수 있습니다. 숫자, 부울 및 문자열과 같은 표준 Go 유형의 경우 이 변환을 용이하게 하기 위해 [aws](#)에서 <Type> 및 From<Type> 편의 함수를 사용할 수 있습니다. 예를 들어 [aws.String](#)을 사용하여 string를 문자열에 대한 포인터가 필요한 입력 파라미터의 *string 유형으로 변환할 수 있습니다. 반대로 [aws.ToString](#)은 nil 포인터를 참조하지 않도록 보호하면서 *string을 string으로 변환하는 데 사용할 수 있습니다. To<Type> 함수는 서비스 응답을 처리할 때 유용합니다.

Amazon S3 클라이언트를 사용하여 GetObject API를 직접 호출하고 types 패키지 및 aws.<Type> 헬퍼를 사용하여 입력을 구성하는 방법의 예를 살펴보겠습니다.

```
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,
})
```

작업 직접 호출을 위한 클라이언트 옵션 재정의

기능 인수를 사용하여 클라이언트를 구성하는 동안 클라이언트 작업 옵션을 수정할 수 있는 방법과 마찬가지로, 서비스 작업 방법에 하나 이상의 기능 인수를 제공하여 작업 메서드를 직접 호출할 때 클라이언트 옵션을 수정할 수 있습니다. 이 작업은 동시성 면에서 안전하며 클라이언트의 다른 동시 작업에 영향을 주지 않습니다.

예를 들어 클라이언트 리전을 'us-west-2'에서 'us-east-1'로 재정의하려면 다음을 수행합니다.

```
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"
})
```

작업 응답 처리

각 서비스 작업에는 서비스의 작업 응답 멤버가 포함된 관련 출력 구조가 있습니다. 출력 구조는 다음 이름 지정 패턴 <OperationName>Output을 따릅니다. 일부 작업에는 작업 출력에 대해 정의된 멤버가 없을 수 있습니다. 서비스 작업을 직접 호출한 후에는 항상 반환 `error` 인수 유형을 확인하여 서비스 작업을 간접적으로 호출하는 동안 오류가 발생했는지 확인해야 합니다. 반환되는 오류는 클라이언트 측 입력 검증 오류부터 클라이언트에 반환되는 서비스 측 오류 응답까지 다양할 수 있습니다. 클라이언트가 `nil`이 아닌 오류를 반환하는 경우 작업의 출력 구조체에 액세스하면 안 됩니다.

예를 들어 작업 오류를 로깅하고 직접 호출 함수에서 조기에 반환하려면 다음을 수행합니다.

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

특정 오류 유형을 검사하는 방법을 포함하여 오류 처리에 대한 자세한 내용은 TODO를 참조하세요.

io.ReadCloser를 사용한 응답

일부 API 작업은 `io.ReadCloser`인 출력 멤버가 포함된 응답 구조를 반환합니다. 이는 HTTP 응답 자체의 본문에 출력의 일부 요소를 노출하는 API 작업이 됩니다.

예를 들어 Amazon S3 `GetObject` 작업은 `Body` 멤버가 객체 페이로드에 액세스하기 위한 `io.ReadCloser`인 응답을 반환합니다.

Warning

콘텐츠를 사용했는지 여부에 관계없이 항상 모든 `io.ReadCloser` 출력 멤버를 `Close()`해야 합니다. 이렇게 하지 않으면 리소스가 유출되고 향후 직접 호출된 작업에 대한 응답 본문을 읽는 데 문제가 발생할 수 있습니다.

```

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
}

```

응답 메타데이터

모든 서비스 작업 출력 구조에는 [미들웨어.메타데이터](#) 유형의 ResultMetadata 멤버가 포함됩니다. middleware.Metadata는 SDK 미들웨어에서 서비스에 의해 모델링되지 않은 서비스 응답의 추가 정보를 제공하는 데 사용됩니다. 여기에는 RequestID와 같은 메타데이터가 포함됩니다. 예를 들어, 서비스 응답과 연결된 RequestID를 검색하여 AWS Support가 요청 문제를 해결하는 데 도움이 되도록 하려면 다음을 수행합니다.

```

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)
```

서비스 클라이언트 동시 사용

동일한 서비스 클라이언트를 동시에 사용하여 여러 요청을 보내는 고루틴을 만들 수 있습니다. 서비스 클라이언트를 원하는 수만큼의 고루틴과 함께 사용할 수 있습니다.

다음 예제에서는 Amazon S3 서비스 클라이언트가 여러 고루틴에서 사용됩니다. 이 예제에서는 Amazon S3 버킷에 두 개의 객체를 동시에 업로드합니다.

```
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))
}

```

작업 페이지네이터 사용

일반적으로 항목 목록을 검색할 때 토큰 또는 마커의 출력 구조를 확인하여 AWS 서비스가 요청의 모든 결과를 반환했는지 확인해야 할 수 있습니다. 토큰 또는 마커가 있는 경우 이를 사용하여 결과의 다음 페이지를 요청합니다. 이러한 토큰 또는 마커를 관리하는 대신 서비스 패키지의 사용 가능한 페이지네이터 유형을 사용할 수 있습니다.

페이지네이터 헬퍼는 지원되는 서비스 작업에 사용할 수 있으며 서비스 클라이언트의 Go 패키지에서 찾을 수 있습니다. 지원되는 작업에 대한 페이지네이터를 구성하려면 `New<OperationName>Paginator` 함수를 사용합니다. 페이지네이터 구성 함수는 서비스 Client, 작업의 `<OperationName>Input` 입력 파라미터 및 다른 선택적 페이지네이터 설정을 구성할 수 있는 선택적 기능 인수 세트를 사용합니다.

반환된 작업 페이지네이터 유형은 마지막 페이지에 도달하거나 애플리케이션이 검색 중인 항목을 찾을 때까지 페이지가 지정된 작업을 반복할 수 있는 편리한 방법을 제공합니다. 페이지네이터 유형에는 HasMorePages와 NextPage의 두 가지 메서드가 있습니다. HasMorePages는 첫 번째 페이지를 검색하지 않았거나 작업을 사용하여 검색할 수 있는 추가 페이지가 있는 경우 true의 부울 값을 반환합니다. 작업의 첫 번째 또는 후속 페이지를 검색하려면 NextPage 작업을 직접 호출해야 합니다. NextPage는 context.Context를 가져오고 작업 출력과 해당 오류를 반환합니다. 클라이언트 작업 메서드 반환 파라미터와 마찬가지로 반환된 응답 구조를 사용하기 전에 항상 반환 오류를 확인해야 합니다. [작업 응답 처리](#)을(를) 참조하세요.

다음 예제에서는 ListObjectsV2 페이지네이터를 사용하여 ListObjectV2 작업에서 최대 3페이지의 객체 키를 나열합니다. 각 페이지는 Limit 페이지네이터 옵션으로 정의되는 최대 10개의 키로 구성됩니다.

```
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++
}

```

클라이언트 작업 메서드와 마찬가지로 NextPage에 하나 이상의 기능적 인수를 제공하여 요청 리전과 같은 클라이언트 옵션을 수정할 수 있습니다. 작업을 직접 호출할 때 클라이언트 옵션을 재정의하는 방법에 대한 자세한 내용은 [작업 직접 호출을 위한 클라이언트 옵션 재정의](#) 섹션을 참조하세요.

웨이터 사용

비동기식 AWS API와 상호 작용하는 경우에는 특정 리소스를 사용할 수 있을 때까지 기다려야 추가 작업을 수행할 수 있습니다.

예를 들어 Amazon DynamoDB CreateTable API는 CREATING의 TableStatus를 즉시 반환하며 테이블 상태가 ACTIVE로 전환될 때까지 읽기 또는 쓰기 작업을 간접 호출할 수 없습니다.

테이블 상태를 지속적으로 폴링하는 로직을 작성하는 것은 번거로우며 오류가 발생하기 쉽습니다. 웨이터는 복잡성을 없애는 데 도움이 되며 폴링 작업을 처리하는 간단한 API입니다.

예를 들어, 웨이터를 사용하여 DynamoDB 테이블이 생성되었고 쓰기 작업을 수행할 준비가 되었는지 폴링할 수 있습니다.

```

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")

```

웨이터 구성 재정의

기본적으로 SDK는 다양한 API에 대해 AWS 서비스에서 정의한 최적의 값으로 구성된 최소 지연 및 최대 지연 값을 사용합니다. 웨이터 구성 중에 또는 웨이터 작업을 간접 호출할 때 기능 옵션을 제공하여 웨이터 구성을 재정의할 수 있습니다.

예를 들어, 웨이터 구성 중에 웨이터 구성을 재정의하려면

```

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
})
```

각 웨이터의 Wait 함수는 기능 옵션도 사용합니다. 위 예제와 마찬가지로 Wait 요청당 웨이터 구성을 재정의할 수 있습니다.

```
// 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")

```

고급 웨이터 구성 재정의

사용자 지정 재시도 가능 함수를 제공하여 웨이터 기본 동작을 추가로 사용자 지정할 수 있습니다. 웨이터별 옵션은 [작업 미들웨어를 사용자 지정](#)하기 위한 `APIOptions`도 제공합니다.

예를 들어 고급 웨이터 재정의의 구성합니다.

```

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
})

```

AWS SDK for Go V2에서 오류 처리

AWS SDK for Go는 Go error 인터페이스 유형을 충족하는 오류를 반환합니다. `Error()` 메서드를 사용하여 특별한 처리 없이 SDK 오류 메시지의 형식이 지정된 문자열을 가져올 수 있습니다. SDK에서 반환한 오류는 `Unwrap` 메서드를 구현할 수 있습니다. SDK는 이 `Unwrap` 메서드를 사용하여 오류에 대한 추가 컨텍스트 정보를 제공하는 동시에 기본 오류 또는 오류 체인에 대한 액세스를 제공합니다. 언래핑 오류 체인을 처리하려면 `Unwrap` 메서드를 [errors.As](#)와 함께 사용해야 합니다.

애플리케이션이 `error` 인터페이스 유형을 반환할 수 있는 함수 또는 메서드를 간접 호출한 후 오류가 발생했는지 확인하는 것이 중요합니다. 오류 처리의 가장 기본적인 형태는 다음 예와 유사합니다.

```

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

```

오류 로깅

가장 간단한 형태의 오류 처리는 일반적으로 애플리케이션을 반환하거나 종료하기 전에 오류 메시지를 로깅하거나 인쇄하는 것입니다.

```

import "log"

// ...

```

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

서비스 클라이언트 오류

SDK는 서비스 클라이언트가 반환한 모든 오류를 [smithy.OperationError](#) 오류 유형으로 래핑합니다. `OperationError`는 기본 오류와 연결된 서비스 이름 및 작업에 대한 컨텍스트 정보를 제공합니다. 이 정보는 중앙 집중식 오류 처리 메커니즘을 사용하여 하나 이상의 서비스에 대해 작업 배치를 수행하는 애플리케이션에 유용할 수 있습니다. 애플리케이션은 `errors.As`를 사용하여 이 `OperationError` 메타데이터에 액세스할 수 있습니다.

```
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
}
```

API 오류 응답

서비스 작업은 모델링된 오류 유형을 반환하여 특정 오류를 나타낼 수 있습니다. 이러한 모델링된 유형을 `errors.As`와 함께 사용하여 언래핑하고 작업 실패가 특정 오류로 인한 것인지 확인할 수 있습니다. 예를 들어 동일한 이름의 버킷이 이미 있는 경우 Amazon S3 `CreateBucket`에서 [BucketAlreadyExists](#) 오류를 반환할 수 있습니다.

예를 들어 `BucketAlreadyExists` 오류가 오류인지 확인하려면 다음을 수행합니다.

```
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
}

```

모든 서비스 API 응답 오류는 [smithy.APIError](#) 인터페이스 유형을 구현합니다. 이 인터페이스는 모델링된 서비스 오류 응답과 모델링되지 않은 서비스 오류 응답을 모두 처리하는 데 사용할 수 있습니다. 이 유형은 서비스에서 반환한 오류 코드 및 메시지에 대한 액세스를 제공합니다. 또한 이 유형은 오류의 결함이 알려진 경우 클라이언트 또는 서버로 인한 것인지 여부를 나타냅니다.

```

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
}

```

요청 식별자 검색

AWS Support에서 작업할 때 문제를 해결하려는 요청을 식별하는 요청 식별자를 제공하라는 메시지가 표시될 수 있습니다. [http.ResponseError](#)를 사용하고 `ServiceRequestID()` 메서드를 사용하여 오류 응답과 연결된 요청 식별자를 검색할 수 있습니다.

```

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
}
```

Amazon S3 요청 식별자

Amazon S3 요청에는 AWS Support에서 요청 문제 해결을 지원하는 데 사용할 수 있는 추가 식별자가 포함되어 있습니다. [s3.ResponseError](#)를 사용하고 `ServiceRequestID()` 및 `ServiceHostID()`를 직접 호출하여 요청 ID와 호스트 ID를 검색할 수 있습니다.

```
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
}
```

AWS SDK for Go v2를 AWS 서비스와 함께 사용

AWS 서비스를 직접적으로 호출하려면 먼저 서비스 클라이언트 인스턴스를 구성해야 합니다. 서비스 클라이언트는 해당 서비스의 모든 API 작업에 대한 하위 수준의 액세스를 제공합니다. 예를 들어 Amazon S3 API 직접 호출을 수행하기 위해 Amazon S3 서비스 클라이언트를 생성합니다.

서비스 작업을 직접 호출할 때 입력 파라미터를 구조체로 전달합니다. 직접 호출이 성공하면 서비스 API 응답이 포함된 출력 구조체가 생성됩니다. 예를 들어 Amazon S3 버킷 생성 작업을 성공적으로 직접 호출한 후 작업은 버킷의 위치와 함께 출력 구조체를 반환합니다.

메서드 및 파라미터를 포함한 서비스 클라이언트 목록은 [AWS SDK for Go API 참조](#)를 참조하세요.

체크섬을 통한 데이터 무결성 보호

Amazon Simple Storage Service(S3)는 객체를 업로드할 때 체크섬을 지정하는 기능을 제공합니다. 체크섬을 지정하면 객체와 함께 저장되며 객체를 다운로드할 때 유효성을 검사할 수 있습니다.

체크섬은 파일을 전송할 때 데이터 무결성을 한층 더 강화합니다. 체크섬을 사용하면 수신된 파일이 원본 파일과 일치하는지 확인하여 데이터 일관성을 확인할 수 있습니다. Amazon S3의 체크섬에 대한 자세한 내용은 [지원되는 알고리즘](#)이 포함된 [Amazon Simple Storage Service 사용 설명서](#)를 참조하세요.

필요에 가장 적합한 알고리즘을 유연하게 선택하고 SDK가 체크섬을 계산하도록 할 수 있습니다. 또는 지원되는 알고리즘 중 하나를 사용하여 미리 계산된 체크섬 값을 제공할 수 있습니다.

Note

[Amazon S3 모듈의 버전 v1.74.1](#)부터 SDK는 업로드에 대한 CRC32 체크섬을 자동으로 계산하여 기본 무결성 보호를 제공합니다. 사전 계산된 체크섬 값을 제공하지 않거나 SDK가 체크섬을 계산하는 데 사용해야 하는 알고리즘을 지정하지 않은 경우 SDK는 이 체크섬을 계산합니다.

또한 SDK는 외부에서 설정할 수 있는 데이터 무결성 보호에 대한 전역 설정을 제공하며, 이에 대한 자세한 내용은 [AWS SDK 및 도구 참조 안내서](#)에서 확인할 수 있습니다.

체크섬은 객체 업로드와 객체 다운로드라는 두 가지 요청 단계로 설명합니다.

객체 업로드

putObject 메서드를 사용하여 객체를 업로드하고 체크섬 알고리즘을 제공하면 SDK는 지정된 알고리즘에 대한 체크섬을 계산합니다.

다음 코드 스니펫은 CRC32 체크섬을 포함하여 객체를 업로드하라는 요청을 보여줍니다. SDK는 요청을 보낼 때 CRC32 체크섬을 계산하고 객체를 업로드합니다. Amazon S3는 체크섬을 계산하고 SDK에서 제공하는 체크섬과 비교하여 콘텐츠의 무결성을 검증합니다. 그런 다음 Amazon S3는 객체와 함께 체크섬을 저장합니다.

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

요청 시 체크섬 알고리즘을 제공하지 않으면 다음 표에 표시된 대로 사용 중인 SDK 버전에 따라 체크섬 동작이 달라집니다.

체크섬 알고리즘이 제공되지 않은 경우 체크섬 동작

AWS SDK for Go의 Amazon S3 모듈 버전	체크섬 동작
v1.74.1 이하	SDK에서 CRC 기반 체크섬을 자동으로 계산해서 요청에 제공하지 않습니다.
v1.74.1 이상	SDK에서 CRC32 알고리즘을 사용하여 체크섬을 계산하고 요청에 제공합니다. Amazon S3는 자체 CRC32 체크섬을 계산하여 전송의 무결성을 검증하고 이를 SDK에서 제공하는 체크섬과 비교합니다. 체크섬이 일치하면 체크섬이 객체와 함께 저장됩니다.

미리 계산된 체크섬 값 사용

요청과 함께 제공되는 사전 계산된 체크섬 값은 SDK의 자동 계산을 비활성화하고 제공된 값을 대신 사용합니다.

다음 예제는 사전 계산된 SHA256 체크섬을 포함하는 요청을 보여줍니다.

```
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"),
})
```

Amazon S3에서 체크섬 값이 지정된 알고리즘에 대해 올바르지 않다고 판단하면 서비스는 오류 응답을 반환합니다.

멀티파트 업로드

멀티파트 업로드에 체크섬을 사용할 수도 있습니다.

AWS SDK for Go는 멀티파트 업로드와 함께 체크섬을 사용할 수 있는 두 가지 옵션을 제공합니다. 첫 번째 옵션은 업로드용 CRC32 알고리즘을 지정하는 전송 관리자를 사용합니다.

```
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,
})
```

전송 관리자를 사용하여 업로드할 때 체크섬 알고리즘을 제공하지 않으면 SDK는 CRC32 알고리즘을 기반으로 체크섬을 자동으로 계산합니다. SDK는 모든 버전의 SDK에 대해 이 계산을 수행합니다.

두 번째 옵션은 [Amazon S3](#) 클라이언트를 사용하여 멀티파트 업로드를 수행합니다. 이 접근법으로 추가 체크섬을 지정하는 경우 업로드를 시작할 때 사용할 알고리즘을 지정해야 합니다. 또한 각 파트 요청에 대한 알고리즘을 지정하고 업로드 후 각 파트에 대해 계산된 체크섬을 제공해야 합니다.

```
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)
    }

```

객체 다운로드

[GetObject](#) 메서드를 사용하여 객체를 다운로드하면 `GetObjectInput`의 `ChecksumMode` 필드가 `types.ChecksumModeEnabled`로 설정될 때 SDK가 체크섬을 자동으로 검증합니다.

다음 스니펫의 요청은 체크섬을 계산하고 값을 비교하여 응답의 체크섬을 검증하도록 SDK에 지시합니다.

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

체크섬과 함께 객체를 업로드하지 않은 경우 검증이 수행되지 않습니다.

AWS SDK for Go 유틸리티 사용

AWS SDK for Go에는 AWS 서비스를 보다 쉽게 사용할 수 있도록 다음과 같은 유틸리티가 포함되어 있습니다. 관련 AWS 서비스 패키지에서 SDK 유틸리티를 찾습니다.

Amazon RDS 유틸리티

[IAM Authentication]

[인증](#) 패키지는 Amazon RDS MySQL 및 PostgreSQL 데이터베이스 인스턴스에 연결하기 위한 인증 토큰을 생성하는 유틸리티를 제공합니다. [BuildAuthToken](#) 메서드를 사용하면 데이터베이스 엔드포인트, AWS 리전, 사용자 이름 및 [aws.CredentialProvider](#) 구현을 제공함으로써 IAM 데이터베이스 인증을 통해 데이터베이스에 연결할 권한이 있는 IAM 자격 증명을 반환하는 데이터베이스 인증 토큰을 생성할 수 있습니다. IAM 인증으로 Amazon RDS를 구성하는 방법에 대한 자세한 내용은 다음 Amazon RDS 개발자 안내서 리소스를 참조하세요.

- [IAM 데이터베이스 인증의 활성화 및 비활성화](#)
- [IAM 데이터베이스 액세스를 위한 IAM 정책 생성 및 사용](#)
- [IAM 인증을 사용하여 데이터베이스 계정 생성](#)

다음 예제에서는 Amazon RDS 데이터베이스에 연결하기 위해 인증 토큰을 생성하는 방법을 보여줍니다.

```
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())
}

```

Amazon CloudFront 유틸리티

Amazon CloudFront URL 서명자

Amazon CloudFront URL 서명자는 서명된 URL의 생성 프로세스를 간소화합니다. 서명된 URL에는 콘텐츠에 대한 액세스를 제어할 수 있는 만료 날짜 및 시간과 같은 정보가 포함됩니다. 서명된 URL은 인터넷을 통해 콘텐츠를 배포하고 싶지만 특정 사용자(예: 요금을 지불한 사용자)로 액세스를 제한하려는 경우에 유용합니다.

URL에 서명하려면 CloudFront 키 페어 ID와 연결된 프라이빗 키를 사용하여 URLSigner 인스턴스를 생성합니다. 그런 다음 Sign 또는 SignWithPolicy 메서드를 직접 호출하고 서명할 URL을 포함합니다. Amazon CloudFront 키 페어에 대한 자세한 내용은 CloudFront 개발자 안내서의 [신뢰할 수 있는 서명자를 위한 CloudFront 키 페어 생성](#)을 참조하세요.

다음 예제에서는 생성된 후 1시간 동안 유효한 서명된 URL을 생성합니다.

```

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
}

```

서명 유틸리티에 대한 자세한 내용은 AWS SDK for Go API 참조의 [서명](#) 패키지를 참조하세요.

Amazon EC2 인스턴스 메타데이터 서비스

AWS SDK for Go를 사용하여 [Amazon EC2 인스턴스 메타데이터 서비스](#)에 액세스할 수 있습니다. [feature/ec2/imds](#) Go 패키지는 Amazon EC2 인스턴스 메타데이터 서비스에 액세스하는 데 사용할 수 있는 [클라이언트](#) 유형을 제공합니다. Client 및 관련 작업은 SDK에서 제공하는 다른 AWS 서비스 클라이언트와 유사하게 사용할 수 있습니다. SDK를 구성하고 서비스 클라이언트를 사용하는 방법에 대한 자세한 내용은 [SDK 구성](#) 및 [AWS SDK for Go v2를 AWS 서비스와 함께 사용](#) 섹션을 참조하세요.

클라이언트를 사용하면 AWS 리전 또는 로컬 IP 주소와 같이 애플리케이션이 실행되는 인스턴스에 대한 정보를 쉽게 검색할 수 있습니다. 일반적으로 인스턴스 메타데이터를 검색하려면 HTTP 요청을 생성하고 제출해야 합니다. 대신 `imds.Client`를 생성하여 다른 AWS 서비스와 같은 프로그래밍 방식 클라이언트를 통해 Amazon EC2 인스턴스 메타데이터 서비스에 액세스합니다.

예를 들어 클라이언트를 구성하는 방법은 다음과 같습니다.

```
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)
```

그런 다음 서비스 클라이언트를 사용하여 `local-ipv4`(인스턴스의 프라이빗 IP 주소)와 같은 메타데이터 범주에서 정보를 검색합니다.

```
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))
```

모든 메타데이터 범주 목록은 Amazon EC2 사용 설명서의 [인스턴스 메타데이터 범주](#)를 참조하세요.

Amazon S3 유틸리티

Amazon S3 Transfer Managers

Amazon S3 업로드 및 다운로드 관리자는 큰 객체를 분할할 수 있으므로 객체를 여러 파트로 병렬 전송할 수 있습니다. 이렇게 하면 중단된 전송을 쉽게 재개할 수 있습니다.

Amazon S3 Upload Manager

Amazon S3 Upload Manager는 파일을 더 작은 파트로 분할하고 병렬로 업로드할 수 있는지 여부를 결정합니다. 병렬 업로드 수와 업로드된 파트의 크기를 사용자 지정할 수 있습니다.

다음 예제에서는 Amazon S3 Uploader를 사용하여 파일을 업로드합니다. Uploader 사용 방법은 `s3.PutObject()` 작업과 비슷합니다.

```
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,
})
```

구성 옵션

[NewUploader](#)를 사용하여 Uploader 인스턴스를 인스턴스화할 때 여러 구성 옵션을 지정하여 객체 업로드 방법을 사용자 지정할 수 있습니다. 옵션은 NewUploader에 하나 이상의 인수를 제공하여 재정 의됩니다. 이 옵션에는 다음이 포함됩니다.

- **PartSize** - 업로드할 각 파트의 버퍼 크기를 바이트 단위로 지정합니다. 파트당 최소 크기는 5MiB 입니다.
- **Concurrency** - 병렬로 업로드할 파트의 수를 지정합니다.
- **LeavePartsOnError** - Amazon S3에 성공적으로 업로드된 파트를 남길지 여부를 나타냅니다.

Concurrency 값은 주어진 Upload 직접 호출에 대해 발생할 수 있는 동시 파트 업로드 수를 제한합니다. 이는 글로벌 클라이언트 동시성 제한이 아닙니다. PartSize 및 Concurrency 구성 값을 조정하여 최적의 구성을 찾습니다. 예를 들어, 연결 대역폭이 높은 시스템은 더 큰 파트와 더 많은 업로드를 병렬로 전송할 수 있습니다.

예를 들어, 애플리케이션에서 Uploader의 Concurrency 설정을 5로 구성합니다. 애플리케이션이 서로 다른 두 개의 고루틴에서 Upload를 직접 호출하면 결과는 10개의 동시 파트 업로드(고루틴 2 * Concurrency 5)입니다.

Warning

애플리케이션 리소스 소진을 방지하기 위해 애플리케이션은 동시 직접 호출을 Upload로 제한해야 합니다.

다음은 Uploader 생성 중에 기본 파트 크기를 설정하는 예제입니다.

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

Uploader 및 구성에 대한 자세한 내용은 AWS SDK for Go API 참조의 [업로더](#)를 참조하세요.

PutObjectInput 본문 필드(io.ReadSeeker와 io.Reader 비교)

s3.PutObjectInput 구조체의 Body 필드는 io.Reader 유형입니다. 그러나 이 필드를 io.ReadSeeker 및 io.ReaderAt 인터페이스를 모두 충족하는 유형으로 채우면 호스트 환경의 애

폴리케이션 리소스 사용률을 개선할 수 있습니다. 다음 예제에서는 두 인터페이스를 모두 충족하는 유형 `ReadSeekerAt`을 생성합니다.

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

`io.Reader` 유형의 경우 파트를 업로드하려면 먼저 리더의 바이트를 메모리에 버퍼링해야 합니다. `PartSize` 또는 `Concurrency` 값을 늘리면 `Uploader`에 필요한 메모리(RAM)가 크게 증가합니다. 필요한 메모리는 약 $PartSize * Concurrency$ 입니다. 예를 들어 `PartSize`에 100MB를 지정하고 `Concurrency`에 10을 지정하려면 최소 1GB가 필요합니다.

`io.Reader` 유형은 바이트를 읽기 전에 크기를 결정할 수 없으므로 `Uploader`는 업로드할 파트 수를 계산할 수 없습니다. 따라서 `PartSize`를 너무 낮게 설정하면 `Uploader`가 대용량 파일의 Amazon S3 업로드 한도인 10,000파트에 도달할 수 있습니다. 10,000개 이상의 파트를 업로드하려고 하면 업로드가 중지되고 오류가 반환됩니다.

`ReadSeekerAt` 유형을 구현하는 `body` 값의 경우, `Uploader`는 Amazon S3로 전송하기 전에 본문 콘텐츠를 메모리에 버퍼링하지 않습니다. `Uploader`는 파일을 Amazon S3에 업로드하기 전에 예상 파트 수를 계산합니다. `PartSize`의 현재 값에 파일을 업로드하는 데 10,000개 이상의 파트가 필요한 경우 `Uploader`는 필요한 파트 수가 적도록 파트 크기 값을 늘립니다.

실패한 업로드 처리

Amazon S3에 대한 업로드가 실패하면 기본적으로 `Uploader`는 Amazon S3 `AbortMultipartUpload` 작업을 사용하여 업로드된 파트를 제거합니다. 이 기능은 실패한 업로드가 Amazon S3 스토리지를 사용하지 않도록 합니다.

`Uploader`가 성공적으로 업로드된 파트를 삭제하지 않도록 `LeavePartsOnError`를 `true`로 설정할 수 있습니다. 이는 부분적으로 완료된 업로드를 재개하는 데 유용합니다. 업로드된 파트에서 작동하려면 실패한 업로드의 `UploadID`를 가져와야 합니다. 다음 예제에서는 `manager.MultiUploadFailure` 오류 인터페이스 유형을 사용하여 `UploadID`를 가져오는 방법을 보여줍니다.

```
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
}
}

```

업로드당 업로더 옵션 재정의

메서드에 하나 이상의 인수를 제공하여 Upload를 직접 호출할 때 Uploader 옵션을 재정의할 수 있습니다. 이러한 재정의는 동시성에 안전한 수정 사항이며 진행 중인 업로드 또는 후속 관리자 Upload 직접 호출에는 영향을 주지 않습니다. 예를 들어 특정 업로드 요청에 대한 PartSize 구성을 재정의하려면 다음을 수행합니다.

```

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
})

```

예제

Amazon S3로 폴더 업로드

다음 예제에서는 path/filepath 패키지를 사용하여 파일 목록을 재귀적으로 수집하고 지정된 Amazon S3 버킷에 업로드합니다. Amazon S3 객체의 키에는 파일의 상대 경로 접두사가 붙습니다.

```

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
}

```

Download Manager

Amazon S3 [Downloader](#) Manager는 파일을 더 작은 파트로 분할하고 병렬로 다운로드할 수 있는지 여부를 결정합니다. 병렬 다운로드 수와 다운로드된 파트의 크기를 사용자 지정할 수 있습니다.

예제: 파일 다운로드

다음 예제에서는 Amazon S3 Downloader를 사용하여 파일을 다운로드합니다. Downloader 사용 방법은 [s3.GetObject](#) 작업과 비슷합니다.

```

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 파라미터는 `io.WriterAt` 유형입니다. `WriterAt` 인터페이스를 사용하면 `Downloader`가 파일의 여러 파트를 병렬로 쓸 수 있습니다.

구성 옵션

`Downloader` 인스턴스를 인스턴스화할 때 구성 옵션을 지정하여 객체 다운로드 방법을 사용자 지정할 수 있습니다.

- `PartSize` - 다운로드할 각 파트의 버퍼 크기를 바이트 단위로 지정합니다. 파트당 최소 크기는 5MB입니다.
- `Concurrency` - 병렬로 다운로드할 파트의 수를 지정합니다.

`Concurrency` 값은 주어진 `Download` 직접 호출에 대해 발생할 수 있는 동시 파트 다운로드 수를 제한합니다. 이는 글로벌 클라이언트 동시성 제한이 아닙니다. `PartSize` 및 `Concurrency` 구성 값을 조정하여 최적의 구성을 찾습니다. 예를 들어, 연결 대역폭이 높은 시스템은 더 큰 파트와 더 많은 다운로드를 병렬로 수신할 수 있습니다.

예를 들어, 애플리케이션에서 `Downloader`의 `Concurrency`를 5로 구성합니다. 그러면 애플리케이션이 서로 다른 두 개의 고루틴에서 `Download`를 직접 호출하고 결과는 10개의 동시 파트 다운로드(고루틴 $2 * \text{Concurrency } 5$)가 됩니다.

⚠ Warning

애플리케이션 리소스 소진을 방지하기 위해 애플리케이션은 동시 직접 호출을 Download로 제한해야 합니다.

Downloader 및 기타 구성 옵션에 대한 자세한 내용은 AWS SDK for Go API 참조의 [manager.Downloader](#)를 참조하세요.

다운로드당 다운로더 옵션 재정의

메서드에 하나 이상의 함수 인수를 제공하여 Download를 직접 호출할 때 Downloader 옵션을 재정의할 수 있습니다. 이러한 재정의는 동시성에 안전한 수정 사항이며 진행 중인 업로드 또는 후속 관리자 Download 직접 호출에는 영향을 주지 않습니다. 예를 들어 특정 업로드 요청에 대한 PartSize 구성을 재정의하려면 다음을 수행합니다.

```
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
})
```

예제**버킷의 모든 객체 다운로드**

다음 예제에서는 페이지 매김을 사용하여 Amazon S3 버킷에서 객체 목록을 수집합니다. 그런 다음 각 객체를 로컬 파일로 다운로드합니다.

```
package main

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

    "github.com/aws/aws-sdk-go-v2/aws"

```

```

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

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

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

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

    paginator := s3.NewListObjectsV2Paginator(client, &s3.ListObjectsV2Input{
        Bucket: &Bucket,
        Prefix: &Prefix,
    })

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

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

```

```

        return err
    }

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

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

    return err
}

```

GetBucketRegion

[GetBucketRegion](#)은 Amazon S3 버킷의 AWS 리전 위치를 결정하는 유틸리티 함수입니다.

GetBucketRegion는 Amazon S3 클라이언트를 가져와 클라이언트의 구성된 리전과 연결된 AWS 파티션 내에서 요청된 버킷의 위치를 결정하는 데 사용됩니다.

예를 들어 버킷 *amzn-s3-demo-bucket*의 리전을 찾으려면 다음을 수행합니다.

```

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)
```

GetBucketRegion이 버킷의 위치를 확인할 수 없는 경우 함수는 예제와 같이 [BucketNotFound](#) 오류 유형을 반환합니다.

확인할 수 없는 스트리밍 입력

PutObject 및 UploadPart와 같은 API 작업의 경우 Amazon S3 클라이언트는 Body 입력 파라미터의 값이 기본적으로 [io.Seeker](#) 인터페이스를 구현할 것으로 예상합니다. 클라이언트는 io.Seeker 인터페이스를 사용하여 업로드할 값의 길이를 결정하고 [요청 서명](#)에 대한 페이로드 해시를 계산합니다. Body 입력 파라미터 값이 io.Seeker를 구현하지 않으면 애플리케이션에 오류가 발생합니다.

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

기능 옵션을 사용하여 작업 메서드의 [미들웨어](#)를 수정하여 이 동작을 변경할 수 있습니다. [WithAPIOptions](#) 헬퍼는 0개 이상의 미들웨어 변형기에 대한 기능 옵션을 반환합니다. 클라이언트가 페이로드 해시를 계산하지 않도록 하고 [서명되지 않은 페이로드](#) 요청 서명을 사용하려면 [v4.SwapComputePayloadSHA256ForUnsignedPayloadMiddleware](#)를 추가합니다.

```
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

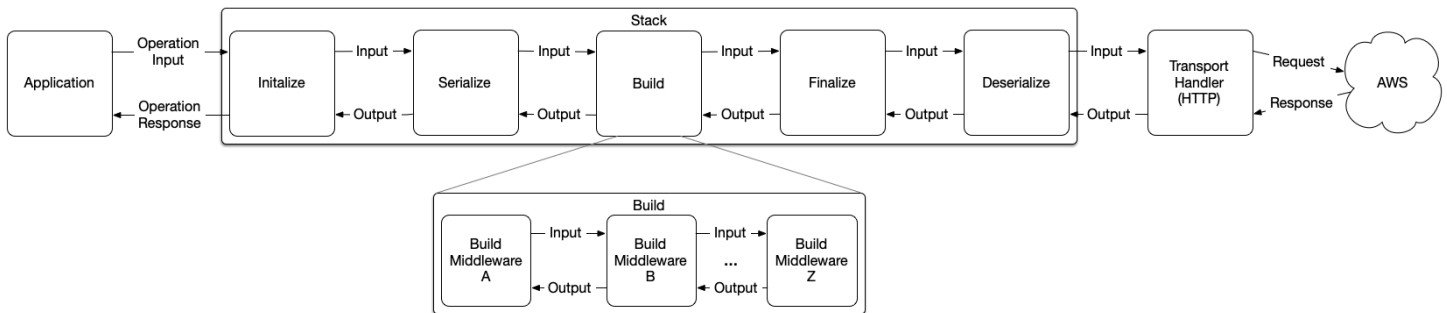
Amazon S3에서는 버킷에 업로드된 모든 객체에 콘텐츠 길이를 제공해야 합니다. Body 입력 파라미터는 io.Seeker 인터페이스를 구현하지 않으므로 클라이언트는 요청에 대한 ContentLength 파라미터를 계산할 수 없습니다. 파라미터는 애플리케이션에서 제공해야 합니다. ContentLength 파라미터가 제공되지 않으면 요청이 실패합니다. 탐색할 수 없고 알려진 길이가 없는 업로드에는 SDK의 [Amazon S3 Upload Manager](#)를 사용합니다.

미들웨어로 AWS SDK for Go v2 클라이언트 요청 사용자 지정

⚠ Warning

클라이언트 요청 파이프라인을 수정하면 형식이 잘못된 또는 유효하지 않은 요청이 발생하거나 예기치 않은 애플리케이션 오류가 발생할 수 있습니다. 이 기능은 SDK 인터페이스에서 기본적으로 제공하지 않는 고급 사용 사례를 위한 것입니다.

서비스 작업의 [스택](#)에 하나 이상의 미들웨어를 등록하여 AWS SDK for Go 클라이언트 요청을 사용자 지정할 수 있습니다. 스택은 초기화, 직렬화, 빌드, 완료 및 역직렬화와 같은 일련의 단계로 구성됩니다. 각 단계에는 해당 단계의 입력 및 출력 유형에서 작동하는 0개 이상의 미들웨어가 포함되어 있습니다. 다음 다이어그램과 표는 작업의 요청 및 응답이 스택을 통과하는 방법에 대한 개요를 제공합니다.



스택 단계	설명
초기화	입력을 준비하고 필요에 따라 기본 파라미터를 설정합니다.
직렬화	대상 전송 계층에 적합한 프로토콜 형식으로 입력을 직렬화합니다.
빌드	HTTP Content-Length와 같은 직렬화된 입력에 추가 메타데이터를 연결합니다.
마무리	재시도 및 인증을 포함한 최종 메시지 준비입니다(SigV4 서명).

스택 단계	설명
역직렬화	프로토콜 형식의 응답을 구조화된 유형 또는 오류로 역직렬화합니다.

지정된 단계 내의 각 미들웨어에는 미들웨어의 ID 메시드에 따라 결정되는 고유 식별자가 있어야 합니다. 미들웨어 식별자는 특정 미들웨어의 인스턴스 하나만 단계에 등록되도록 하고 다른 단계 미들웨어가 해당 미들웨어에 대해 삽입되도록 허용합니다.

단계의 `Insert` 또는 `Add` 메시지를 사용하여 단계 미들웨어를 연결합니다. `Add`를 사용하여 [middleware.Before](#)를 [RelativePosition](#)으로 지정하고 [middleware.After](#)를 단계 끝에 연결하여 단계 시작 부분에 미들웨어를 연결합니다. `Insert`를 사용하여 다른 단계 미들웨어를 기준으로 미들웨어를 삽입하여 미들웨어를 단계에 연결합니다.

Warning

`Add` 메시지를 사용하여 사용자 지정 단계 미들웨어를 안전하게 삽입해야 합니다. `Insert`를 사용하면 사용자 지정 미들웨어와 삽입하려는 미들웨어 간에 종속성이 생성됩니다. 애플리케이션에 변경 사항이 발생하지 않도록 스택 단계 내의 미들웨어를 불투명한 것으로 간주해야 합니다.

사용자 지정 미들웨어 작성

각 스택 단계에는 지정된 단계에 미들웨어를 연결하기 위해 충족해야 하는 인터페이스가 있습니다. 제공된 `StepMiddlewareFunc` 함수 중 하나를 사용하여 이 인터페이스를 빠르게 충족할 수 있습니다. 다음 표에는 인터페이스를 충족하는 데 사용할 수 있는 단계, 인터페이스 및 헬퍼 함수가 요약되어 있습니다.

단계	인터페이스	Helper 함수
초기화	InitializeMiddleware	InitializeMiddlewareFunc
빌드	BuildMiddleware	BuildMiddlewareFunc
직렬화	SerializeMiddleware	SerializeMiddlewareFunc

단계	인터페이스	Helper 함수
마무리	FinalizeMiddleware	FinalizeMiddlewareFunc
역직렬화	DeserializeMiddleware	DeserializeMiddlewareFunc

다음 예제에서는 제공되지 않은 경우 사용자 지정 미들웨어를 작성하여 Amazon S3 GetObject API 직접 호출의 버킷 멤버를 채우는 방법을 보여줍니다. 이 미들웨어는 다음 예제에서 참조되어 스택에 단계 미들웨어를 연결하는 방법을 보여줍니다.

```
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)
})
```

모든 클라이언트에 미들웨어 연결

[aws.Config](#) 유형의 `APIOptions` 멤버를 사용해 미들웨어를 추가하여 사용자 지정 단계 미들웨어를 모든 클라이언트에 연결할 수 있습니다. 다음 예제에서는 애플리케이션 `aws.Config` 객체를 사용하여 구성된 모든 클라이언트에 `defaultBucket` 미들웨어를 연결합니다.

```
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)
```

특정 작업에 미들웨어 연결

작업에 대한 가변 인수 목록을 사용하여 클라이언트의 `APIOptions` 멤버를 수정하는 방법으로 사용자 정의 단계 미들웨어를 특정 클라이언트 작업에 첨부할 수 있습니다. 다음 예제에서는 `defaultBucket` 미들웨어를 특정 Amazon S3 `GetObject` 작업 간접 호출에 연결합니다.

```
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)
})
```

스택으로 메타데이터 전달

특정 상황에서는 정보 또는 상태를 공유하여 둘 이상의 미들웨어가 동시에 작동해야 할 수 있습니다. [context.Context](#)를 통해 [middleware.WithStackValue](#)를 사용하여 이 메타데이터를 전달할 수 있습니다. [middleware.WithStackValue](#)는 지정된 키-값 페어를 제공된 컨텍스트에 연결하고 범위를 현재 실행 중인 스택으로 안전하게 제한합니다. 이러한 스택 범위 값은 [middleware.GetStackValue](#)를 사용하여 해당 값을 저장하는 데 사용되는 키를 제공하여 컨텍스트에서 검색할 수 있습니다. 키는 비슷해야 하며 충돌을 방지하려면 자체 유형을 컨텍스트 키로 정의해야 합니다. 다음 예제에서는 두 미들웨어가 [context.Context](#)를 사용하여 스택에 정보를 전달하는 방법을 보여줍니다.

```
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 customInititalize = 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)
}))

```

SDK에서 제공하는 메타데이터

AWS SDK for Go는 제공된 컨텍스트에서 검색할 수 있는 여러 메타데이터 값을 제공합니다. 이러한 값을 사용하여 실행 중인 서비스, 작업 또는 대상 리전에 따라 동작을 수정하는 보다 동적인 미들웨어를 활성화할 수 있습니다. 사용 가능한 몇 가지 키가 아래 표에 나와 있습니다.

키	리트리버	설명
ServiceID	GetServiceID	실행 중인 스택의 서비스 식별자를 검색합니다. 이를 서비스 클라이언트 패키지의 ServiceID 상수와 비교할 수 있습니다.
OperationName	GetOperationName	실행 중인 스택의 작업 이름을 검색합니다.
Logger	GetLogger	미들웨어에서 메시지를 로깅하는 데 사용할 수 있는 로거를 검색합니다.

스택에 메타데이터 전달

[미들웨어 메타데이터](#)를 사용하여 메타데이터 키와 값 페어를 추가하는 방법으로 스택을 통해 메타데이터를 전달할 수 있습니다. 각 미들웨어 단계는 출력 구조, 메타데이터 및 오류를 반환합니다. 사용자 지정 미들웨어는 단계에서 다음 핸들러를 호출하여 수신한 메타데이터를 반환해야 합니다. 이렇게 하면 다운스트림 미들웨어에 의해 추가된 메타데이터가 서비스 작업을 간접 호출하는 애플리케이션에 전파됩니다. 결과 메타데이터는 ResultMetadata 구조 멤버를 통해 작업의 출력 세이프를 따라 간접 호출 애플리케이션에 액세스할 수 있습니다.

다음 예제에서는 사용자 지정 미들웨어가 작업 출력의 일부로 반환되는 메타데이터를 추가하는 방법을 보여줍니다.

```
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)

```

FAQ

SDK의 HTTP 클라이언트를 구성하려면 어떻게 해야 합니까? 지침이나 모범 사례가 있습니까?

특정 워크로드에 가장 효과적인 방식으로 HTTP 워크플로를 구성하는 방법에 대한 지침을 고객에게 제공할 수 없습니다. 이에 대한 답은 다변수 방정식의 결과물이며, 입력 요소는 다음을 포함하지만 이에 국한되지 않습니다.

- 애플리케이션의 네트워크 공간(TPS, 처리량 등)
- 사용 중인 서비스
- 배포의 컴퓨팅 특성
- 배포의 지리적 특성
- 원하는 애플리케이션 동작 또는 애플리케이션 자체의 요구 사항(SLA, 타이밍 등)

작업 제한 시간을 구성하려면 어떻게 해야 합니까?

이전 질문과 마찬가지로 이는 종속적입니다. 여기서 고려해야 할 요소는 다음과 같습니다.

- HTTP 클라이언트 구성과 관련된 위의 모든 요소
- 자체 애플리케이션 타이밍 또는 SLA 제약 조건(예: 다른 소비자에게 트래픽을 제공하는 경우)

이 질문에 대한 답변은 업스트림 동작에 대한 순수한 경험적 관찰을 기반으로 해서는 안 됩니다. 예: "이 작업을 1,000번 호출했습니다. 최대 5초가 걸렸으므로 안전 계수가 2배~10초인 시간을 기준으로 제한 시간을 설정하겠습니다." 환경 조건이 변경될 수 있고, 서비스가 일시적으로 저하될 수 있으며, 이러한 유형의 가정은 경고 없이 잘못될 수 있습니다.

SDK의 요청이 시간 초과되거나 너무 오래 걸리는데, 이 문제를 해결하려면 어떻게 해야 합니까?

네트워크에 소요된 시간이 길어 작업 호출이 연장되거나 제한 시간이 초과되면 지원할 수 없습니다. SDK의 '네트워크 시간'은 다음 중 하나로 정의됩니다.

- SDK 클라이언트의 `HTTPClient.Do()` 메서드에 소요된 시간

- 호출자에게 전달된 HTTP 응답 본문의 `Read()`에서 소요된 시간(예: `GetObject`)

작업 지연 시간 또는 제한 시간으로 인해 문제가 발생하는 경우, 첫 번째 조치는 SDK 작업 수명 주기의 원격 측정을 확보하여 네트워크에 소요된 시간과 작업의 주변 오버헤드 간의 타이밍 분석을 결정하는 것입니다. 이를 달성할 수 있는 재사용 가능한 코드 조각이 포함된 [타이밍 SDK 작업](#)에 대한 가이드를 참조하세요.

read: connection reset 오류를 해결하려면 어떻게 해야 할까요?

SDK는 기본적으로 `connection reset` 패턴과 일치하는 모든 오류를 재시도합니다. 여기에는 작업의 HTTP 응답이 완전히 사용되고 모델링된 결과 유형으로 역직렬화되는 대부분의 작업에 대한 오류 처리가 포함됩니다.

그러나 이 오류는 여전히 재시도 루프 외부의 컨텍스트에서 발생할 수 있습니다. 특정 서비스 작업은 API의 HTTP 응답 본문을 직접 호출자에게 전달하여 `io.ReadCloser`(예: `GetObject`의 객체 페이로드)를 통해 네트워크에서 직접 사용할 수 있습니다. 응답 본문에서 `Read`를 수행할 때 이 오류가 발생할 수 있습니다.

이 오류는 호스트, 서비스 또는 중개자(예: NAT 게이트웨이, 프록시, 로드 밸런서)가 응답을 읽으려고 시도하는 동안 연결을 종료했음을 나타냅니다.

이는 다음과 같은 여러 가지 이유로 발생할 수 있습니다.

- 응답 자체가 수신된 후(서비스 작업이 호출된 후) 일정 시간 동안 응답 본문을 사용하지 않았습니다. 이러한 유형의 작업에 대해 가능한 한 빨리 HTTP 응답 본문을 사용하는 것이 좋습니다.
- 이전에 수신한 응답 본문을 닫지 않았습니다. 이로 인해 특정 플랫폼에서 연결이 재설정될 수 있습니다. 콘텐츠를 사용하는지 여부에 관계없이 작업의 응답에 제공된 **`io.ReadCloser`** 인스턴스를 모두 닫아야 합니다.

그 외에도 네트워크 엣지에서 영향을 받는 연결에 대해 `tcpdump`를 실행해 봅니다(예: 제어하는 프록시 이후). AWS 엔드포인트가 TCP RST를 전송하는 것으로 보이는 경우 AWS 지원 콘솔을 사용하여 문제가 되는 서비스에 대한 사례를 열어야 합니다. 문제가 발생한 시점의 요청 ID와 특정 타임스탬프를 제공할 준비를 합니다.

SDK에서 HTTP 프록시를 사용할 때 "잘못된 서명" 오류가 발생하는 이유는 무엇입니까?

AWS 서비스에 대한 서명 알고리즘(일반적으로 sigv4)은 직렬화된 요청의 헤더, 특히 접두사가 인 대부분의 헤더에 연결됩니다X-. 프록시는 SDK가 계산한 서명을 효과적으로 해제하는 추가 전달 정보(종종 X-Forwarded-For 헤더를 통해)를 추가하여 발신 요청을 수정하는 경향이 있습니다.

HTTP 프록시를 사용하고 있고 서명 오류가 발생하는 경우 프록시에서 나가는 것처럼 보이는 요청을 캡처하고 차이가 있는지 확인해야 합니다.

SDK 작업 타이밍

SDK에서 제한 시간/지연 시간 문제를 디버깅할 때는 예상보다 실행 시간이 더 오래 걸리는 작업 수명 주기의 구성 요소를 식별하는 것이 중요합니다. 시작점으로, 일반적으로 전체 작업 직접 호출과 HTTP 직접 호출 자체 간의 타이밍 분석을 검사해야 합니다.

다음 샘플 프로그램은 SQS 클라이언트용 smithy-go 미들웨어 측면에서 기본 계측 프로브를 구현하고 사용 방법을 보여줍니다. 프로브는 각 작업 직접 호출에 대해 다음 정보를 내보냅니다.

- AWS 요청 ID
- 서비스 ID
- 작업 이름
- 작업 간접 호출 시간
- http 직접 호출 시간

내보낸 각 메시지에는 핸들러 스택의 시작 부분에 설정된 고유한(단일 작업으로) '간접 호출 ID' 접두사가 붙습니다.

계측의 진입점은 WithOperationTiming으로 노출되며, 계측 '이벤트'를 형식이 지정된 문자열로 수신하는 메시지 처리 함수를 수락하도록 파라미터화됩니다. PrintfMSGHandler는 편의상 stdout에 메시지를 덤프합니다.

여기에 사용되는 서비스는 서로 바꿔 사용할 수 있습니다. 모든 서비스 클라이언트 옵션은 APIOptions 및 HTTPClient를 구성으로 수락합니다. 예를 들어 WithOperationTiming은 대신 다음과 같이 선언될 수 있습니다.

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

```
// etc.
```

이를 변경하는 경우 반환하는 함수의 서명도 변경해야 합니다.

```
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()
}

```

```

    }
    }()
}
wg.Wait()
}

```

이 프로그램의 샘플 출력:

```

[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
```

AWS SDK for Go v2를 사용한 유닛 테스트

애플리케이션에서 SDK를 사용하는 경우 애플리케이션의 유닛 테스트를 위해 SDK의 모의 테스트를 진행해야 할 수 있습니다. SDK를 모의 테스트하면 SDK 내부가 아닌 테스트 대상에 중점을 두고 테스트를 진행할 수 있습니다.

모의 테스트를 지원하려면 `s3.Client`와 같은 구체적인 서비스 클라이언트, 페이지네이터 및 웨이터 유형 대신 Go 인터페이스를 사용합니다. 이를 통해 애플리케이션은 종속성 주입과 같은 패턴을 사용하여 애플리케이션 로직을 테스트할 수 있습니다.

클라이언트 작업 모의 테스트

이 예제에서 `S3GetObjectAPI`는 `GetObjectFromS3` 함수에 필요한 Amazon S3 API 작업 세트를 정의하는 인터페이스입니다. `S3GetObjectAPI`는 Amazon S3 클라이언트의 [GetObject](#) 메서드에 의해 충족됩니다.

```
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)
}
```

GetObjectFromS3 함수를 테스트하려면 mockGetObjectAPI를 사용하여 S3GetObjectAPI 인터페이스 정의를 충족합니다. 그런 다음 mockGetObjectAPI 유형을 사용하여 서비스 클라이언트에서 반환된 출력 및 오류 응답을 모의 테스트합니다.

```
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)
        }
    })
}
}

```

모의 테스트 페이지네이터

서비스 클라이언트와 마찬가지로 페이지네이터에 대한 Go 인터페이스를 정의하여 페이지네이터를 모의 테스트할 수 있습니다. 해당 인터페이스는 애플리케이션의 코드에서 사용됩니다. 이를 통해 애플리케이션을 실행할 때 SDK의 구현을 사용하고 테스트에 구현을 모의 테스트할 수 있습니다.

다음 예제에서 `ListObjectsV2Pager`는 `CountObjects` 함수에 필요한 Amazon S3 [ListObjectsV2Paginator](#)의 동작을 정의하는 인터페이스입니다.

```

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
}

```

CountObjects를 테스트하려면 ListObjectsV2Pager 인터페이스 정의를 충족하는 mockListObjectsV2Pager 유형을 생성합니다. 그런 다음 mockListObjectsV2Pager를 사용하여 서비스 작업 페이지네이터의 출력 및 오류 응답의 페이지징 동작을 복제합니다.

```

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)
    }
}
```

SDK for Go V2 코드 예제

이 주제의 코드 예제에서는와 함께 AWS SDK for Go V2를 사용하는 방법을 보여줍니다 AWS.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

일부 서비스에는 서비스와 관련된 라이브러리 또는 함수를 활용하는 방법을 보여주는 추가 예제 범주가 포함되어 있습니다.

서비스

- [SDK for Go V2를 사용한 API Gateway 예제](#)
- [SDK for Go V2를 사용한 Aurora 예제](#)
- [SDK for Go V2를 사용한 Amazon Bedrock 예제](#)
- [SDK for Go V2를 사용한 Amazon Bedrock Runtime 예제](#)
- [CloudFormation SDK for Go V2를 사용한 예제](#)
- [SDK for Go V2를 사용한 CloudWatch Logs 예제](#)
- [SDK for Go V2를 사용한 Amazon Cognito ID 공급자 예제](#)
- [SDK for Go V2를 사용한 Amazon DocumentDB 예제](#)
- [SDK for Go V2를 사용한 DynamoDB 예제](#)
- [SDK for Go V2를 사용한 IAM 예제](#)
- [SDK for Go V2를 사용한 Kinesis 예제](#)
- [SDK for Go V2를 사용한 Lambda 예제](#)
- [SDK for Go V2를 사용한 Amazon MSK 예제](#)
- [SDK for Go V2를 사용한 Partner Central 예제](#)
- [SDK for Go V2를 사용한 Amazon RDS 예제](#)
- [SDK for Go V2를 사용한 Amazon Redshift 예제](#)

- [SDK for Go V2를 사용한 Amazon S3 예제](#)
- [SDK for Go V2를 사용한 Amazon SNS 예제](#)
- [SDK for Go V2를 사용한 Amazon SQS 예제](#)

SDK for Go V2를 사용한 API Gateway 예제

다음 코드 예제에서는 API Gateway와 함께 AWS SDK for Go V2를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

AWS 커뮤니티 기여는 여러 팀이 생성하고 유지 관리하는 예입니다 AWS. 피드백을 제공하려면 연결된 리포지토리에 제공된 메커니즘을 사용합니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- [AWS 커뮤니티 기여](#)

AWS 커뮤니티 기여

서버리스 애플리케이션 빌드 및 테스트

다음 코드 예제에서는 Lambda 및 DynamoDB와 함께 API 게이트웨이를 사용하여 서버리스 애플리케이션을 빌드하고 테스트하는 방법을 보여줍니다.

SDK for Go V2

Go SDK를 사용하여 Lambda 및 DynamoDB가 포함된 API 게이트웨이로 구성된 서버리스 애플리케이션을 빌드하고 테스트하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda

SDK for Go V2를 사용한 Aurora 예제

다음 코드 예제에서는 Aurora와 함께 AWS SDK for Go V2를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- [시작하기](#)
- [기본 사항](#)
- [작업](#)

시작하기

Hello Aurora

다음 코드 예제에서는 Aurora를 사용하여 시작하는 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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)
        }
    }
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [DescribeDBClusters](#)를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 사용자 지정 Aurora DB 클러스터 파라미터 그룹을 만들고 파라미터 값을 설정합니다.
- 파라미터 그룹을 사용하는 DB 클러스터를 생성합니다.
- 데이터베이스가 포함된 DB 인스턴스를 생성합니다.
- DB 클러스터의 스냅샷을 만든 다음, 리소스를 정리합니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
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\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()
}
```

Aurora 작업을 관리하기 위해 시나리오가 호출하는 함수를 정의합니다.

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

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

type DbClusters struct {
    AuroraClient *rds.Client
}

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

```

    return &output.DBClusterParameterGroups[0], err
}
}

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

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

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

```

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

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

// UpdateParameters updates parameters in a named DB cluster parameter group.
func (clusters *DbClusters) UpdateParameters(ctx context.Context, parameterGroupName
string, params []types.Parameter) error {
_, err := clusters.AuroraClient.ModifyDBClusterParameterGroup(ctx,
&rds.ModifyDBClusterParameterGroupInput{
    DBClusterParameterGroupName: aws.String(parameterGroupName),
    Parameters:                  params,
})
if err != nil {
    log.Printf("Couldn't update parameters in %v: %v\n", parameterGroupName, err)
    return err
} else {
    return nil
}
```

```

    }
}

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

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

    output, err := clusters.AuroraClient.CreateDBCluster(ctx,
        &rds.CreateDBClusterInput{
            DBClusterIdentifier:      aws.String(clusterName),
            Engine:                 aws.String(dbEngine),
            DBClusterParameterGroupName: aws.String(parameterGroupName),
        })

```

```

    DatabaseName:      aws.String(dbName),
    EngineVersion:     aws.String(dbEngineVersion),
    MasterUserPassword: aws.String(adminPassword),
    MasterUsername:     aws.String(adminName),
  })
  if err != nil {
    log.Printf("Couldn't create DB cluster %v: %v\n", clusterName, err)
    return nil, err
  } else {
    return output.DBCluster, err
  }
}

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

// CreateClusterSnapshot creates a snapshot of a DB cluster.
func (clusters *DbClusters) CreateClusterSnapshot(ctx context.Context, clusterName
string, snapshotName string) (
  *types.DBClusterSnapshot, error) {
  output, err := clusters.AuroraClient.CreateDBClusterSnapshot(ctx,
&rds.CreateDBClusterSnapshotInput{
    DBClusterIdentifier:      aws.String(clusterName),
    DBClusterSnapshotIdentifier: aws.String(snapshotName),
  })
  if err != nil {
    log.Printf("Couldn't create snapshot %v: %v\n", snapshotName, err)
    return nil, err
  }
}

```

```

    } else {
        return output.DBClusterSnapshot, nil
    }
}

// GetClusterSnapshot gets a DB cluster snapshot.
func (clusters *DbClusters) GetClusterSnapshot(ctx context.Context, snapshotName
string) (*types.DBClusterSnapshot, error) {
    output, err := clusters.AuroraClient.DescribeDBClusterSnapshots(ctx,
        &rds.DescribeDBClusterSnapshotsInput{
            DBClusterSnapshotIdentifier: aws.String(snapshotName),
        })
    if err != nil {
        log.Printf("Couldn't get snapshot %v: %v\n", snapshotName, err)
        return nil, err
    } else {
        return &output.DBClusterSnapshots[0], nil
    }
}

// CreateInstanceInCluster creates a database instance in an existing DB cluster.
// The first database that is
// created defaults to a read-write DB instance.
func (clusters *DbClusters) CreateInstanceInCluster(ctx context.Context, clusterName
string, instanceName string,
dbEngine string, dbInstanceClass string) (*types.DBInstance, error) {
    output, err := clusters.AuroraClient.CreateDBInstance(ctx,
        &rds.CreateDBInstanceInput{
            DBInstanceIdentifier: aws.String(instanceName),
            DBClusterIdentifier: aws.String(clusterName),
            Engine:              aws.String(dbEngine),
            DBInstanceClass:      aws.String(dbInstanceClass),
        })
    if err != nil {
        log.Printf("Couldn't create instance %v: %v\n", instanceName, err)
        return nil, err
    } else {
        return output.DBInstance, nil
    }
}

```

```
// GetInstance gets data about a DB instance.
func (clusters *DbClusters) GetInstance(ctx context.Context, instanceName string) (
    *types.DBInstance, error) {
    output, err := clusters.AuroraClient.DescribeDBInstances(ctx,
        &rds.DescribeDBInstancesInput{
            DBInstanceIdentifier: aws.String(instanceName),
        })
    if err != nil {
        var notFoundError *types.DBInstanceNotFoundFault
        if errors.As(err, &notFoundError) {
            log.Printf("DB instance %v does not exist.\n", instanceName)
            err = nil
        } else {
            log.Printf("Couldn't get instance %v: %v\n", instanceName, err)
        }
        return nil, err
    } else {
        return &output.DBInstances[0], nil
    }
}

// DeleteInstance deletes a DB instance.
func (clusters *DbClusters) DeleteInstance(ctx context.Context, instanceName string)
error {
    _, err := clusters.AuroraClient.DeleteDBInstance(ctx, &rds.DeleteDBInstanceInput{
        DBInstanceIdentifier:    aws.String(instanceName),
        SkipFinalSnapshot:      aws.Bool(true),
        DeleteAutomatedBackups: aws.Bool(true),
    })
    if err != nil {
        log.Printf("Couldn't delete instance %v: %v\n", instanceName, err)
        return err
    } else {
        return nil
    }
}
```

```
// GetEngineVersions gets database engine versions that are available for the
// specified engine
// and parameter group family.
func (clusters *DbClusters) GetEngineVersions(ctx context.Context, engine string,
parameterGroupFamily string) (
[]types.DBEngineVersion, error) {
output, err := clusters.AuroraClient.DescribeDBEngineVersions(ctx,
&rds.DescribeDBEngineVersionsInput{
Engine:          aws.String(engine),
DBParameterGroupFamily: aws.String(parameterGroupFamily),
})
if err != nil {
log.Printf("Couldn't get engine versions for %v: %v\n", engine, err)
return nil, err
} else {
return output.DBEngineVersions, nil
}
}

// GetOrderableInstances uses a paginator to get DB instance options that can be
// used to create DB instances that are
// compatible with a set of specifications.
func (clusters *DbClusters) GetOrderableInstances(ctx context.Context, engine
string, engineVersion string) (
[]types.OrderableDBInstanceOption, error) {

var output *rds.DescribeOrderableDBInstanceOptionsOutput
var instances []types.OrderableDBInstanceOption
var err error
orderablePaginator :=
rds.NewDescribeOrderableDBInstanceOptionsPaginator(clusters.AuroraClient,
&rds.DescribeOrderableDBInstanceOptionsInput{
Engine:          aws.String(engine),
EngineVersion: aws.String(engineVersion),
})
for orderablePaginator.HasMorePages() {
output, err = orderablePaginator.NextPage(ctx)
if err != nil {
log.Printf("Couldn't get orderable DB instances: %v\n", err)
break
} else {
instances = append(instances, output.OrderableDBInstanceOptions...)
}
```



```
    }  
  }  
  return instances, err  
}
```

• API 세부 정보는 AWS SDK for Go API 참조의 다음 항목을 참조하세요.

- [CreateDBCluster](#)
- [CreateDBClusterParameterGroup](#)
- [CreateDBClusterSnapshot](#)
- [CreateDBInstance](#)
- [DeleteDBCluster](#)
- [DeleteDBClusterParameterGroup](#)
- [DeleteDBInstance](#)
- [DescribeDBClusterParameterGroups](#)
- [DescribeDBClusterParameters](#)
- [DescribeDBClusterSnapshots](#)
- [DescribeDBClusters](#)
- [DescribeDBEngineVersions](#)
- [DescribeDBInstances](#)
- [DescribeOrderableDBInstanceOptions](#)
- [ModifyDBClusterParameterGroup](#)

작업

CreateDBCluster

다음 코드 예시는 CreateDBCluster의 사용 방법을 보여줍니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
  }
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [CreateDBCluster](#)를 참조하세요.

CreateDBClusterParameterGroup

다음 코드 예시는 CreateDBClusterParameterGroup의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
    }
}

```

- API 세부 정보는 AWS SDK for Go SDK for Rust API 참조의 [CreateDBClusterParameterGroup](#)를 참조하세요.

CreateDBClusterSnapshot

다음 코드 예시는 CreateDBClusterSnapshot의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
    }
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [CreateDBClusterSnapshot](#)을 참조하세요.

CreateDBInstance

다음 코드 예시는 CreateDBInstance의 사용 방법을 보여줍니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [CreateDBInstance](#)를 참조하세요.

DeleteDBCluster

다음 코드 예시는 DeleteDBCluster의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
  }
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [DeleteDBCluster](#)를 참조하세요.

DeleteDBClusterParameterGroup

다음 코드 예시는 DeleteDBClusterParameterGroup의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
    }
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [DeleteDBClusterParameterGroup](#)을 참조하세요.

DeleteDBInstance

다음 코드 예시는 DeleteDBInstance의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
    }
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [DeleteDBInstance](#)를 참조하세요.

DescribeDBClusterParameterGroups

다음 코드 예시는 DescribeDBClusterParameterGroups의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
    }
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [DescribeDBClusterParameterGroups](#)를 참조하세요.

DescribeDBClusterParameters

다음 코드 예시는 DescribeDBClusterParameters의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

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

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

type DbClusters struct {
    AuroraClient *rds.Client
}

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

    var output *rds.DescribeDBClusterParametersOutput
    var params []types.Parameter
    var err error
    parameterPaginator :=
    rds.NewDescribeDBClusterParametersPaginator(clusters.AuroraClient,
        &rds.DescribeDBClusterParametersInput{
            DBClusterParameterGroupName: aws.String(parameterGroupName),
            Source:                       aws.String(source),
```

```

    })
    for parameterPaginator.HasMorePages() {
        output, err = parameterPaginator.NextPage(ctx)
        if err != nil {
            log.Printf("Couldn't get parameters for %v: %v\n", parameterGroupName, err)
            break
        } else {
            params = append(params, output.Parameters...)
        }
    }
    return params, err
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [DescribeDBClusterParameters](#)를 참조하세요.

DescribeDBClusterSnapshots

다음 코드 예시는 DescribeDBClusterSnapshots의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
    }
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [DescribeDBClusterSnapshots](#)를 참조하세요.

DescribeDBClusters

다음 코드 예시는 DescribeDBClusters의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
    }
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [DescribeDBClusters](#)를 참조하세요.

DescribeDBEngineVersions

다음 코드 예시는 DescribeDBEngineVersions의 사용 방법을 보여줍니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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  
    }  
}
```



```
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [DescribeDBEngineVersions](#)을 참조하세요.

DescribeDBInstances

다음 코드 예시는 DescribeDBInstances의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
    }
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [DescribeDBInstances](#)를 참조하세요.

DescribeOrderableDBInstanceOptions

다음 코드 예시는 DescribeOrderableDBInstanceOptions의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [DescribeOrderableDBInstanceOptions](#)를 참조하세요.

ModifyDBClusterParameterGroup

다음 코드 예시는 ModifyDBClusterParameterGroup의 사용 방법을 보여줍니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
    }
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [ModifyDBClusterParameterGroup](#)을 참조하세요.

SDK for Go V2를 사용한 Amazon Bedrock 예제

다음 코드 예제에서는 Amazon Bedrock과 함께 AWS SDK for Go V2를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- [시작하기](#)
- [작업](#)

시작하기

Amazon Bedrock 시작

다음 코드 예제에서는 Amazon Bedrock 사용을 시작하는 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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)
    }
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [ListFoundationModels](#)를 참조하세요.

작업

ListFoundationModels

다음 코드 예시는 ListFoundationModels의 사용 방법을 보여줍니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

사용 가능한 Bedrock 기본 모델을 나열합니다.

```
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
}
```

```
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [ListFoundationModels](#)를 참조하세요.

SDK for Go V2를 사용한 Amazon Bedrock Runtime 예시

다음 코드 예제에서는 Amazon Bedrock 런타임과 함께 AWS SDK for Go V2를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- [시작하기](#)
- [시나리오](#)
- [Amazon Titan Image Generator](#)
- [Anthropic Claude](#)

시작하기

Amazon Bedrock 시작

다음 코드 예제에서는 Amazon Bedrock 사용을 시작하는 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.


```
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)
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [InvokeModel](#)을 참조하세요.

시나리오

Amazon Bedrock에서 여러 파운데이션 모델 간접 호출

다음 코드 예제에서는 Amazon Bedrock에서 프롬프트를 준비하고 다양한 대규모 언어 모델(LLM)에 전송하는 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon Bedrock에서 여러 파운데이션 모델을 간접 호출합니다.

```

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)
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 다음 주제를 참조하세요.
 - [InvokeModel](#)
 - [InvokeModelWithResponseStream](#)

Amazon Titan Image Generator

InvokeModel

다음 코드 예제에서는 이미지 생성을 위해 Amazon Bedrock에서 Amazon Titan Image를 간접 호출하는 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon Titan Image Generator를 사용하여 이미지를 생성합니다.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [InvokeModel](#)을 참조하세요.

Anthropic Claude

Converse

다음 코드 예제에서는 Bedrock의 Converse API를 사용하여 Anthropic Claude에 텍스트 메시지를 보내는 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Bedrock의 Converse API를 사용하여 Anthropic Claude에 텍스트 메시지를 보냅니다.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [Converse](#)를 참조하세요.

InvokeModel

다음 코드 예제에서는 Invoke Model API를 사용하여 Anthropic Claude에 텍스트 메시지를 보내는 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Anthropic Claude 2 파운데이션 모델을 간접 호출하여 텍스트를 생성합니다.

```
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
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [InvokeModel](#)을 참조하세요.

InvokeModelWithResponseStream

다음 코드 예제에서는 Invoke Model API를 사용하여 Anthropic Claude 모델에 텍스트 메시지를 보내고 응답 스트림을 프린트하는 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Invoke Model API를 사용하여 텍스트 메시지를 보내고 응답 스트림을 실시간으로 처리합니다.

```
import (
    "bytes"
    "context"
    "encoding/json"
    "fmt"
    "log"
    "strings"

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

// InvokeModelWithResponseStreamWrapper encapsulates Amazon Bedrock actions used in
// the examples.
// It contains a Bedrock Runtime client that is used to invoke foundation models.
type InvokeModelWithResponseStreamWrapper struct {
    BedrockRuntimeClient *bedrockruntime.Client
}
```

```
// Each model provider defines their own individual request and response formats.
// For the format, ranges, and default values for the different models, refer to:
// https://docs.aws.amazon.com/bedrock/latest/userguide/model-parameters.html

type Request struct {
    Prompt      string `json:"prompt"`
    MaxTokensToSample int    `json:"max_tokens_to_sample"`
    Temperature float64 `json:"temperature,omitempty"`
}

type Response struct {
    Completion string `json:"completion"`
}

// Invokes Anthropic Claude on Amazon Bedrock to run an inference and asynchronously
// process the response stream.

func (wrapper InvokeModelWithResponseStreamWrapper)
    InvokeModelWithResponseStream(ctx context.Context, prompt string) (string, error) {

    modelId := "anthropic.claude-v2"

    // Anthropic Claude requires you to enclose the prompt as follows:
    prefix := "Human: "
    postfix := "\n\nAssistant:"
    prompt = prefix + prompt + postfix

    request := ClaudeRequest{
        Prompt:      prompt,
        MaxTokensToSample: 200,
        Temperature: 0.5,
        StopSequences: []string{"\n\nHuman:"},
    }

    body, err := json.Marshal(request)
    if err != nil {
        log.Panicln("Couldn't marshal the request: ", err)
    }

    output, err := wrapper.BedrockRuntimeClient.InvokeModelWithResponseStream(ctx,
        &bedrockruntime.InvokeModelWithResponseStreamInput{
            Body:      body,
            ModelId:   aws.String(modelId),
        })
}
```

```

    ContentType: aws.String("application/json"),
  })

  if err != nil {
    errMsg := err.Error()
    if strings.Contains(errMsg, "no such host") {
      log.Printf("The Bedrock service is not available in the selected region. Please
double-check the service availability for your region at https://aws.amazon.com/
about-aws/global-infrastructure/regional-product-services/.\\n")
    } else if strings.Contains(errMsg, "Could not resolve the foundation model") {
      log.Printf("Could not resolve the foundation model from model identifier: \"%v
\\n. Please verify that the requested model exists and is accessible within the
specified region.\\n", modelId)
    } else {
      log.Printf("Couldn't invoke Anthropic Claude. Here's why: %v\\n", err)
    }
  }

  resp, err := processStreamingOutput(ctx, output, func(ctx context.Context, part
[]byte) error {
    fmt.Print(string(part))
    return nil
  })

  if err != nil {
    log.Fatal("streaming output processing error: ", err)
  }

  return resp.Completion, nil
}

type StreamingOutputHandler func(ctx context.Context, part []byte) error

func processStreamingOutput(ctx context.Context, output
*bedrockruntime.InvokeModelWithResponseStreamOutput, handler
StreamingOutputHandler) (Response, error) {

  var combinedResult string
  resp := Response{}

  for event := range output.GetStream().Events() {
    switch v := event.(type) {
      case *types.ResponseStreamMemberChunk:

```

```

//fmt.Println("payload", string(v.Value.Bytes))

var resp Response
err := json.NewDecoder(bytes.NewReader(v.Value.Bytes)).Decode(&resp)
if err != nil {
    return resp, err
}

err = handler(ctx, []byte(resp.Completion))
if err != nil {
    return resp, err
}

combinedResult += resp.Completion

case *types.UnknownUnionMember:
    fmt.Println("unknown tag:", v.Tag)

default:
    fmt.Println("union is nil or unknown type")
}
}

resp.Completion = combinedResult

return resp, nil
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [InvokeModelWithResponseStream](#)을 참조하세요.

CloudFormation SDK for Go V2를 사용한 예제

다음 코드 예제에서는와 함께 AWS SDK for Go V2를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다 CloudFormation.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- [작업](#)

작업

DescribeStacks

다음 코드 예시는 DescribeStacks의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [DescribeStacks](#)를 참조하세요.

SDK for Go V2를 사용한 CloudWatch Logs 예시

다음 코드 예제에서는 CloudWatch Logs와 함께 AWS SDK for Go V2를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- [작업](#)

작업

StartLiveTail

다음 코드 예시는 StartLiveTail의 사용 방법을 보여줍니다.

SDK for Go V2

필수 파일을 포함합니다.

```
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"  
)
```

Live Tail 세션의 이벤트를 처리합니다.

```
func handleEventStreamAsync(stream *cloudwatchlogs.StartLiveTailEventStream) {  
    eventsChan := stream.Events()  
    for {  
        event := <-eventsChan  
        switch e := event.(type) {  
        case *types.StartLiveTailResponseStreamMemberSessionStart:  
            log.Println("Received SessionStart event")  
        case *types.StartLiveTailResponseStreamMemberSessionUpdate:  
            for _, logEvent := range e.Value.SessionResults {  
                log.Println(*logEvent.Message)  
            }  
        default:  
            // Handle on-stream exceptions  
            if err := stream.Err(); err != nil {  
                log.Fatalf("Error occurred during streaming: %v", err)  
            } else if event == nil {  
                log.Println("Stream is Closed")  
                return  
            } else {  
                log.Fatalf("Unknown event type: %T", e)  
            }  
        }  
    }  
}
```

Live Tail 세션을 시작합니다.

```
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)
```

일정 시간이 경과하면 Live Tail 세션을 중단합니다.

```
// Close the stream (which ends the session) after a timeout
time.Sleep(10 * time.Second)
stream.Close()
log.Println("Event stream closed")
```

- API 세부 정보는 AWS SDK for Go API 참조의 [StartLiveTail](#)을 참조하세요.

SDK for Go V2를 사용한 Amazon Cognito ID 공급자 예제

다음 코드 예제에서는 Amazon Cognito 자격 증명 공급자와 함께 AWS SDK for Go V2를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- [시작하기](#)
- [작업](#)
- [시나리오](#)

시작하기

Hello Amazon Cognito

다음 코드 예시에서는 Amazon Cognito 사용을 시작하는 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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"
```


작업

AdminCreateUser

다음 코드 예시는 AdminCreateUser의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [AdminCreateUser](#)를 참조하세요.

AdminSetUserPassword

다음 코드 예시는 AdminSetUserPassword의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [AdminSetUserPassword](#)를 참조하세요.

ConfirmForgotPassword

다음 코드 예시는 ConfirmForgotPassword의 사용 방법을 보여줍니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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  
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [ConfirmForgotPassword](#)를 참조하세요.

DeleteUser

다음 코드 예시는 DeleteUser의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [DeleteUser](#)를 참조하세요.

ForgotPassword

다음 코드 예시는 ForgotPassword의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [ForgotPassword](#)를 참조하세요.

InitiateAuth

다음 코드 예시는 InitiateAuth의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [InitiateAuth](#)를 참조하세요.

ListUserPools

다음 코드 예시는 ListUserPools의 사용 방법을 보여줍니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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)
    }
}
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [ListFunctions](#)를 참조하세요.

SignUp

다음 코드 예시는 SignUp의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [SignUp](#)를 참조하세요.

UpdateUserPool

다음 코드 예시는 UpdateUserPool의 사용 방법을 보여줍니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [UpdateUserPool](#)을 참조하세요.

시나리오

Lambda 함수를 사용하여 알려진 사용자를 자동으로 확인

다음 코드 예제는 Lambda 함수를 사용하여 알려진 Amazon Cognito 사용자를 자동으로 확인하는 방법을 보여줍니다.

- PreSignUp 트리거에 대해 Lambda 함수를 호출하도록 사용자 풀을 구성합니다.
- Amazon Cognito를 사용하여 사용자 가입시키기
- Lambda 함수는 DynamoDB 테이블을 스캔하고 알려진 사용자를 자동으로 확인합니다.
- 새 사용자로 로그인한 다음 리소스를 정리합니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
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))
}
```

Lambda 함수를 사용하여 PreSignUp 트리거를 처리합니다.

```
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"`
    Email 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.Email)
    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)
}

```

일반적인 작업을 수행하는 구조체를 생성합니다.

```

import (
    "context"
    "log"
    "strings"
    "time"
    "user_pools_and_lambda_triggers/actions"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cloudformation"
    "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this example.

```

```

type IScenarioHelper interface {
    Pause(secs int)
    GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
        error)
    PopulateUserTable(ctx context.Context, tableName string)
    GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
    AddKnownUser(ctx context.Context, tableName string, user actions.User)
    ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
// example.
type ScenarioHelper struct {
    questioner demotools.IQuestioner
    dynamoActor *actions.DynamoActions
    cfnActor     *actions.CloudFormationActions
    cwlActor     *actions.CloudWatchLogsActions
    isTestRun    bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
    ScenarioHelper {
    scenario := ScenarioHelper{
        questioner: questioner,
        dynamoActor: &actions.DynamoActions{DynamoClient:
            dynamodb.NewFromConfig(sdkConfig)},
        cfnActor:     &actions.CloudFormationActions{CfnClient:
            cloudformation.NewFromConfig(sdkConfig)},
        cwlActor:     &actions.CloudWatchLogsActions{CwlClient:
            cloudwatchlogs.NewFromConfig(sdkConfig)},
    }
    return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
    if !helper.isTestRun {
        time.Sleep(time.Duration(secs) * time.Second)
    }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
// structured format.

```

```

func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName string)
(actions.StackOutputs, error) {
    return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
string) {
    log.Printf("First, let's add some users to the DynamoDB %v table we'll use for this
example.\n", tableName)
    err := helper.dynamoActor.PopulateTable(ctx, tableName)
    if err != nil {
        panic(err)
    }
}

// GetKnownUsers gets the users from the known users table in a structured format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
(actions.UserList, error) {
    knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
    if err != nil {
        log.Printf("Couldn't get known users from table %v. Here's why: %v\n", tableName,
err)
    }
    return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
user actions.User) {
    log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users table...
\n",
        user.UserName, user.UserEmail)
    err := helper.dynamoActor.AddUser(ctx, tableName, user)
    if err != nil {
        panic(err)
    }
}

// ListRecentLogEvents gets the most recent log stream and events for the specified
Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context, functionName
string) {
    log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")

```

```

helper.Pause(10)
log.Println("Okay, let's check the logs to find what's happened recently with your
Lambda function.")
logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
if err != nil {
    panic(err)
}
log.Printf("Getting some recent events from log stream %v\n",
*logStream.LogStreamName)
events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
*logStream.LogStreamName, 10)
if err != nil {
    panic(err)
}
for _, event := range events {
    log.Printf("\t\t%v", *event.Message)
}
log.Println(strings.Repeat("-", 88))
}

```

Amazon Cognito 작업을 래핑하는 구조체를 생성합니다.

```

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),
        })
    return err
}

```



```

    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
}

```

DynamoDB 작업을 래핑하는 구조체를 생성합니다.

```

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
    Email 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
}
```

CloudWatch Logs 작업을 래핑하는 구조체를 생성합니다.

```
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
}

```

CloudFormation 작업을 래핑하는 구조체를 생성합니다.

```

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
}

```

리소스를 정리합니다.

```

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!")
}
}
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 다음 주제를 참조하세요.
 - [DeleteUser](#)
 - [InitiateAuth](#)
 - [SignUp](#)
 - [UpdateUserPool](#)

Lambda 함수를 사용하여 알려진 사용자를 자동으로 마이그레이션

다음 코드 예제는 Lambda 함수를 사용하여 알려진 Amazon Cognito 사용자를 자동으로 마이그레이션하는 방법을 보여줍니다.

- MigrateUser 트리거에 대해 Lambda 함수를 호출하도록 사용자 풀을 구성합니다.
- 사용자 풀에 없는 사용자 이름과 이메일을 사용하여 Amazon Cognito에 로그인합니다.
- Lambda 함수는 DynamoDB 테이블을 스캔하고 알려진 사용자를 사용자 풀로 자동으로 마이그레이션합니다.
- 암호 찾기 흐름을 수행하여 마이그레이션된 사용자의 암호를 재설정합니다.
- 새 사용자로 로그인한 다음 리소스를 정리합니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
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))
}

```

Lambda 함수를 사용하여 MigrateUser 트리거를 처리합니다.

```
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)
}
```

일반적인 작업을 수행하는 구조체를 생성합니다.

```
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))
}

```

Amazon Cognito 작업을 래핑하는 구조체를 생성합니다.

```

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
}

```

DynamoDB 작업을 래핑하는 구조체를 생성합니다.

```

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
}

```

CloudWatch Logs 작업을 래핑하는 구조체를 생성합니다.

```

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
}

```

CloudFormation 작업을 래핑하는 구조체를 생성합니다.

```

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
}

```

리소스를 정리합니다.

```

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!")
}
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 다음 주제를 참조하세요.
 - [ConfirmForgotPassword](#)
 - [DeleteUser](#)
 - [ForgotPassword](#)
 - [InitiateAuth](#)
 - [SignUp](#)
 - [UpdateUserPool](#)

Amazon Cognito 사용자 인증 후 Lambda 함수를 사용하여 사용자 지정 활동 데이터 작성

다음 코드 예제는 Amazon Cognito 사용자 인증 후 Lambda 함수를 사용하여 사용자 지정 작업 데이터를 쓰는 방법을 보여줍니다.

- 관리자 함수를 사용하여 사용자 풀에 사용자를 추가합니다.
- PostAuthentication 트리거에 대해 Lambda 함수를 호출하도록 사용자 풀을 구성합니다.
- 새로운 사용자를 Amazon Cognito에 로그인시킵니다.
- Lambda 함수는 CloudWatch Logs와 DynamoDB 테이블에 사용자 지정 정보를 작성합니다.
- DynamoDB 테이블에서 사용자 지정 데이터를 가져오기 및 표시한 다음 리소스를 정리합니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
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))
}
```

Lambda 함수를 사용하여 PostAuthentication 트리거를 처리합니다.

```
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)
}

```

일반적인 작업을 수행하는 구조체를 생성합니다.

```

import (
  "context"
  "log"
  "strings"
  "time"
  "user_pools_and_lambda_triggers/actions"

  "github.com/aws/aws-sdk-go-v2/aws"
  "github.com/aws/aws-sdk-go-v2/service/cloudformation"
  "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
  "github.com/aws/aws-sdk-go-v2/service/dynamodb"
  "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

```

```
// IScenarioHelper defines common functions used by the workflows in this example.
type IScenarioHelper interface {
    Pause(secs int)
    GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
        error)
    PopulateUserTable(ctx context.Context, tableName string)
    GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
    AddKnownUser(ctx context.Context, tableName string, user actions.User)
    ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
// example.
type ScenarioHelper struct {
    questioner demotools.IQuestioner
    dynamoActor *actions.DynamoActions
    cfnActor     *actions.CloudFormationActions
    cwlActor     *actions.CloudWatchLogsActions
    isTestRun    bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
    ScenarioHelper {
    scenario := ScenarioHelper{
        questioner: questioner,
        dynamoActor: &actions.DynamoActions{DynamoClient:
            dynamodb.NewFromConfig(sdkConfig)},
        cfnActor:     &actions.CloudFormationActions{CfnClient:
            cloudformation.NewFromConfig(sdkConfig)},
        cwlActor:     &actions.CloudWatchLogsActions{CwlClient:
            cloudwatchlogs.NewFromConfig(sdkConfig)},
    }
    return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
    if !helper.isTestRun {
        time.Sleep(time.Duration(secs) * time.Second)
    }
}
```



```
// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
// structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName string)
(actions.StackOutputs, error) {
    return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
string) {
    log.Printf("First, let's add some users to the DynamoDB %v table we'll use for this
example.\n", tableName)
    err := helper.dynamoActor.PopulateTable(ctx, tableName)
    if err != nil {
        panic(err)
    }
}

// GetKnownUsers gets the users from the known users table in a structured format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
(actions.UserList, error) {
    knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
    if err != nil {
        log.Printf("Couldn't get known users from table %v. Here's why: %v\n", tableName,
err)
    }
    return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
user actions.User) {
    log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users table...
\n",
        user.UserName, user.UserEmail)
    err := helper.dynamoActor.AddUser(ctx, tableName, user)
    if err != nil {
        panic(err)
    }
}

// ListRecentLogEvents gets the most recent log stream and events for the specified
// Lambda function and displays them.
```

```

func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context, functionName
string) {
    log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
    helper.Pause(10)
    log.Println("Okay, let's check the logs to find what's happened recently with your
Lambda function.")
    logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
    if err != nil {
        panic(err)
    }
    log.Printf("Getting some recent events from log stream %v\n",
*logStream.LogStreamName)
    events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
*logStream.LogStreamName, 10)
    if err != nil {
        panic(err)
    }
    for _, event := range events {
        log.Printf("\t\t%v", *event.Message)
    }
    log.Println(strings.Repeat("-", 88))
}

```

Amazon Cognito 작업을 래핑하는 구조체를 생성합니다.

```

import (
    "context"
    "errors"
    "log"

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

type CognitoActions struct {
    CognitoClient *cognitoidentityprovider.Client
}

```

```
// Trigger and TriggerInfo define typed data for updating an Amazon Cognito trigger.
type Trigger int

const (
    PreSignUp Trigger = iota
    UserMigration
    PostAuthentication
)

type TriggerInfo struct {
    Trigger      Trigger
    HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger is
// specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId string,
    triggers ...TriggerInfo) error {
    output, err := actor.CognitoClient.DescribeUserPool(ctx,
        &cognitoidentityprovider.DescribeUserPoolInput{
            UserPoolId: aws.String(userPoolId),
        })
    if err != nil {
        log.Printf("Couldn't get info about user pool %v. Here's why: %v\n", userPoolId,
            err)
        return err
    }
    lambdaConfig := output.UserPool.LambdaConfig
    for _, trigger := range triggers {
        switch trigger.Trigger {
        case PreSignUp:
            lambdaConfig.PreSignUp = trigger.HandlerArn
        case UserMigration:
            lambdaConfig.UserMigration = trigger.HandlerArn
        case PostAuthentication:
            lambdaConfig.PostAuthentication = trigger.HandlerArn
        }
    }
    _, err = actor.CognitoClient.UpdateUserPool(ctx,
        &cognitoidentityprovider.UpdateUserPoolInput{
            UserPoolId:    aws.String(userPoolId),
            LambdaConfig: lambdaConfig,
        })
}
```

```

    })
    if err != nil {
        log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
    }
    return err
}

// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
string, password string, userEmail string) (bool, error) {
    confirmed := false
    output, err := actor.CognitoClient.SignUp(ctx,
    &cognitoidentityprovider.SignUpInput{
        ClientId: aws.String(clientId),
        Password: aws.String(password),
        Username: aws.String(userName),
        UserAttributes: []types.AttributeType{
            {Name: aws.String("email"), Value: aws.String(userEmail)},
        },
    })
    if err != nil {
        var invalidPassword *types.InvalidPasswordException
        if errors.As(err, &invalidPassword) {
            log.Println(*invalidPassword.Message)
        } else {
            log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
        }
    } else {
        confirmed = output.UserConfirmed
    }
    return confirmed, err
}

// SignIn signs in a user to Amazon Cognito using a username and password
authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
string, password string) (*types.AuthenticationResultType, error) {
    var authResult *types.AuthenticationResultType
    output, err := actor.CognitoClient.InitiateAuth(ctx,
    &cognitoidentityprovider.InitiateAuthInput{

```

```

    AuthFlow:      "USER_PASSWORD_AUTH",
    ClientId:      aws.String(clientId),
    AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
  })
  if err != nil {
    var resetRequired *types.PasswordResetRequiredException
    if errors.As(err, &resetRequired) {
      log.Println(*resetRequired.Message)
    } else {
      log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
    }
  } else {
    authResult = output.AuthenticationResult
  }
  return authResult, err
}

```

// ForgotPassword starts a password recovery flow for a user. This flow typically sends a confirmation code

// to the user's configured notification destination, such as email.

```

func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
  userName string) (*types.CodeDeliveryDetailsType, error) {
  output, err := actor.CognitoClient.ForgotPassword(ctx,
    &cognitoidentityprovider.ForgotPasswordInput{
      ClientId: aws.String(clientId),
      Username: aws.String(userName),
    })
  if err != nil {
    log.Printf("Couldn't start password reset for user '%v'. Here's why: %v\n",
      userName, err)
  }
  return output.CodeDeliveryDetails, err
}

```

// ConfirmForgotPassword confirms a user with a confirmation code and a new password.

```

func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
  string, code string, userName string, password string) error {
  _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
    &cognitoidentityprovider.ConfirmForgotPasswordInput{

```

```

    ClientId:      aws.String(clientId),
    ConfirmationCode: aws.String(code),
    Password:      aws.String(password),
    Username:      aws.String(userName),
  })
  if err != nil {
    var invalidPassword *types.InvalidPasswordException
    if errors.As(err, &invalidPassword) {
      log.Println(*invalidPassword.Message)
    } else {
      log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
    }
  }
  return err
}

// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken string)
  error {
  _, err := actor.CognitoClient.DeleteUser(ctx,
    &cognitoidentityprovider.DeleteUserInput{
      AccessToken: aws.String(userAccessToken),
    })
  if err != nil {
    log.Printf("Couldn't delete user. Here's why: %v\n", err)
  }
  return err
}

// AdminCreateUser uses administrator credentials to add a user to a user pool. This
  method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId string,
  userName string, userEmail string) error {
  _, err := actor.CognitoClient.AdminCreateUser(ctx,
    &cognitoidentityprovider.AdminCreateUserInput{
      UserPoolId:      aws.String(userPoolId),
      Username:        aws.String(userName),
      MessageAction:   types.MessageActionTypeSuppress,
    })
  if err != nil {
    log.Printf("Couldn't create user. Here's why: %v\n", err)
  }
  return err
}

```

```

    UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
aws.String(userEmail)}}},
})
if err != nil {
    var userExists *types.UsernameExistsException
    if errors.As(err, &userExists) {
        log.Printf("User %v already exists in the user pool.", userName)
        err = nil
    } else {
        log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
    }
}
return err
}

// AdminSetUserPassword uses administrator credentials to set a password for a user
// without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
string, userName string, password string) error {
    _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
    &cognitoidentityprovider.AdminSetUserPasswordInput{
        Password:    aws.String(password),
        UserPoolId:  aws.String(userPoolId),
        Username:    aws.String(userName),
        Permanent:   true,
    })
    if err != nil {
        var invalidPassword *types.InvalidPasswordException
        if errors.As(err, &invalidPassword) {
            log.Println(*invalidPassword.Message)
        } else {
            log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName, err)
        }
    }
    return err
}

```

DynamoDB 작업을 래핑하는 구조체를 생성합니다.

```
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
}

```

CloudWatch Logs 작업을 래핑하는 구조체를 생성합니다.

```

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
}

```

CloudFormation 작업을 래핑하는 구조체를 생성합니다.

```

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
}

```

리소스를 정리합니다.

```

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!")
}
}
}

```

• API 세부 정보는 AWS SDK for Go API 참조의 다음 주제를 참조하세요.

- [AdminCreateUser](#)
- [AdminSetUserPassword](#)
- [DeleteUser](#)
- [InitiateAuth](#)
- [UpdateUserPool](#)

SDK for Go V2를 사용한 Amazon DocumentDB 예제

다음 코드 예제에서는 Amazon DocumentDB와 함께 AWS SDK for Go V2를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- [서버리스 예제](#)

서버리스 예제

Amazon DocumentDB 트리거에서 간접적으로 Lambda 함수 호출

다음 코드 예제에서는 DocumentDB 변경 스트림에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 DocumentDB 페이로드를 검색하고 레코드 콘텐츠를 로깅합니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [서버리스 예제](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Go를 사용하여 Lambda로 Amazon DocumentDB 이벤트 소비

```
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))
}

```

SDK for Go V2를 사용한 DynamoDB 예제

다음 코드 예제에서는 DynamoDB와 함께 AWS SDK for Go V2를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

AWS 커뮤니티 기여는 여러 팀이 생성하고 유지 관리하는 예입니다 AWS. 피드백을 제공하려면 연결된 리포지토리에 제공된 메커니즘을 사용합니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- [기본 사항](#)
- [작업](#)
- [시나리오](#)
- [서버리스 예제](#)
- [AWS 커뮤니티 기여](#)

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 영화 데이터를 저장할 수 있는 테이블을 생성합니다.
- 테이블에 하나의 영화를 추가하고 가져오고 업데이트합니다.
- 샘플 JSON 파일에서 테이블에 영화 데이터를 씁니다.
- 특정 연도에 개봉된 영화를 쿼리합니다.
- 특정 연도 범위 동안 개봉된 영화를 스캔합니다.
- 테이블에서 영화를 삭제한 다음, 테이블을 삭제합니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

대화식 시나리오를 실행하여 테이블을 생성하고 테이블에 대한 작업을 수행합니다.

```
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\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\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))
}

```

이 예시에서 사용되는 Movie 구조체를 정의합니다.

```

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"])
}
```

DynamoDB 작업을 호출하는 구문과 메서드를 생성합니다.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 다음 주제를 참조하세요.
 - [BatchWriteItem](#)
 - [CreateTable](#)
 - [DeleteItem](#)
 - [DeleteTable](#)

- [DescribeTable](#)
- [GetItem](#)
- [PutItem](#)
- [Query](#)
- [Scan](#)
- [UpdateItem](#)

작업

BatchExecuteStatement

다음 코드 예시는 BatchExecuteStatement의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

예제를 위한 함수 수신기 구문을 정의합니다.

```
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
}
```

INSERT 문 배치를 사용하여 항목을 추가합니다.

```
// 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
}
```

SELECT 문 배치를 사용하여 항목을 가져옵니다.

```
// 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
}
```

UPDATE 문 배치를 사용하여 항목을 업데이트합니다.

```
// 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
}
```

DELETE 문 배치를 사용하여 항목을 삭제합니다.

```
// 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
}

```

이 예시에서 사용되는 Movie 구조체를 정의합니다.

```

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"])
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [BatchExecuteStatement](#)를 참조하세요.

BatchWriteItem

다음 코드 예시는 BatchWriteItem의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
}

```

이 예시에서 사용되는 Movie 구조체를 정의합니다.

```

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"])
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [BatchWriteItem](#)을 참조하세요.

CreateTable

다음 코드 예시는 CreateTable의 사용 방법을 보여줍니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [CreateTable](#)을 참조하세요.

DeleteItem

다음 코드 예시는 DeleteItem의 사용 방법을 보여줍니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}
```

```
}

```

이 예시에서 사용되는 Movie 구조체를 정의합니다.

```
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"])
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [DeleteItem](#)을 참조하세요.

DeleteTable

다음 코드 예시는 DeleteTable의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [DeleteTable](#)을 참조하세요.

DescribeTable

다음 코드 예시는 DescribeTable의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [DescribeTable](#)을 참조하세요.

ExecuteStatement

다음 코드 예시는 ExecuteStatement의 사용 방법을 보여줍니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

예제를 위한 함수 수신기 구문을 정의합니다.

```
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
}
```

INSERT 문을 사용하여 항목을 추가합니다.

```
// 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
}

```

SELECT 문을 사용하여 항목을 가져옵니다.

```

// 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
}

```

SELECT 문을 사용하여 항목 목록을 가져오고 결과를 투영합니다.

```

// 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
}

```

UPDATE 문을 사용하여 항목을 업데이트합니다.

```

// 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
}

```

DELETE 문을 사용하여 항목을 삭제합니다.

```

// 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
}

```

이 예시에서 사용되는 Movie 구조체를 정의합니다.

```

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"])
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [ExecuteStatement](#)를 참조하세요.

GetItem

다음 코드 예시는 GetItem의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

이 예시에서 사용되는 Movie 구조체를 정의합니다.

```

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"])
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [GetItem](#)을 참조하세요.

ListTables

다음 코드 예시는 ListTables의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [ListTables](#)를 참조하세요.

PutItem

다음 코드 예시는 PutItem의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

이 예시에서 사용되는 Movie 구조체를 정의합니다.

```

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"])
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [PutItem](#)을 참조하세요.

Query

다음 코드 예시는 Query의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

이 예시에서 사용되는 Movie 구조체를 정의합니다.

```

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"])
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [Query](#)를 참조하세요.

Scan

다음 코드 예시는 Scan의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```


이 예시에서 사용되는 Movie 구조체를 정의합니다.

```
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"])
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [Scan](#)을 참조하세요.

UpdateItem

다음 코드 예시는 UpdateItem의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

이 예시에서 사용되는 Movie 구조체를 정의합니다.

```

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"])
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [UpdateItem](#)을 참조하세요.

시나리오

PartiQL 문 배치를 사용하여 테이블 쿼리

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 여러 SELECT 문을 실행하여 항목 배치를 가져옵니다.
- 여러 INSERT 문을 실행하여 항목 배치를 추가합니다.
- 여러 UPDATE 문을 실행하여 항목 배치를 업데이트합니다.
- 여러 DELETE 문을 실행하여 항목 배치를 삭제합니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

테이블을 생성하고 PartiQL 쿼리 배치를 실행하는 시나리오를 실행합니다.

```
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))
}

```

이 예시에서 사용되는 Movie 구조체를 정의합니다.

```

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"])
}

```

PartiQL 문을 실행하는 구조체와 메서드를 생성합니다.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [BatchExecuteStatement](#)를 참조하세요.

PartiQL을 사용하여 테이블 쿼리

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- SELECT 문을 실행하여 항목을 가져옵니다.
- INSERT 문을 실행하여 항목을 추가합니다.
- UPDATE 문을 실행하여 항목을 업데이트합니다.
- DELETE 문을 실행하여 항목을 삭제합니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

테이블을 생성하고 PartiQL 쿼리를 실행하는 시나리오를 실행합니다.

```
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))
}
```

이 예시에서 사용되는 Movie 구조체를 정의합니다.

```
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"])
}
```

PartiQL 문을 실행하는 구조체와 메서드를 생성합니다.

```
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
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [ExecuteStatement](#)를 참조하세요.

서버리스 예제

DynamoDB 트리거에서 간접적으로 Lambda 함수 간접 호출

다음 코드 예제에서는 DynamoDB 스트림에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 DynamoDB 페이로드를 검색하고 레코드 콘텐츠를 로깅합니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [서버리스 예제](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Go를 사용하여 Lambda로 DynamoDB 이벤트 사용.

```
// 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)
}

```

DynamoDB 트리거로 Lambda 함수에 대한 배치 항목 실패 보고

다음 코드 예제에서는 DynamoDB 스트림에서 이벤트를 수신하는 Lambda 함수에 대한 부분 배치 응답을 구현하는 방법을 보여줍니다. 이 함수는 응답으로 배치 항목 실패를 보고하고 나중에 해당 메시지를 다시 시도하도록 Lambda에 신호를 보냅니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [서버리스 예제](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Go를 사용하여 Lambda로 DynamoDB 배치 항목 실패 보고.

```

// 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 커뮤니티 기여

서버리스 애플리케이션 빌드 및 테스트

다음 코드 예제에서는 Lambda 및 DynamoDB와 함께 API 게이트웨이를 사용하여 서버리스 애플리케이션을 빌드하고 테스트하는 방법을 보여줍니다.

SDK for Go V2

Go SDK를 사용하여 Lambda 및 DynamoDB가 포함된 API 게이트웨이로 구성된 서버리스 애플리케이션을 빌드하고 테스트하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda

SDK for Go V2를 사용한 IAM 예제

다음 코드 예제에서는 IAM과 함께 AWS SDK for Go V2를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- [시작하기](#)
- [기본 사항](#)
- [작업](#)

시작하기

Hello IAM

다음 코드 예제에서는 IAM 사용을 시작하는 방법을 보여줍니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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)
    }
  }
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [ListPolicies](#)를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 사용자를 생성하고 역할을 수입하는 방법을 보여줍니다.

Warning

보안 위험을 방지하려면 목적별 소프트웨어를 개발하거나 실제 데이터로 작업할 때 IAM 사용자를 인증에 사용하지 마세요. 대신 [AWS IAM Identity Center](#)과 같은 보안 인증 공급자를 통한 페더레이션을 사용하십시오.

- 권한이 없는 사용자를 생성합니다.
- 계정에 대한 Amazon S3 버킷을 나열할 수 있는 권한을 부여하는 역할을 생성합니다.
- 사용자가 역할을 수입할 수 있도록 정책을 추가합니다.
- 역할을 수입하고 임시 자격 증명 정보를 사용하여 S3 버킷을 나열한 후 리소스를 정리합니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
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)
}

```

계정 작업을 래핑하는 구조체를 정의합니다.

```

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
}
```

정책 작업을 래핑하는 구조체를 정의합니다.

```
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
}

```

역할 작업을 래핑하는 구조체를 정의합니다.

```

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
}

```

사용자 작업을 래핑하는 구조체를 정의합니다.

```

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 for user %v. Here's why: %v\n",
            keyId, userName, err)
    }
    return err
}
```

```

}))
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 다음 주제를 참조하십시오.
 - [AttachRolePolicy](#)
 - [CreateAccessKey](#)
 - [CreatePolicy](#)
 - [CreateRole](#)
 - [CreateUser](#)
 - [DeleteAccessKey](#)
 - [DeletePolicy](#)
 - [DeleteRole](#)
 - [DeleteUser](#)
 - [DeleteUserPolicy](#)
 - [DetachRolePolicy](#)

- [PutUserPolicy](#)

작업

AttachRolePolicy

다음 코드 예시는 AttachRolePolicy의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [AttachRolePolicy](#)를 참조하세요.

CreateAccessKey

다음 코드 예시는 CreateAccessKey의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [CreateAccessKey](#)를 참조하세요.

CreatePolicy

다음 코드 예시는 CreatePolicy의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [CreatePolicy](#)를 참조하세요.

CreateRole

다음 코드 예시는 CreateRole의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
}

```

- API 세부 정보는 [AWS SDK for Go API 참조](#)의 CreateRole을 참조하세요.

CreateServiceLinkedRole

다음 코드 예시는 CreateServiceLinkedRole의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [CreateServiceLinkedRole](#)을 참조하세요.

CreateUser

다음 코드 예시는 CreateUser의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

- API 세부 정보는 [AWS SDK for Go API 참조](#)의 CreateUser를 참조하세요.

DeleteAccessKey

다음 코드 예시는 DeleteAccessKey의 사용 방법을 보여줍니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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  
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [DeleteAccessKey](#)를 참조하세요.

DeletePolicy

다음 코드 예시는 DeletePolicy의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [DeletePolicy](#)를 참조하세요.

DeleteRole

다음 코드 예시는 DeleteRole의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [DeleteRole](#)을 참조하세요.

DeleteServiceLinkedRole

다음 코드 예시는 DeleteServiceLinkedRole의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [DeleteServiceLinkedRole](#)을 참조하세요.

DeleteUser

다음 코드 예시는 DeleteUser의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [DeleteUser](#)를 참조하세요.

DeleteUserPolicy

다음 코드 예시는 DeleteUserPolicy의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [DeleteUserPolicy](#)를 참조하세요.

DetachRolePolicy

다음 코드 예시는 DetachRolePolicy의 사용 방법을 보여줍니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [DetachRolePolicy](#)를 참조하세요.

GetAccountPasswordPolicy

다음 코드 예시는 GetAccountPasswordPolicy의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [GetAccountPasswordPolicy](#)를 참조하세요.

GetPolicy

다음 코드 예시는 GetPolicy의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [GetPolicy](#)를 참조하세요.

GetRole

다음 코드 예시는 GetRole의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [GetRole](#)을 참조하세요.

GetUser

다음 코드 예시는 GetUser의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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  
}
```


- API 세부 정보는 AWS SDK for Go API 참조의 [GetUser](#)을 참조하세요.

ListAccessKeys

다음 코드 예시는 ListAccessKeys의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [ListAccessKeys](#)를 참조하세요.

ListAttachedRolePolicies

다음 코드 예시는 ListAttachedRolePolicies의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [ListAttachedRolePolicies](#)를 참조하세요.

ListGroups

다음 코드 예시는 ListGroups의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [ListGroups](#)를 참조하세요.

ListPolicies

다음 코드 예시는 ListPolicies의 사용 방법을 보여줍니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}
```

```
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [ListPolicies](#)를 참조하세요.

ListRolePolicies

다음 코드 예시는 ListRolePolicies의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [ListRolePolicies](#)를 참조하세요.

ListRoles

다음 코드 예시는 ListRoles의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [ListRoles](#)를 참조하세요.

ListSAMLProviders

다음 코드 예시는 ListSAMLProviders의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [ListSAMLProviders](#)를 참조하세요.

ListUserPolicies

다음 코드 예시는 ListUserPolicies의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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  
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [ListUserPolicies](#)을 참조하세요.

ListUsers

다음 코드 예시는 ListUsers의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [ListUsers](#)를 참조하세요.

PutUserPolicy

다음 코드 예시는 PutUserPolicy의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [PutUserPolicy](#)를 참조하세요.

SDK for Go V2를 사용한 Kinesis 예시

다음 코드 예제에서는 Kinesis와 함께 AWS SDK for Go V2를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- [서버리스 예제](#)

서버리스 예제

Kinesis 트리거에서 간접적으로 Lambda 함수 간접 호출

다음 코드 예제에서는 Kinesis 스트림에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 Kinesis 페이로드를 검색하고, Base64에서 디코딩하고, 레코드 콘텐츠를 로깅합니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [서버리스 예제](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Go를 사용하여 Lambda로 Kinesis 이벤트를 사용합니다.

```
// 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)
}

```

Kinesis 트리거로 Lambda 함수에 대한 배치 항목 실패 보고

다음 코드 예제는 Kinesis 스트림에서 이벤트를 수신하는 Lambda 함수에 대한 부분 배치 응답을 구현하는 방법을 보여줍니다. 이 함수는 응답으로 배치 항목 실패를 보고하고 나중에 해당 메시지를 다시 시도하도록 Lambda에 신호를 보냅니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [서버리스 예제](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Go를 사용하여 Lambda로 Kinesis 배치 항목 실패를 보고합니다.

```

// 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)
}

```

SDK for Go V2를 사용한 Lambda 예제

다음 코드 예제에서는 Lambda와 함께 AWS SDK for Go V2를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

AWS 커뮤니티 기여는 여러 팀이 생성하고 유지 관리하는 예입니다 AWS. 피드백을 제공하려면 연결된 리포지토리에 제공된 메커니즘을 사용합니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- [시작하기](#)
- [기본 사항](#)
- [작업](#)
- [시나리오](#)
- [서버리스 예제](#)
- [AWS 커뮤니티 기여](#)

시작하기

Hello Lambda

다음 코드 예제에서는 Lambda 사용을 시작하는 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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)
        }
    }
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [ListFunctions](#)를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- IAM 역할과 Lambda 함수를 생성하고 핸들러 코드를 업로드합니다.
- 단일 파라미터로 함수를 간접적으로 간접 호출하고 결과를 가져옵니다.
- 함수 코드를 업데이트하고 환경 변수로 구성합니다.
- 새 파라미터로 함수를 간접적으로 간접 호출하고 결과를 가져옵니다. 반환된 실행 로그를 표시합니다.
- 계정의 함수를 나열합니다.

자세한 내용은 [콘솔로 Lambda 함수 생성](#)을 참조하세요.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Lambda 함수를 시작하는 방법을 보여주는 대화형 시나리오를 생성합니다.

```
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
}

```

개별 Lambda 작업을 래핑하는 구조체를 생성합니다.

```
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"`
}

```

숫자를 증가시키는 Lambda 핸들러를 정의합니다.

```

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

```

산술 연산을 수행하는 두 번째 Lambda 핸들러를 정의합니다.

```
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
```

• API 세부 정보는 AWS SDK for Go API 참조의 다음 항목을 참조하세요.

- [CreateFunction](#)
- [DeleteFunction](#)
- [GetFunction](#)
- [간접 호출](#)
- [ListFunctions](#)
- [UpdateFunctionCode](#)
- [UpdateFunctionConfiguration](#)

작업

CreateFunction

다음 코드 예시는 CreateFunction의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [CreateFunction](#)을 참조하세요.

DeleteFunction

다음 코드 예시는 DeleteFunction의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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)
    }
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [DeleteFunction](#)을 참조하세요.

GetFunction

다음 코드 예시는 GetFunction의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [GetFunction](#)을 참조하세요.

Invoke

다음 코드 예시는 Invoke의 사용 방법을 보여줍니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [간접 호출](#)을 참조하세요.

ListFunctions

다음 코드 예시는 ListFunctions의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [ListFunctions](#)를 참조하세요.

UpdateFunctionCode

다음 코드 예시는 UpdateFunctionCode의 사용 방법을 보여줍니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [UpdateFunctionCode](#)를 참조하세요.

UpdateFunctionConfiguration

다음 코드 예시는 UpdateFunctionConfiguration의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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)
    }
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [UpdateFunctionConfiguration](#)을 참조하세요.

시나리오

Lambda 함수를 사용하여 알려진 사용자를 자동으로 확인

다음 코드 예제는 Lambda 함수를 사용하여 알려진 Amazon Cognito 사용자를 자동으로 확인하는 방법을 보여줍니다.

- PreSignUp 트리거에 대해 Lambda 함수를 호출하도록 사용자 풀을 구성합니다.
- Amazon Cognito를 사용하여 사용자 가입시키기

- Lambda 함수는 DynamoDB 테이블을 스캔하고 알려진 사용자를 자동으로 확인합니다.
- 새 사용자로 로그인한 다음 리소스를 정리합니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
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))
}

```

Lambda 함수를 사용하여 PreSignUp 트리거를 처리합니다.

```

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)
}

```

일반적인 작업을 수행하는 구조체를 생성합니다.

```

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))
}

```

Amazon Cognito 작업을 래핑하는 구조체를 생성합니다.

```

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
}

```

DynamoDB 작업을 래핑하는 구조체를 생성합니다.

```

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
    Email 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
}

```

CloudWatch Logs 작업을 래핑하는 구조체를 생성합니다.

```

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
}

```

CloudFormation 작업을 래핑하는 구조체를 생성합니다.

```

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
}

```

리소스를 정리합니다.

```

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!")
}
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 다음 주제를 참조하세요.
 - [DeleteUser](#)
 - [InitiateAuth](#)
 - [SignUp](#)
 - [UpdateUserPool](#)

Lambda 함수를 사용하여 알려진 사용자를 자동으로 마이그레이션

다음 코드 예제는 Lambda 함수를 사용하여 알려진 Amazon Cognito 사용자를 자동으로 마이그레이션하는 방법을 보여줍니다.

- MigrateUser 트리거에 대해 Lambda 함수를 호출하도록 사용자 풀을 구성합니다.
- 사용자 풀에 없는 사용자 이름과 이메일을 사용하여 Amazon Cognito에 로그인합니다.
- Lambda 함수는 DynamoDB 테이블을 스캔하고 알려진 사용자를 사용자 풀로 자동으로 마이그레이션합니다.
- 암호 찾기 흐름을 수행하여 마이그레이션된 사용자의 암호를 재설정합니다.
- 새 사용자로 로그인한 다음 리소스를 정리합니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
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))
}

```

Lambda 함수를 사용하여 MigrateUser 트리거를 처리합니다.

```

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)
}

```

일반적인 작업을 수행하는 구조체를 생성합니다.

```

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))
}

```

Amazon Cognito 작업을 래핑하는 구조체를 생성합니다.

```
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
}

```

DynamoDB 작업을 래핑하는 구조체를 생성합니다.

```

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
}

```

CloudWatch Logs 작업을 래핑하는 구조체를 생성합니다.

```
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
}

```

CloudFormation 작업을 래핑하는 구조체를 생성합니다.

```

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
}

```

리소스를 정리합니다.

```

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!")
    }
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 다음 주제를 참조하세요.
 - [ConfirmForgotPassword](#)
 - [DeleteUser](#)
 - [ForgotPassword](#)
 - [InitiateAuth](#)
 - [SignUp](#)
 - [UpdateUserPool](#)

Amazon Cognito 사용자 인증 후 Lambda 함수를 사용하여 사용자 지정 활동 데이터 작성

다음 코드 예제는 Amazon Cognito 사용자 인증 후 Lambda 함수를 사용하여 사용자 지정 작업 데이터를 쓰는 방법을 보여줍니다.

- 관리자 함수를 사용하여 사용자 풀에 사용자를 추가합니다.
- PostAuthentication 트리거에 대해 Lambda 함수를 호출하도록 사용자 풀을 구성합니다.
- 새로운 사용자를 Amazon Cognito에 로그인시킵니다.
- Lambda 함수는 CloudWatch Logs와 DynamoDB 테이블에 사용자 지정 정보를 작성합니다.
- DynamoDB 테이블에서 사용자 지정 데이터를 가져오기 및 표시한 다음 리소스를 정리합니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
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.Email)
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))
}

```

Lambda 함수를 사용하여 PostAuthentication 트리거를 처리합니다.

```
import (
    "context"
    "fmt"
    "log"
    "os"
    "time"

    "github.com/aws/aws-lambda-go/events"
    "github.com/aws/aws-lambda-go/lambda"
    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/config"
    "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
    dynamodbtypes "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

const TABLE_NAME = "TABLE_NAME"

// LoginInfo defines structured login data that can be marshalled to a DynamoDB
// format.
type LoginInfo struct {
    UserPoolId string `dynamodbav:"UserPoolId"`
    ClientId   string `dynamodbav:"ClientId"`
    Time      string `dynamodbav:"Time"`
}

// UserInfo defines structured user data that can be marshalled to a DynamoDB
// format.
type UserInfo struct {
    Username   string    `dynamodbav:"Username"`
    UserEmail  string    `dynamodbav:"UserEmail"`
    LastLogin LoginInfo `dynamodbav:"LastLogin"`
}

// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
    userEmail, err := attributevalue.Marshal(user.UserEmail)
    if err != nil {
        panic(err)
    }
    return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
```

```

}

type handler struct {
    dynamoClient *dynamodb.Client
}

// HandleRequest handles the PostAuthentication event by writing custom data to the
// logs and
// to an Amazon DynamoDB table.
func (h *handler) HandleRequest(ctx context.Context,
    event events.CognitoEventUserPoolsPostAuthentication)
(event.CognitoEventUserPoolsPostAuthentication, error) {
    log.Printf("Received post authentication trigger from %v for user '%v'",
        event.TriggerSource, event.UserName)
    tableName := os.Getenv(TABLE_NAME)
    user := UserInfo{
        UserName: event.UserName,
        UserEmail: event.Request.UserAttributes["email"],
        LastLogin: LoginInfo{
            UserPoolId: event.UserPoolID,
            ClientId: event.CallerContext.ClientID,
            Time: time.Now().Format(time.UnixDate),
        },
    }
    // Write to CloudWatch Logs.
    fmt.Printf("#v", user)

    // Also write to an external system. This examples uses DynamoDB to demonstrate.
    userMap, err := attributevalue.MarshalMap(user)
    if err != nil {
        log.Printf("Couldn't marshal to DynamoDB map. Here's why: %v\n", err)
    } else if len(userMap) == 0 {
        log.Printf("User info marshaled to an empty map.")
    } else {
        _, err := h.dynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
            Item: userMap,
            TableName: aws.String(tableName),
        })
        if err != nil {
            log.Printf("Couldn't write to DynamoDB. Here's why: %v\n", err)
        } else {
            log.Printf("Wrote user info to DynamoDB table %v.\n", tableName)
        }
    }
}

```



```

    return event, nil
}

func main() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        log.Panicln(err)
    }
    h := handler{
        dynamoClient: dynamodb.NewFromConfig(sdkConfig),
    }
    lambda.Start(h.HandleRequest)
}

```

일반적인 작업을 수행하는 구조체를 생성합니다.

```

import (
    "context"
    "log"
    "strings"
    "time"
    "user_pools_and_lambda_triggers/actions"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/cloudformation"
    "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
    "github.com/aws/aws-sdk-go-v2/service/dynamodb"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this example.
type IScenarioHelper interface {
    Pause(secs int)
    GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs, error)
    PopulateUserTable(ctx context.Context, tableName string)
    GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
    AddKnownUser(ctx context.Context, tableName string, user actions.User)
}

```

```

    ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
// example.
type ScenarioHelper struct {
    questioner demotools.IQuestioner
    dynamoActor *actions.DynamoActions
    cfnActor     *actions.CloudFormationActions
    cwlActor     *actions.CloudWatchLogsActions
    isTestRun    bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
    ScenarioHelper {
    scenario := ScenarioHelper{
        questioner: questioner,
        dynamoActor: &actions.DynamoActions{DynamoClient:
            dynamodb.NewFromConfig(sdkConfig)},
        cfnActor:     &actions.CloudFormationActions{CfnClient:
            cloudformation.NewFromConfig(sdkConfig)},
        cwlActor:     &actions.CloudWatchLogsActions{CwlClient:
            cloudwatchlogs.NewFromConfig(sdkConfig)},
    }
    return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
    if !helper.isTestRun {
        time.Sleep(time.Duration(secs) * time.Second)
    }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
// structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName string)
    (actions.StackOutputs, error) {
    return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.

```

```

func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
    string) {
    log.Printf("First, let's add some users to the DynamoDB %v table we'll use for this
    example.\n", tableName)
    err := helper.dynamoActor.PopulateTable(ctx, tableName)
    if err != nil {
        panic(err)
    }
}

// GetKnownUsers gets the users from the known users table in a structured format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
    (actions.UserList, error) {
    knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
    if err != nil {
        log.Printf("Couldn't get known users from table %v. Here's why: %v\n", tableName,
        err)
    }
    return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
    user actions.User) {
    log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users table...
    \n",
        user.UserName, user.UserEmail)
    err := helper.dynamoActor.AddUser(ctx, tableName, user)
    if err != nil {
        panic(err)
    }
}

// ListRecentLogEvents gets the most recent log stream and events for the specified
    Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context, functionName
    string) {
    log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
    helper.Pause(10)
    log.Println("Okay, let's check the logs to find what's happened recently with your
    Lambda function.")
    logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
    if err != nil {
        panic(err)
    }
}

```

```

}
log.Printf("Getting some recent events from log stream %v\n",
*logStream.LogStreamName)
events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
*logStream.LogStreamName, 10)
if err != nil {
    panic(err)
}
for _, event := range events {
    log.Printf("\t\t%v", *event.Message)
}
log.Println(strings.Repeat("-", 88))
}

```

Amazon Cognito 작업을 래핑하는 구조체를 생성합니다.

```

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
}

```

DynamoDB 작업을 래핑하는 구조체를 생성합니다.

```

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
    Email 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
}

```

CloudWatch Logs 작업을 래핑하는 구조체를 생성합니다.

```

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
}

```

CloudFormation 작업을 래핑하는 구조체를 생성합니다.

```

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
}

```

리소스를 정리합니다.

```

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!")
}
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 다음 주제를 참조하세요.
 - [AdminCreateUser](#)
 - [AdminSetUserPassword](#)
 - [DeleteUser](#)
 - [InitiateAuth](#)
 - [UpdateUserPool](#)

서버리스 예제

Lambda 함수를 사용하여 Amazon RDS 데이터베이스에 연결

다음 코드 예제는 RDS 데이터베이스에 연결하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 간단한 데이터베이스 요청을 하고 결과를 반환합니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [서버리스 예제](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Go를 사용하여 Lambda 함수에서 Amazon RDS 데이터베이스에 연결

```

/*
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)
}

```

Kinesis 트리거에서 간접적으로 Lambda 함수 간접 호출

다음 코드 예제에서는 Kinesis 스트림에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 Kinesis 페이로드를 검색하고, Base64에서 디코딩하고, 레코드 콘텐츠를 로깅합니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [서버리스 예제](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Go를 사용하여 Lambda로 Kinesis 이벤트를 사용합니다.

```
// 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)
}
```

DynamoDB 트리거에서 간접적으로 Lambda 함수 간접 호출

다음 코드 예제에서는 DynamoDB 스트림에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 DynamoDB 페이로드를 검색하고 레코드 콘텐츠를 로깅합니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [서버리스 예제](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Go를 사용하여 Lambda로 DynamoDB 이벤트 사용.

```
// 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)
}

```

Amazon DocumentDB 트리거에서 간접적으로 Lambda 함수 호출

다음 코드 예제에서는 DocumentDB 변경 스트림에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 DocumentDB 페이로드를 검색하고 레코드 콘텐츠를 로깅합니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [서버리스 예제](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Go를 사용하여 Lambda로 Amazon DocumentDB 이벤트 소비

```

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))
}

```

Amazon MSK 트리거를 사용하여 Lambda 함수 간접 호출

다음 코드 예제에서는 Amazon MSK 클러스터에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 MSK 페이로드를 검색하고 레코드 콘텐츠를 로깅합니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [서버리스 예제](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Go를 사용하여 Lambda로 Amazon MSK 이벤트 사용

```
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)
}
```

Amazon S3 트리거를 사용하여 Lambda 함수 간접 호출

다음 코드 예제는 S3 버킷에 객체를 업로드하여 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 해당 함수는 이벤트 파라미터에서 S3 버킷 이름과 객체 키를 검색하고 Amazon S3 API를 호출하여 객체의 콘텐츠 유형을 검색하고 로깅합니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [서버리스 예제](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Go를 사용하여 Lambda로 S3 이벤트를 사용합니다.

```
// 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)
}

```

Amazon SNS 트리거를 사용하여 Lambda 함수 간접 호출

다음 코드 예제에서는 SNS 주제의 메시지를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 함수는 이벤트 파라미터에서 메시지를 검색하고 각 메시지의 내용을 로깅합니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [서버리스 예제](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Go를 사용하여 Lambda로 SNS 이벤트를 사용합니다.

```

// 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)
}

```

Amazon SQS 트리거에서 간접적으로 Lambda 함수 간접 호출

다음 코드 예제는 SQS 대기열에서 메시지를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 함수는 이벤트 파라미터에서 메시지를 검색하고 각 메시지의 내용을 로깅합니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [서버리스 예제](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Go를 사용하여 Lambda로 SQS 이벤트를 사용합니다.

```
// 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)
}
```

Kinesis 트리거로 Lambda 함수에 대한 배치 항목 실패 보고

다음 코드 예제는 Kinesis 스트림에서 이벤트를 수신하는 Lambda 함수에 대한 부분 배치 응답을 구현하는 방법을 보여줍니다. 이 함수는 응답으로 배치 항목 실패를 보고하고 나중에 해당 메시지를 다시 시도하도록 Lambda에 신호를 보냅니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [서버리스 예제](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Go를 사용하여 Lambda로 Kinesis 배치 항목 실패를 보고합니다.

```
// 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)
}

```

DynamoDB 트리거로 Lambda 함수에 대한 배치 항목 실패 보고

다음 코드 예제에서는 DynamoDB 스트림에서 이벤트를 수신하는 Lambda 함수에 대한 부분 배치 응답을 구현하는 방법을 보여줍니다. 이 함수는 응답으로 배치 항목 실패를 보고하고 나중에 해당 메시지를 다시 시도하도록 Lambda에 신호를 보냅니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [서버리스 예제](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Go를 사용하여 Lambda로 DynamoDB 배치 항목 실패 보고.

```

// 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)
}

```

Amazon SQS 트리거로 Lambda 함수에 대한 배치 항목 실패 보고

다음 코드 예제는 SQS 대기열에서 이벤트를 수신하는 Lambda 함수에 대한 부분 배치 응답을 구현하는 방법을 보여줍니다. 이 함수는 응답으로 배치 항목 실패를 보고하고 나중에 해당 메시지를 다시 시도하도록 Lambda에 신호를 보냅니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [서버리스 예제](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Go를 사용하여 Lambda로 SQS 배치 항목 실패 보고

```
// 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 커뮤니티 기여

서버리스 애플리케이션 빌드 및 테스트

다음 코드 예제에서는 Lambda 및 DynamoDB와 함께 API 게이트웨이를 사용하여 서버리스 애플리케이션을 빌드하고 테스트하는 방법을 보여줍니다.

SDK for Go V2

Go SDK를 사용하여 Lambda 및 DynamoDB가 포함된 API 게이트웨이로 구성된 서버리스 애플리케이션을 빌드하고 테스트하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 [GitHub](#)에서 전체 예제를 참조하세요.

이 예제에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda

SDK for Go V2를 사용한 Amazon MSK 예제

다음 코드 예제에서는 Amazon MSK와 함께 AWS SDK for Go V2를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- [서버리스 예제](#)

서버리스 예제

Amazon MSK 트리거를 사용하여 Lambda 함수 간접 호출

다음 코드 예제에서는 Amazon MSK 클러스터에서 레코드를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 MSK 페이로드를 검색하고 레코드 콘텐츠를 로깅합니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [서버리스 예제](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Go를 사용하여 Lambda로 Amazon MSK 이벤트 사용

```
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)
}
```

SDK for Go V2를 사용한 Partner Central 예제

다음 코드 예제에서는 Partner Central과 함께 AWS SDK for Go V2를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- [작업](#)

작업

GetOpportunity

다음 코드 예시는 GetOpportunity의 사용 방법을 보여줍니다.

SDK for Go V2

기회를 가져옵니다.

```
package main

import (
    "context"
    "encoding/json"
    "fmt"
    "log"

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

func main() {
```

```

config, err := config.LoadDefaultConfig(context.TODO())

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

config.Region = "us-east-1"

client := partnercentralselling.NewFromConfig(config)

output, err := client.GetOpportunity(context.TODO(),
&partnercentralselling.GetOpportunityInput{
    Identifier: aws.String("01111111"),
    Catalog:    aws.String("AWS"),
})

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

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

fmt.Println(string(jsonOutput))
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [GetOpportunity](#)를 참조하세요.

ListOpportunities

다음 코드 예시는 ListOpportunities의 사용 방법을 보여줍니다.

SDK for Go V2

기회를 나열합니다.

```

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))
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [ListOpportunities](#)를 참조하세요.

SDK for Go V2를 사용한 Amazon RDS 예제

다음 코드 예제에서는 Amazon RDS에서 AWS SDK for Go V2를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- [시작하기](#)
- [기본 사항](#)
- [작업](#)
- [서버리스 예제](#)

시작하기

Hello Amazon RDS

다음 코드 예제에서는 Amazon RDS 사용을 시작하는 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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)
        }
    }
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [DescribeDBInstances](#)를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 사용자 지정 DB 파라미터 그룹을 생성하고 파라미터 값을 설정합니다.
- 파라미터 그룹을 사용하도록 구성된 DB 인스턴스를 생성합니다. DB 인스턴스에는 데이터베이스도 포함되어 있습니다.
- 인스턴스의 스냅샷을 만듭니다.
- 인스턴스 및 파라미터 그룹을 삭제합니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배우보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
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()
}

```

Amazon RDS 작업을 관리하기 위해 시나리오에서 호출하는 함수를 정의합니다.

```

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.HasNextPage() {
    page := parameterPaginator.NextPage(ctx)
    output = page.Output()
    params = append(params, output.Parameters...)
  }
  return params, err
}

```

```

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 DB instance %v: %v\n", instanceName, err)
        return nil, err
    }
    return &output.DBInstance, nil
}

```

```

    })
    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
}
```

• API 세부 정보는 AWS SDK for Go API 참조의 다음 주제를 참조하세요.

- [CreateDBInstance](#)
- [CreateDBParameterGroup](#)
- [CreateDBSnapshot](#)
- [DeleteDBInstance](#)
- [DeleteDBParameterGroup](#)
- [DescribeDBEngineVersions](#)
- [DescribeDBInstances](#)
- [DescribeDBParameterGroups](#)
- [DescribeDBParameters](#)
- [DescribeDBSnapshots](#)
- [DescribeOrderableDBInstanceOptions](#)
- [ModifyDBParameterGroup](#)

작업

CreateDBInstance

다음 코드 예시는 CreateDBInstance의 사용 방법을 보여줍니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
  }
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [CreateDBInstance](#)를 참조하세요.

CreateDBParameterGroup

다음 코드 예시는 CreateDBParameterGroup의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
    }
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [CreateDBParameterGroup](#)을 참조하세요.

CreateDBSnapshot

다음 코드 예시는 CreateDBSnapshot의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
    }
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [CreateDBSnapshot](#)을 참조하세요.

DeleteDBInstance

다음 코드 예시는 DeleteDBInstance의 사용 방법을 보여줍니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
    }
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [DeleteDBInstance](#)를 참조하세요.

DeleteDBParameterGroup

다음 코드 예시는 DeleteDBParameterGroup의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [DeleteDBParameterGroup](#)을 참조하세요.

DescribeDBEngineVersions

다음 코드 예시는 DescribeDBEngineVersions의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [DescribeDBEngineVersions](#)을 참조하세요.

DescribeDBInstances

다음 코드 예시는 DescribeDBInstances의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
    }
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [DescribeDBInstances](#)를 참조하세요.

DescribeDBParameterGroups

다음 코드 예시는 DescribeDBParameterGroups의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
    }
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [DescribeDBParameterGroups](#)를 참조하세요.

DescribeDBParameters

다음 코드 예시는 DescribeDBParameters의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [DescribeDBParameters](#) 참조하세요.

DescribeDBSnapshots

다음 코드 예시는 DescribeDBSnapshots의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
    }
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [DescribeDBSnapshots](#)를 참조하세요.

DescribeOrderableDBInstanceOptions

다음 코드 예시는 DescribeOrderableDBInstanceOptions의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [DescribeOrderableDBInstanceOptions](#)를 참조하세요.

ModifyDBParameterGroup

다음 코드 예시는 ModifyDBParameterGroup의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
    }
}
```

```
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [ModifyDBParameterGroup](#)을 참조하세요.

서버리스 예제

Lambda 함수를 사용하여 Amazon RDS 데이터베이스에 연결

다음 코드 예제는 RDS 데이터베이스에 연결하는 Lambda 함수를 구현하는 방법을 보여줍니다. 이 함수는 간단한 데이터베이스 요청을 하고 결과를 반환합니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [서버리스 예제](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Go를 사용하여 Lambda 함수에서 Amazon RDS 데이터베이스에 연결

```
/*
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)
}

```

SDK for Go V2를 사용한 Amazon Redshift 예제

다음 코드 예제에서는 Amazon Redshift에서 AWS SDK for Go V2를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접적으로 호출하는 방법을 보여주며 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- [시작하기](#)
- [기본 사항](#)
- [작업](#)

시작하기

Hello Amazon Redshift

다음 코드 예시에서는 Amazon Redshift 사용을 시작하는 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
package main

import (
    "context"
    "fmt"

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

// main uses the AWS SDK for Go V2 to create a Redshift client
// and list up to 10 clusters in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
    ctx := context.Background()
    sdkConfig, err := config.LoadDefaultConfig(ctx)
    if err != nil {
        fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
        fmt.Println(err)
        return
    }
    redshiftClient := redshift.NewFromConfig(sdkConfig)
    count := 20
    fmt.Printf("Let's list up to %v clusters for your account.\n", count)
```

```

result, err := redshiftClient.DescribeClusters(ctx,
&redshift.DescribeClustersInput{
    MaxRecords: aws.Int32(int32(count)),
})
if err != nil {
    fmt.Printf("Couldn't list clusters for your account. Here's why: %v\n", err)
    return
}
if len(result.Clusters) == 0 {
    fmt.Println("You don't have any clusters!")
    return
}
for _, cluster := range result.Clusters {
    fmt.Printf("\t\t%v : %v\n", *cluster.ClusterIdentifier, *cluster.ClusterStatus)
}
}

```

- API에 대한 세부 정보는 AWS SDK for Go API 참조의 [DescribeClusters](#)를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- Redshift 클러스터를 생성합니다.
- 클러스터의 데이터베이스를 나열합니다.
- Movies라는 테이블을 생성합니다.
- Movies 테이블을 채웁니다.
- Movies 테이블을 연도별로 쿼리합니다.
- Redshift 클러스터를 수정합니다.
- Amazon Redshift 클러스터를 삭제합니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예제 리포지토리](#)에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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)
        }
    }
}
}
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 다음 주제를 참조하세요.
 - [CreateCluster](#)
 - [DescribeClusters](#)
 - [DescribeStatement](#)
 - [ExecuteStatement](#)
 - [GetStatementResult](#)
 - [ListDatabasesPaginator](#)
 - [ModifyCluster](#)

작업

CreateCluster

다음 코드 예시는 CreateCluster의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
}

```

- API에 대한 세부 정보는 AWS SDK for Go API 참조의 [CreateCluster](#)를 참조하세요.

DeleteCluster

다음 코드 예시는 DeleteCluster의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [DeleteCluster](#)를 참조하세요.

DescribeClusters

다음 코드 예시는 DescribeClusters의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.


```
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  
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [DescribeClusters](#)를 참조하세요.

ModifyCluster

다음 코드 예시는 ModifyCluster의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

- API에 대한 세부 정보는 AWS SDK for Go API 참조의 [ModifyCluster](#)를 참조하세요.

SDK for Go V2를 사용한 Amazon S3 예제

다음 코드 예제에서는 Amazon S3에서 AWS SDK for Go V2를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

기본 사항은 서비스 내에서 필수 작업을 수행하는 방법을 보여주는 코드 예제입니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- [시작하기](#)
- [기본 사항](#)
- [작업](#)
- [시나리오](#)

- [서버리스 예제](#)

시작하기

Hello Amazon S3

다음 코드 예제에서는 Amazon S3 사용을 시작하는 방법을 보여줍니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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)
    buckets, err := s3Client.ListBuckets(ctx, &s3.ListBucketsInput{
        MaxKeys: &int32{10},
    })
    if err != nil {
        fmt.Println("Couldn't list buckets: %v", err)
        return
    }
    for _, bucket := range buckets.Buckets {
        fmt.Println("Bucket Name:", *bucket.Name)
    }
}
```

```

}
s3Client := s3.NewFromConfig(sdkConfig)
count := 10
fmt.Printf("Let's list up to %v buckets for your account.\n", count)
result, err := s3Client.ListBuckets(ctx, &s3.ListBucketsInput{})
if err != nil {
    var ae smithy.APIError
    if errors.As(err, &ae) && ae.ErrorCode() == "AccessDenied" {
        fmt.Println("You don't have permission to list buckets for this account.")
    } else {
        fmt.Printf("Couldn't list buckets for your account. Here's why: %v\n", err)
    }
    return
}
if len(result.Buckets) == 0 {
    fmt.Println("You don't have any buckets!")
} else {
    if count > len(result.Buckets) {
        count = len(result.Buckets)
    }
    for _, bucket := range result.Buckets[:count] {
        fmt.Printf("\t\t%v\n", *bucket.Name)
    }
}
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [ListBuckets](#)를 참조하세요.

기본 사항

기본 사항 알아보기

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 버킷을 만들고 버킷에 파일을 업로드합니다.
- 버킷에서 객체를 다운로드합니다.
- 버킷의 하위 폴더에 객체를 복사합니다.
- 버킷의 객체를 나열합니다.
- 버킷 객체와 버킷을 삭제합니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

시나리오가 사용하는 버킷 및 객체 작업을 래핑하는 구조를 정의합니다.

```
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
}

```

S3 버킷과 객체를 다루는 방법을 보여주는 대화형 시나리오를 실행합니다.

```

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\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))
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 다음 항목을 참조하세요.
 - [CopyObject](#)
 - [CreateBucket](#)
 - [DeleteBucket](#)
 - [DeleteObjects](#)
 - [GetObject](#)
 - [ListObjectsV2](#)
 - [PutObject](#)

작업

CopyObject

다음 코드 예시는 CopyObject의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [CopyObject](#)를 참조하세요.

CreateBucket

다음 코드 예시는 CreateBucket의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

기본 구성으로 버킷을 생성합니다.

```

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
}
```

객체 잠금 기능이 있는 버킷을 만들고 버킷이 존재할 때까지 기다립니다.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [CreateBucket](#)을 참조하세요.

DeleteBucket

다음 코드 예시는 DeleteBucket의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배우보세요.

```

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
}

```

```
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [DeleteBucket](#)을 참조하세요.

DeleteObject

다음 코드 예시는 DeleteObject의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [DeleteObject](#)를 참조하세요.

DeleteObjects

다음 코드 예시는 DeleteObjects의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
            }
        }
        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 _, delObj := range delOut.Deleted {
            err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
                ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: delObj.Key},
                time.Minute)
            if err != nil {
                log.Printf("Failed attempt to wait for object %s to be deleted.\n",
                    *delObj.Key)
            } else {
                log.Printf("Deleted %s.\n", *delObj.Key)
            }
        }
    }
}
```

```

    }
  }
}
return err
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [DeleteObjects](#)를 참조하세요.

GetObject

다음 코드 예시는 GetObject의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [GetObject](#)를 참조하세요.

GetObjectLegalHold

다음 코드 예시는 GetObjectLegalHold의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [GetObjectLegalHold](#)를 참조하세요.

GetObjectLockConfiguration

다음 코드 예시는 GetObjectLockConfiguration의 사용 방법을 보여줍니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [GetObjectLockConfiguration](#)을 참조하세요.

GetObjectRetention

다음 코드 예시는 GetObjectRetention의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [GetObjectRetention](#)을 참조하세요.

HeadBucket

다음 코드 예시는 HeadBucket의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [HeadBucket](#)을 참조하세요.

ListBuckets

다음 코드 예시는 ListBuckets의 사용 방법을 보여줍니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [ListBuckets](#)를 참조하세요.

ListObjectVersions

다음 코드 예시는 ListObjectVersions의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [ListObjectVersions](#)를 참조하세요.

ListObjectsV2

다음 코드 예시는 ListObjectsV2의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [ListObjectsV2](#)를 참조하세요.

PutObject

다음 코드 예시는 PutObject의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

하위 수준 API를 사용하여 버킷에 객체를 추가합니다.

```
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
}

```

전송 관리자를 사용하여 버킷에 객체를 업로드합니다.

```

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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [PutObject](#)를 참조하세요.

PutObjectLegalHold

다음 코드 예시는 PutObjectLegalHold의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [PutObjectLegalHold](#)를 참조하세요.

PutObjectLockConfiguration

다음 코드 예시는 PutObjectLockConfiguration의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

버킷의 객체 잠금 구성을 설정합니다.

```

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
}

```

버킷의 기본 보존 기간을 설정합니다.

```

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
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [PutObjectLockConfiguration](#)을 참조하세요.

PutObjectRetention

다음 코드 예시는 PutObjectRetention의 사용 방법을 보여줍니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [PutObjectRetention](#)을 참조하세요.

시나리오

미리 서명된 URL 생성

다음 코드 예제에서는 Amazon S3에 대해 미리 서명된 URL을 생성하고 객체를 업로드하는 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

S3 미리 서명 작업을 래핑하는 함수를 만듭니다.

```

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
}

```

미리 서명된 URL을 생성하고 사용하여 S3 객체를 업로드, 다운로드, 삭제하는 대화형 예제를 실행합니다.

```

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))
}

```

예제가 HTTP 요청을 만드는 데 사용되는 HTTP 요청 래퍼를 정의합니다.

```

// 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)
}
```

Amazon S3 객체 잠그기

다음 코드 예시에는 S3 객체 잠금 기능을 사용하는 방법을 보여줍니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon S3 객체 잠금 기능을 시연하는 대화형 시나리오를 실행합니다.

```
import (
    "context"
    "fmt"
    "log"
    "strings"

    "s3_object_lock/actions"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/feature/s3/manager"
    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// ObjectLockScenario contains the steps to run the S3 Object Lock workflow.
type ObjectLockScenario struct {
    questioner demotools.IQuestioner
    resources  Resources
    s3Actions  *actions.S3Actions
    sdkConfig  aws.Config
}

// NewObjectLockScenario constructs a new ObjectLockScenario instance.
func NewObjectLockScenario(sdkConfig aws.Config, questioner demotools.IQuestioner)
    ObjectLockScenario {
    scenario := ObjectLockScenario{
        questioner: questioner,
        resources:  Resources{},
        s3Actions:  &actions.S3Actions{S3Client: s3.NewFromConfig(sdkConfig)},
        sdkConfig:  sdkConfig,
    }
}
```

```

scenario.s3Actions.S3Manager = manager.NewUploader(scenario.s3Actions.S3Client)
scenario.resources.init(scenario.s3Actions, questioner)
return scenario
}

type nameLocked struct {
    name    string
    locked  bool
}

var createInfo = []nameLocked{
    {"standard-bucket", false},
    {"lock-bucket", true},
    {"retention-bucket", false},
}

// CreateBuckets creates the S3 buckets required for the workflow.
func (scenario *ObjectLockScenario) CreateBuckets(ctx context.Context) {
    log.Println("Let's create some S3 buckets to use for this workflow.")
    success := false
    for !success {
        prefix := scenario.questioner.Ask(
            "This example creates three buckets. Enter a prefix to name your buckets\n(remember bucket names must be globally unique):")

        for _, info := range createInfo {
            log.Println(fmt.Sprintf("%s.%s", prefix, info.name))
            bucketName, err := scenario.s3Actions.CreateBucketWithLock(ctx, fmt.Sprintf("%s.%s", prefix, info.name), scenario.sdkConfig.Region, info.locked)
            if err != nil {
                switch err.(type) {
                    case *types.BucketAlreadyExists, *types.BucketAlreadyOwnedByYou:
                        log.Printf("Couldn't create bucket %s.\n", bucketName)
                    default:
                        panic(err)
                }
            }
            break
        }
        scenario.resources.demoBuckets[info.name] = &DemoBucket{
            name:        bucketName,
            objectKeys: []string{},
        }
        log.Printf("Created bucket %s.\n", bucketName)
    }
}

```

```

    if len(scenario.resources.demoBuckets) < len(createInfo) {
        scenario.resources.deleteBuckets(ctx)
    } else {
        success = true
    }
}

log.Println("S3 buckets created.")
log.Println(strings.Repeat("-", 88))
}

// EnableLockOnBucket enables object locking on an existing bucket.
func (scenario *ObjectLockScenario) EnableLockOnBucket(ctx context.Context) {
    log.Println("\nA bucket can be configured to use object locking.")
    scenario.questioner.Ask("Press Enter to continue.")

    var err error
    bucket := scenario.resources.demoBuckets["retention-bucket"]
    err = scenario.s3Actions.EnableObjectLockOnBucket(ctx, bucket.name)
    if err != nil {
        switch err.(type) {
        case *types.NoSuchBucket:
            log.Printf("Couldn't enable object locking on bucket %s.\n", bucket.name)
        default:
            panic(err)
        }
    } else {
        log.Printf("Object locking enabled on bucket %s.", bucket.name)
    }

    log.Println(strings.Repeat("-", 88))
}

// SetDefaultRetentionPolicy sets a default retention governance policy on a bucket.
func (scenario *ObjectLockScenario) SetDefaultRetentionPolicy(ctx context.Context) {
    log.Println("\nA bucket can be configured to use object locking with a default
    retention period.")

    bucket := scenario.resources.demoBuckets["retention-bucket"]
    retentionPeriod := scenario.questioner.AskInt("Enter the default retention period
    in days: ")

```



```

err := scenario.s3Actions.ModifyDefaultBucketRetention(ctx,
bucket.name, types.ObjectLockEnabledEnabled, int32(retentionPeriod),
types.ObjectLockRetentionModeGovernance)
if err != nil {
    switch err.(type) {
    case *types.NoSuchBucket:
        log.Printf("Couldn't configure a default retention period on bucket %s.\n",
bucket.name)
    default:
        panic(err)
    }
} else {
    log.Printf("Default retention policy set on bucket %s with %d day retention
period.", bucket.name, retentionPeriod)
    bucket.retentionEnabled = true
}

log.Println(strings.Repeat("-", 88))
}

// UploadTestObjects uploads test objects to the S3 buckets.
func (scenario *ObjectLockScenario) UploadTestObjects(ctx context.Context) {
    log.Println("Uploading test objects to S3 buckets.")

    for _, info := range createInfo {
        bucket := scenario.resources.demoBuckets[info.name]
        for i := 0; i < 2; i++ {
            key, err := scenario.s3Actions.UploadObject(ctx, bucket.name,
fmt.Sprintf("example-%d", i),
            fmt.Sprintf("Example object content #%d in bucket %s.", i, bucket.name))
            if err != nil {
                switch err.(type) {
                case *types.NoSuchBucket:
                    log.Printf("Couldn't upload %s to bucket %s.\n", key, bucket.name)
                default:
                    panic(err)
                }
            } else {
                log.Printf("Uploaded %s to bucket %s.\n", key, bucket.name)
                bucket.objectKeys = append(bucket.objectKeys, key)
            }
        }
    }
}

```

```

scenario.questioner.Ask("Test objects uploaded. Press Enter to continue.")
log.Println(strings.Repeat("-", 88))
}

// SetObjectLockConfigurations sets object lock configurations on the test objects.
func (scenario *ObjectLockScenario) SetObjectLockConfigurations(ctx context.Context)
{
    log.Println("Now let's set object lock configurations on individual objects.")

    buckets := []*DemoBucket{scenario.resources.demoBuckets["lock-bucket"],
    scenario.resources.demoBuckets["retention-bucket"]}
    for _, bucket := range buckets {
        for index, objKey := range bucket.objectKeys {
            switch index {
            case 0:
                if scenario.questioner.AskBool(fmt.Sprintf("\nDo you want to add a legal hold to
                %s in %s (y/n)? ", objKey, bucket.name), "y") {
                    err := scenario.s3Actions.PutObjectLegalHold(ctx, bucket.name, objKey, "",
                    types.ObjectLockLegalHoldStatusOn)
                    if err != nil {
                        switch err.(type) {
                        case *types.NoSuchKey:
                            log.Printf("Couldn't set legal hold on %s.\n", objKey)
                        default:
                            panic(err)
                        }
                    } else {
                        log.Printf("Legal hold set on %s.\n", objKey)
                    }
                }
            case 1:
                q := fmt.Sprintf("\nDo you want to add a 1 day Governance retention period to %s
                in %s?\n"+
                "Reminder: Only a user with the s3:BypassGovernanceRetention permission is able
                to delete this object\n"+
                "or its bucket until the retention period has expired. (y/n) ", objKey,
                bucket.name)
                if scenario.questioner.AskBool(q, "y") {
                    err := scenario.s3Actions.PutObjectRetention(ctx, bucket.name, objKey,
                    types.ObjectLockRetentionModeGovernance, 1)
                    if err != nil {
                        switch err.(type) {
                        case *types.NoSuchKey:

```

```

        log.Printf("Couldn't set retention period on %s in %s.\n", objKey,
bucket.name)
        default:
            panic(err)
        }
    } else {
        log.Printf("Retention period set to 1 for %s.", objKey)
        bucket.retentionEnabled = true
    }
}
}
}
}
log.Println(strings.Repeat("-", 88))
}

const (
    ListAll = iota
    DeleteObject
    DeleteRetentionObject
    OverwriteObject
    ViewRetention
    ViewLegalHold
    Finish
)

// InteractWithObjects allows the user to interact with the objects and test the
// object lock configurations.
func (scenario *ObjectLockScenario) InteractWithObjects(ctx context.Context) {
    log.Println("Now you can interact with the objects to explore the object lock
configurations.")
    interactiveChoices := []string{
        "List all objects and buckets.",
        "Attempt to delete an object.",
        "Attempt to delete an object with retention period bypass.",
        "Attempt to overwrite a file.",
        "View the retention settings for an object.",
        "View the legal hold settings for an object.",
        "Finish the workflow."}

    choice := ListAll
    for choice != Finish {
        objList := scenario.GetAllObjects(ctx)
        objChoices := scenario.makeObjectChoiceList(objList)

```

```

    choice = scenario.questioner.AskChoice("Choose an action from the menu:\n",
interactiveChoices)
    switch choice {
    case ListAll:
        log.Println("The current objects in the example buckets are:")
        for _, objChoice := range objChoices {
            log.Println("\t", objChoice)
        }
    case DeleteObject, DeleteRetentionObject:
        objChoice := scenario.questioner.AskChoice("Enter the number of the object to
delete:\n", objChoices)
        obj := objList[objChoice]
        deleted, err := scenario.s3Actions.DeleteObject(ctx, obj.bucket, obj.key,
obj.versionId, choice == DeleteRetentionObject)
        if err != nil {
            switch err.(type) {
            case *types.NoSuchKey:
                log.Println("Nothing to delete.")
            default:
                panic(err)
            }
        } else if deleted {
            log.Printf("Object %s deleted.\n", obj.key)
        }
    case OverwriteObject:
        objChoice := scenario.questioner.AskChoice("Enter the number of the object to
overwrite:\n", objChoices)
        obj := objList[objChoice]
        _, err := scenario.s3Actions.UploadObject(ctx, obj.bucket, obj.key,
fmt.Sprintf("New content in object %s.", obj.key))
        if err != nil {
            switch err.(type) {
            case *types.NoSuchBucket:
                log.Println("Couldn't upload to nonexistent bucket.")
            default:
                panic(err)
            }
        } else {
            log.Printf("Uploaded new content to object %s.\n", obj.key)
        }
    case ViewRetention:
        objChoice := scenario.questioner.AskChoice("Enter the number of the object to
view:\n", objChoices)
        obj := objList[objChoice]

```

```

    retention, err := scenario.s3Actions.GetObjectRetention(ctx, obj.bucket, obj.key)
    if err != nil {
        switch err.(type) {
        case *types.NoSuchKey:
            log.Printf("Can't get retention configuration for %s.\n", obj.key)
        default:
            panic(err)
        }
    } else if retention != nil {
        log.Printf("Object %s has retention mode %s until %v.\n", obj.key,
retention.Mode, retention.RetainUntilDate)
    } else {
        log.Printf("Object %s does not have object retention configured.\n", obj.key)
    }
    case ViewLegalHold:
        objChoice := scenario.questioner.AskChoice("Enter the number of the object to
view:\n", objChoices)
        obj := objList[objChoice]
        legalHold, err := scenario.s3Actions.GetObjectLegalHold(ctx, obj.bucket, obj.key,
obj.versionId)
        if err != nil {
            switch err.(type) {
            case *types.NoSuchKey:
                log.Printf("Can't get legal hold configuration for %s.\n", obj.key)
            default:
                panic(err)
            }
        } else if legalHold != nil {
            log.Printf("Object %s has legal hold %v.", obj.key, *legalHold)
        } else {
            log.Printf("Object %s does not have legal hold configured.", obj.key)
        }
    case Finish:
        log.Println("Let's clean up.")
    }
    log.Println(strings.Repeat("-", 88))
}
}

type BucketKeyVersionId struct {
    bucket    string
    key       string
    versionId string
}

```

```
// GetAllObjects gets the object versions in the example S3 buckets and returns them
// in a flattened list.
func (scenario *ObjectLockScenario) GetAllObjects(ctx context.Context)
[]BucketKeyVersionId {
    var objectList []BucketKeyVersionId
    for _, info := range createInfo {
        bucket := scenario.resources.demoBuckets[info.name]
        versions, err := scenario.s3Actions.ListObjectVersions(ctx, bucket.name)
        if err != nil {
            switch err.(type) {
            case *types.NoSuchBucket:
                log.Printf("Couldn't get object versions for %s.\n", bucket.name)
            default:
                panic(err)
            }
        } else {
            for _, version := range versions {
                objectList = append(objectList,
                    BucketKeyVersionId{bucket: bucket.name, key: *version.Key, versionId:
                    *version.VersionId})
            }
        }
    }
    return objectList
}

// makeObjectChoiceList makes the object version list into a list of strings that
// are displayed
// as choices.
func (scenario *ObjectLockScenario) makeObjectChoiceList(bucketObjects
[]BucketKeyVersionId) []string {
    choices := make([]string, len(bucketObjects))
    for i := 0; i < len(bucketObjects); i++ {
        choices[i] = fmt.Sprintf("%s in %s with VersionId %s.",
            bucketObjects[i].key, bucketObjects[i].bucket, bucketObjects[i].versionId)
    }
    return choices
}

// Run runs the S3 Object Lock scenario.
func (scenario *ObjectLockScenario) Run(ctx context.Context) {
    defer func() {
        if r := recover(); r != nil {
```

```

    log.Println("Something went wrong with the demo.")
    _, isMock := scenario.questioner.(*demotools.MockQuestioner)
    if isMock || scenario.questioner.AskBool("Do you want to see the full error
message (y/n)?", "y") {
        log.Println(r)
    }
    scenario.resources.Cleanup(ctx)
}
}()

log.Println(strings.Repeat("-", 88))
log.Println("Welcome to the Amazon S3 Object Lock Feature Scenario.")
log.Println(strings.Repeat("-", 88))

scenario.CreateBuckets(ctx)
scenario.EnableLockOnBucket(ctx)
scenario.SetDefaultRetentionPolicy(ctx)
scenario.UploadTestObjects(ctx)
scenario.SetObjectLockConfigurations(ctx)
scenario.InteractWithObjects(ctx)

scenario.resources.Cleanup(ctx)

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

```

이 예시에서 사용되는 S3 작업을 래핑하는 구조를 정의합니다.

```

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 _, delObj := range delOut.Deleted {
        err = s3.NewObjectNotExistsWaiter(actor.S3Client).Wait(
            ctx, &s3.HeadObjectInput{Bucket: aws.String(bucket), Key: delObj.Key},
            time.Minute)
        if err != nil {
            log.Printf("Failed attempt to wait for object %s to be deleted.\n",
                *delObj.Key)
        } else {
            log.Printf("Deleted %s.\n", *delObj.Key)
        }
    }
}
return err
}

```

리소스를 정리합니다.

```
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)
    }
}
}
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 다음 주제를 참조하세요.
 - [GetObjectLegalHold](#)
 - [GetObjectLockConfiguration](#)
 - [GetObjectRetention](#)
 - [PutObjectLegalHold](#)
 - [PutObjectLockConfiguration](#)
 - [PutObjectRetention](#)

대용량 파일 업로드 또는 다운로드

다음 코드 예제는 Amazon S3에 대용량 파일을 업로드하고 Amazon S3에서 대용량 파일을 다운로드 하는 방법을 보여줍니다.

자세한 내용은 [멀티파트 업로드를 사용하여 객체 업로드](#)를 참조하세요.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

업로드 및 다운로드 관리자를 사용하여 데이터를 여러 부분으로 나누고 동시에 전송하는 함수를 생성합니다.

```
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
}

```

컨텍스트에서 업로드 및 다운로드 관리자를 사용하는 방법을 보여주는 대화형 시나리오를 실행합니다.

```

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))
}

```

서버리스 예제

Amazon S3 트리거를 사용하여 Lambda 함수 간접 호출

다음 코드 예제는 S3 버킷에 객체를 업로드하여 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 해당 함수는 이벤트 파라미터에서 S3 버킷 이름과 객체 키를 검색하고 Amazon S3 API를 호출하여 객체의 콘텐츠 유형을 검색하고 로깅합니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [서버리스 예제](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Go를 사용하여 Lambda로 S3 이벤트를 사용합니다.

```
// 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)
}

```

SDK for Go V2를 사용한 Amazon SNS 예제

다음 코드 예제에서는 Amazon SNS에서 AWS SDK for Go V2를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- [시작하기](#)
- [작업](#)

- [시나리오](#)
- [서버리스 예제](#)

시작하기

Hello Amazon SNS

다음 코드 예제에서는 Amazon SNS 사용을 시작하는 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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)
    }
}
}
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [ListTopics](#)를 참조하세요.

작업

CreateTopic

다음 코드 예시는 CreateTopic의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [CreateTopic](#)을 참조하세요.

DeleteTopic

다음 코드 예시는 DeleteTopic의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [DeleteTopic](#)을 참조하세요.

ListTopics

다음 코드 예시는 ListTopics의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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.
```


SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [Publish](#)를 참조하세요.

Subscribe

다음 코드 예시는 Subscribe의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

선택적 필터를 사용하여 주제 대기열을 구독하세요.

```

import (
    "context"
    "encoding/json"
    "log"

```

```

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

// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
// actions
// used in the examples.
type SnsActions struct {
    SnsClient *sns.Client
}

// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
// policy
// so that messages are only sent to the queue when the message has the specified
// attributes.
func (actor SnsActions) SubscribeQueue(ctx context.Context, topicArn string,
    queueArn string, filterMap map[string][]string) (string, error) {
    var subscriptionArn string
    var attributes map[string]string
    if filterMap != nil {
        filterBytes, err := json.Marshal(filterMap)
        if err != nil {
            log.Printf("Couldn't create filter policy, here's why: %v\n", err)
            return "", err
        }
        attributes = map[string]string{"FilterPolicy": string(filterBytes)}
    }
    output, err := actor.SnsClient.Subscribe(ctx, &sns.SubscribeInput{
        Protocol:          aws.String("sqs"),
        TopicArn:          aws.String(topicArn),
        Attributes:         attributes,
        Endpoint:           aws.String(queueArn),
        ReturnSubscriptionArn: true,
    })
    if err != nil {
        log.Printf("Couldn't subscribe queue %v to topic %v. Here's why: %v\n",
            queueArn, topicArn, err)
    } else {
        subscriptionArn = *output.SubscriptionArn
    }
}

```

```

}

return subscriptionArn, err
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [Subscribe](#)를 참조하십시오.

시나리오

대기열에 메시지 게시

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 주제(FIFO 또는 비 FIFO)를 생성합니다.
- 필터 적용 옵션을 사용하여 여러 개의 대기열로 주제를 구독합니다.
- 주제에 메시지를 게시합니다.
- 대기열에서 받은 메시지를 폴링합니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```

import (
    "context"
    "encoding/json"
    "fmt"
    "log"
    "strings"
    "topics_and_queues/actions"

    "github.com/aws/aws-sdk-go-v2/aws"

```

```

"github.com/aws/aws-sdk-go-v2/service/sns"
"github.com/aws/aws-sdk-go-v2/service/sqs"
"github.com/aws/aws-sdk-go-v2/service/sqs/types"
"github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

const FIFO_SUFFIX = ".fifo"
const TONE_KEY = "tone"

var ToneChoices = []string{"cheerful", "funny", "serious", "sincere"}

// MessageBody is used to deserialize the body of a message from a JSON string.
type MessageBody struct {
    Message string
}

// ScenarioRunner separates the steps of this scenario into individual functions so
// that
// they are simpler to read and understand.
type ScenarioRunner struct {
    questioner demotools.IQuestioner
    snsActor   *actions.SnsActions
    sqsActor   *actions.SqsActions
}

func (runner ScenarioRunner) CreateTopic(ctx context.Context) (string, string, bool,
bool) {
    log.Println("SNS topics can be configured as FIFO (First-In-First-Out) or standard.
\n" +
        "FIFO topics deliver messages in order and support deduplication and message
filtering.")
    isFifoTopic := runner.questioner.AskBool("\nWould you like to work with FIFO
topics? (y/n) ", "y")

    contentBasedDeduplication := false
    if isFifoTopic {
        log.Println(strings.Repeat("-", 88))
        log.Println("Because you have chosen a FIFO topic, deduplication is supported.\n"
+
            "Deduplication IDs are either set in the message or are automatically generated
\n" +
            "from content using a hash function. If a message is successfully published to\n"
+
            "an SNS FIFO topic, any message published and determined to have the same\n" +

```

```

    "deduplication ID, within the five-minute deduplication interval, is accepted\n"
+
    "but not delivered. For more information about deduplication, see:\n" +
    "\thttps://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.")
contentBasedDeduplication = runner.questioner.AskBool(
    "\nDo you want to use content-based deduplication instead of entering a
deduplication ID? (y/n) ", "y")
}
log.Println(strings.Repeat("-", 88))

topicName := runner.questioner.Ask("Enter a name for your SNS topic. ")
if isFifoTopic {
    topicName = fmt.Sprintf("%v%v", topicName, FIFO_SUFFIX)
    log.Printf("Because you have selected a FIFO topic, '%v' must be appended to\n"+
        "the topic name.", FIFO_SUFFIX)
}

topicArn, err := runner.snsActor.CreateTopic(ctx, topicName, isFifoTopic,
contentBasedDeduplication)
if err != nil {
    panic(err)
}
log.Printf("Your new topic with the name '%v' and Amazon Resource Name (ARN) \n"+
    "'%v' has been created.", topicName, topicArn)

return topicName, topicArn, isFifoTopic, contentBasedDeduplication
}

func (runner ScenarioRunner) CreateQueue(ctx context.Context, ordinal string,
isFifoTopic bool) (string, string) {
    queueName := runner.questioner.Ask(fmt.Sprintf("Enter a name for the %v SQS queue.
", ordinal))
    if isFifoTopic {
        queueName = fmt.Sprintf("%v%v", queueName, FIFO_SUFFIX)
        if ordinal == "first" {
            log.Printf("Because you are creating a FIFO SQS queue, '%v' must "+
                "be appended to the queue name.\n", FIFO_SUFFIX)
        }
    }
    queueUrl, err := runner.sqsActor.CreateQueue(ctx, queueName, isFifoTopic)
    if err != nil {
        panic(err)
    }
    log.Printf("Your new SQS queue with the name '%v' and the queue URL "+

```



```

    "'%v' has been created.", queueName, queueUrl)

    return queueName, queueUrl
}

func (runner ScenarioRunner) SubscribeQueueToTopic(
    ctx context.Context, queueName string, queueUrl string, topicName string, topicArn
    string, ordinal string,
    isFifoTopic bool) (string, bool) {

    queueArn, err := runner.sqsActor.GetQueueArn(ctx, queueUrl)
    if err != nil {
        panic(err)
    }
    log.Printf("The ARN of your queue is: %v.\n", queueArn)

    err = runner.sqsActor.AttachSendMessagePolicy(ctx, queueUrl, queueArn, topicArn)
    if err != nil {
        panic(err)
    }
    log.Println("Attached an IAM policy to the queue so the SNS topic can send " +
        "messages to it.")
    log.Println(strings.Repeat("-", 88))

    var filterPolicy map[string][]string
    if isFifoTopic {
        if ordinal == "first" {
            log.Println("Subscriptions to a FIFO topic can have filters.\n" +
                "If you add a filter to this subscription, then only the filtered messages\n" +
                "will be received in the queue.\n" +
                "For information about message filtering, see\n" +
                "\thttps://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n" +
                "For this example, you can filter messages by a \"tone\" attribute.")
        }

        wantFiltering := runner.questioner.AskBool(
            fmt.Sprintf("Do you want to filter messages that are sent to \"%v\"\n"+
                "from the %v topic? (y/n) ", queueName, topicName), "y")
        if wantFiltering {
            log.Println("You can filter messages by one or more of the following \"tone\"
            attributes.")

            var toneSelections []string
            askAboutTones := true

```

```

    for askAboutTones {
        toneIndex := runner.questioner.AskChoice(
            "Enter the number of the tone you want to filter by:\n", ToneChoices)
        toneSelections = append(toneSelections, ToneChoices[toneIndex])
        askAboutTones = runner.questioner.AskBool("Do you want to add another tone to
the filter? (y/n) ", "y")
    }
    log.Printf("Your subscription will be filtered to only pass the following tones:
%v\n", toneSelections)
    filterPolicy = map[string][]string{TONE_KEY: toneSelections}
}

subscriptionArn, err := runner.snsActor.SubscribeQueue(ctx, topicArn, queueArn,
filterPolicy)
if err != nil {
    panic(err)
}
log.Printf("The queue %v is now subscribed to the topic %v with the subscription
ARN %v.\n",
    queueName, topicName, subscriptionArn)

return subscriptionArn, filterPolicy != nil
}

func (runner ScenarioRunner) PublishMessages(ctx context.Context, topicArn string,
isFifoTopic bool, contentBasedDeduplication bool, usingFilters bool) {
    var message string
    var groupId string
    var dedupId string
    var toneSelection string
    publishMore := true
    for publishMore {
        groupId = ""
        dedupId = ""
        toneSelection = ""
        message = runner.questioner.Ask("Enter a message to publish: ")
        if isFifoTopic {
            log.Println("Because you are using a FIFO topic, you must set a message group ID.
\n" +
                "All messages within the same group will be received in the order they were
published.")
            groupId = runner.questioner.Ask("Enter a message group ID: ")
            if !contentBasedDeduplication {

```

```

    log.Println("Because you are not using content-based deduplication,\n" +
        "you must enter a deduplication ID.")
    dedupId = runner.questioner.Ask("Enter a deduplication ID: ")
}
}
if usingFilters {
    if runner.questioner.AskBool("Add a tone attribute so this message can be
filtered? (y/n) ", "y") {
        toneIndex := runner.questioner.AskChoice(
            "Enter the number of the tone you want to filter by:\n", ToneChoices)
        toneSelection = ToneChoices[toneIndex]
    }
}

err := runner.snsActor.Publish(ctx, topicArn, message, groupId, dedupId, TONE_KEY,
toneSelection)
if err != nil {
    panic(err)
}
log.Println(("Your message was published.))

publishMore = runner.questioner.AskBool("Do you want to publish another message?
(y/n) ", "y")
}
}

func (runner ScenarioRunner) PollForMessages(ctx context.Context, queueUrls
[]string) {
    log.Println("Polling queues for messages...")
    for _, queueUrl := range queueUrls {
        var messages []types.Message
        for {
            currentMsgs, err := runner.sqsActor.GetMessages(ctx, queueUrl, 10, 1)
            if err != nil {
                panic(err)
            }
            if len(currentMsgs) == 0 {
                break
            }
            messages = append(messages, currentMsgs...)
        }
        if len(messages) == 0 {
            log.Printf("No messages were received by queue %v.\n", queueUrl)
        } else if len(messages) == 1 {

```

```

    log.Printf("One message was received by queue %v:\n", queueUrl)

} else {
    log.Printf("%v messages were received by queue %v:\n", len(messages), queueUrl)
}
for msgIndex, message := range messages {
    messageBody := MessageBody{}
    err := json.Unmarshal([]byte(*message.Body), &messageBody)
    if err != nil {
        panic(err)
    }
    log.Printf("Message %v: %v\n", msgIndex+1, messageBody.Message)
}

if len(messages) > 0 {
    log.Printf("Deleting %v messages from queue %v.\n", len(messages), queueUrl)
    err := runner.sqsActor.DeleteMessages(ctx, queueUrl, messages)
    if err != nil {
        panic(err)
    }
}
}
}

// RunTopicsAndQueuesScenario is an interactive example that shows you how to use
// the
// AWS SDK for Go to create and use Amazon SNS topics and Amazon SQS queues.
//
// 1. Create a topic (FIFO or non-FIFO).
// 2. Subscribe several queues to the topic with an option to apply a filter.
// 3. Publish messages to the topic.
// 4. Poll the queues for messages received.
// 5. Delete the topic and the queues.
//
// This example creates service clients from the specified sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
// example.
// This package can be found in the ../..demotools folder of this repo.
func RunTopicsAndQueuesScenario(
    ctx context.Context, sdkConfig aws.Config, questioner demotools.IQuestioner) {
    resources := Resources{}
    defer func() {

```

```

    if r := recover(); r != nil {
        log.Println("Something went wrong with the demo.\n" +
            "Cleaning up any resources that were created...")
        resources.Cleanup(ctx)
    }
}()
queueCount := 2

log.Println(strings.Repeat("-", 88))
log.Printf("Welcome to messaging with topics and queues.\n\n"+
    "In this scenario, you will create an SNS topic and subscribe %v SQS queues to the\n"+
    "topic. You can select from several options for configuring the topic and the\n"+
    "subscriptions for the queues. You can then post to the topic and see the results\n"+
    "in the queues.\n", queueCount)

log.Println(strings.Repeat("-", 88))

runner := ScenarioRunner{
    questioner: questioner,
    snsActor:   &actions.SnsActions{SnsClient: sns.NewFromConfig(sdkConfig)},
    sqsActor:   &actions.SqsActions{SqsClient: sqs.NewFromConfig(sdkConfig)},
}
resources.snsActor = runner.snsActor
resources.sqsActor = runner.sqsActor

topicName, topicArn, isFifoTopic, contentBasedDeduplication :=
runner.CreateTopic(ctx)
resources.topicArn = topicArn
log.Println(strings.Repeat("-", 88))

log.Printf("Now you will create %v SQS queues and subscribe them to the topic.\n",
queueCount)
ordinals := []string{"first", "next"}
usingFilters := false
for _, ordinal := range ordinals {
    queueName, queueUrl := runner.CreateQueue(ctx, ordinal, isFifoTopic)
    resources.queueUrls = append(resources.queueUrls, queueUrl)

    _, filtering := runner.SubscribeQueueToTopic(ctx, queueName, queueUrl, topicName,
topicArn, ordinal, isFifoTopic)
    usingFilters = usingFilters || filtering
}

```

```

log.Println(strings.Repeat("-", 88))
runner.PublishMessages(ctx, topicArn, isFifoTopic, contentBasedDeduplication,
usingFilters)
log.Println(strings.Repeat("-", 88))
runner.PollForMessages(ctx, resources.queueUrls)

log.Println(strings.Repeat("-", 88))

wantCleanup := questioner.AskBool("Do you want to remove all AWS resources created
for this scenario? (y/n) ", "y")
if wantCleanup {
    log.Println("Cleaning up resources...")
    resources.Cleanup(ctx)
}

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

```

이 예시에서 사용되는 Amazon SNS 작업을 래핑하는 구조를 정의합니다.

```

import (
    "context"
    "encoding/json"
    "log"

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

// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
actions
// used in the examples.
type SnsActions struct {
    SnsClient *sns.Client
}

```

```
// CreateTopic creates an Amazon SNS topic with the specified name. You can
// optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
// based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(ctx context.Context, topicName string,
    isFifoTopic bool, contentBasedDeduplication bool) (string, error) {
    var topicArn string
    topicAttributes := map[string]string{}
    if isFifoTopic {
        topicAttributes["FifoTopic"] = "true"
    }
    if contentBasedDeduplication {
        topicAttributes["ContentBasedDeduplication"] = "true"
    }
    topic, err := actor.SnsClient.CreateTopic(ctx, &sns.CreateTopicInput{
        Name:      aws.String(topicName),
        Attributes: topicAttributes,
    })
    if err != nil {
        log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
    } else {
        topicArn = *topic.TopicArn
    }

    return topicArn, err
}

// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(ctx context.Context, topicArn string) error {
    _, err := actor.SnsClient.DeleteTopic(ctx, &sns.DeleteTopicInput{
        TopicArn: aws.String(topicArn)})
    if err != nil {
        log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
    }
    return err
}
```

```
// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
// policy
// so that messages are only sent to the queue when the message has the specified
// attributes.
func (actor SnsActions) SubscribeQueue(ctx context.Context, topicArn string,
queueArn string, filterMap map[string][]string) (string, error) {
    var subscriptionArn string
    var attributes map[string]string
    if filterMap != nil {
        filterBytes, err := json.Marshal(filterMap)
        if err != nil {
            log.Printf("Couldn't create filter policy, here's why: %v\n", err)
            return "", err
        }
        attributes = map[string]string{"FilterPolicy": string(filterBytes)}
    }
    output, err := actor.SnsClient.Subscribe(ctx, &sns.SubscribeInput{
        Protocol:          aws.String("sqs"),
        TopicArn:          aws.String(topicArn),
        Attributes:         attributes,
        Endpoint:          aws.String(queueArn),
        ReturnSubscriptionArn: true,
    })
    if err != nil {
        log.Printf("Couldn't subscribe queue %v to topic %v. Here's why: %v\n",
            queueArn, topicArn, err)
    } else {
        subscriptionArn = *output.SubscriptionArn
    }

    return subscriptionArn, err
}

// Publish publishes a message to an Amazon SNS topic. The message is then sent to
// all
// subscribers. When the topic is a FIFO topic, the message must also contain a
// group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
// value
// filter attribute can be specified so that the message can be filtered according
// to
```



```
// a filter policy.
func (actor SnsActions) Publish(ctx context.Context, topicArn string, message
string, groupId string, dedupId string, filterKey string, filterValue string) error
{
    publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
aws.String(message)}
    if groupId != "" {
        publishInput.MessageGroupId = aws.String(groupId)
    }
    if dedupId != "" {
        publishInput.MessageDeduplicationId = aws.String(dedupId)
    }
    if filterKey != "" && filterValue != "" {
        publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
            filterKey: {DataType: aws.String("String"), StringValue:
aws.String(filterValue)},
        }
    }
    _, err := actor.SnsClient.Publish(ctx, &publishInput)
    if err != nil {
        log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn, err)
    }
    return err
}
```

이 예시에서 사용되는 Amazon SQS 작업을 래핑하는 구조를 정의합니다.

```
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
}

```

리소스를 정리합니다.

```

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)
        }
    }
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 다음 주제를 참조하세요.
 - [CreateQueue](#)
 - [CreateTopic](#)
 - [DeleteMessageBatch](#)

- [DeleteQueue](#)
- [DeleteTopic](#)
- [GetQueueAttributes](#)
- [Publish](#)
- [ReceiveMessage](#)
- [SetQueueAttributes](#)
- [Subscribe](#)
- [Unsubscribe](#)

서버리스 예제

Amazon SNS 트리거를 사용하여 Lambda 함수 간접 호출

다음 코드 예제에서는 SNS 주제의 메시지를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 함수는 이벤트 파라미터에서 메시지를 검색하고 각 메시지의 내용을 로깅합니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [서버리스 예제](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Go를 사용하여 Lambda로 SNS 이벤트를 사용합니다.

```
// 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)
}
```

SDK for Go V2를 사용한 Amazon SQS 예제

다음 코드 예제에서는 Amazon SQS에서 AWS SDK for Go V2를 사용하여 작업을 수행하고 일반적인 시나리오를 구현하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 직접 호출하는 방법을 보여주며, 관련 시나리오의 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 또는 다른 AWS 서비스와 결합된 상태에서 여러 함수를 직접적으로 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예제입니다.

각 예시에는 전체 소스 코드에 대한 링크가 포함되어 있으며, 여기에서 컨텍스트에 맞춰 코드를 설정하고 실행하는 방법에 대한 지침을 찾을 수 있습니다.

주제

- [시작하기](#)
- [작업](#)
- [시나리오](#)
- [서버리스 예제](#)

시작하기

Hello Amazon SNS

다음 코드 예제에서는 Amazon SQS 사용을 시작하는 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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)
    }
}
}

```

- API에 대한 세부 정보는 AWS SDK for Go API 참조의 [ListQueues](#)를 참조하세요.

작업

CreateQueue

다음 코드 예시는 CreateQueue의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [CreateQueue](#)를 참조하세요.

DeleteMessageBatch

다음 코드 예시는 DeleteMessageBatch의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [DeleteMessageBatch](#)를 참조하세요.

DeleteQueue

다음 코드 예시는 DeleteQueue의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
}

```

- API 세부 정보는 AWS SDK for Go API 참조의 [DeleteQueue](#)를 참조하세요.

GetQueueAttributes

다음 코드 예시는 GetQueueAttributes의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [GetQueueAttributes](#)를 참조하세요.

ListQueues

다음 코드 예시는 ListQueues의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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)
        }
    }
}
```



```

    }
  }
}

```

- API에 대한 세부 정보는 AWS SDK for Go API 참조의 [ListQueues](#)를 참조하세요.

ReceiveMessage

다음 코드 예시는 ReceiveMessage의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```

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
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 [ReceiveMessage](#)를 참조하세요.

SetQueueAttributes

다음 코드 예시는 SetQueueAttributes의 사용 방법을 보여줍니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

```
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

```

- API 세부 정보는 AWS SDK for Go API 참조의 [SetQueueAttributes](#)를 참조하세요.

시나리오

대기열에 메시지 게시

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 주제(FIFO 또는 비 FIFO)를 생성합니다.
- 필터 적용 옵션을 사용하여 여러 개의 대기열로 주제를 구독합니다.
- 주제에 메시지를 게시합니다.
- 대기열에서 받은 메시지를 폴링합니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [AWS 코드 예 리포지토리](#)에서 전체 예를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
import (
    "context"
    "encoding/json"
    "fmt"
    "log"
    "strings"
    "topics_and_queues/actions"

    "github.com/aws/aws-sdk-go-v2/aws"
    "github.com/aws/aws-sdk-go-v2/service/sns"
    "github.com/aws/aws-sdk-go-v2/service/sqs"
    "github.com/aws/aws-sdk-go-v2/service/sqs/types"
    "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

const FIFO_SUFFIX = ".fifo"
const TONE_KEY = "tone"

var ToneChoices = []string{"cheerful", "funny", "serious", "sincere"}

// MessageBody is used to deserialize the body of a message from a JSON string.
type MessageBody struct {
    Message string
}

// ScenarioRunner separates the steps of this scenario into individual functions so
// that
// they are simpler to read and understand.
type ScenarioRunner struct {
    questioner demotools.IQuestioner
    snsActor   *actions.SnsActions
    sqsActor   *actions.SqsActions
}
```

```

}

func (runner ScenarioRunner) CreateTopic(ctx context.Context) (string, string, bool,
bool) {
    log.Println("SNS topics can be configured as FIFO (First-In-First-Out) or standard.
\n" +
        "FIFO topics deliver messages in order and support deduplication and message
filtering.")
    isFifoTopic := runner.questioner.AskBool("\nWould you like to work with FIFO
topics? (y/n) ", "y")

    contentBasedDeduplication := false
    if isFifoTopic {
        log.Println(strings.Repeat("-", 88))
        log.Println("Because you have chosen a FIFO topic, deduplication is supported.\n"
+
            "Deduplication IDs are either set in the message or are automatically generated
\n" +
            "from content using a hash function. If a message is successfully published to\n"
+
            "an SNS FIFO topic, any message published and determined to have the same\n" +
            "deduplication ID, within the five-minute deduplication interval, is accepted\n"
+
            "but not delivered. For more information about deduplication, see:\n" +
            "\thttps://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.")
        contentBasedDeduplication = runner.questioner.AskBool(
            "\nDo you want to use content-based deduplication instead of entering a
deduplication ID? (y/n) ", "y")
    }
    log.Println(strings.Repeat("-", 88))

    topicName := runner.questioner.Ask("Enter a name for your SNS topic. ")
    if isFifoTopic {
        topicName = fmt.Sprintf("%v%v", topicName, FIFO_SUFFIX)
        log.Printf("Because you have selected a FIFO topic, '%v' must be appended to\n"+
            "the topic name.", FIFO_SUFFIX)
    }

    topicArn, err := runner.snsActor.CreateTopic(ctx, topicName, isFifoTopic,
contentBasedDeduplication)
    if err != nil {
        panic(err)
    }
    log.Printf("Your new topic with the name '%v' and Amazon Resource Name (ARN) \n"+

```

```

    "'%v' has been created.", topicName, topicArn)

    return topicName, topicArn, isFifoTopic, contentBasedDeduplication
}

func (runner ScenarioRunner) CreateQueue(ctx context.Context, ordinal string,
    isFifoTopic bool) (string, string) {
    queueName := runner.questioner.Ask(fmt.Sprintf("Enter a name for the %v SQS queue.",
        ordinal))
    if isFifoTopic {
        queueName = fmt.Sprintf("%v%v", queueName, FIFO_SUFFIX)
        if ordinal == "first" {
            log.Printf("Because you are creating a FIFO SQS queue, '%v' must "+
                "be appended to the queue name.\n", FIFO_SUFFIX)
        }
    }
    queueUrl, err := runner.sqsActor.CreateQueue(ctx, queueName, isFifoTopic)
    if err != nil {
        panic(err)
    }
    log.Printf("Your new SQS queue with the name '%v' and the queue URL "+
        "'%v' has been created.", queueName, queueUrl)

    return queueName, queueUrl
}

func (runner ScenarioRunner) SubscribeQueueToTopic(
    ctx context.Context, queueName string, queueUrl string, topicName string, topicArn
    string, ordinal string,
    isFifoTopic bool) (string, bool) {

    queueArn, err := runner.sqsActor.GetQueueArn(ctx, queueUrl)
    if err != nil {
        panic(err)
    }
    log.Printf("The ARN of your queue is: %v.\n", queueArn)

    err = runner.sqsActor.AttachSendMessagePolicy(ctx, queueUrl, queueArn, topicArn)
    if err != nil {
        panic(err)
    }
    log.Println("Attached an IAM policy to the queue so the SNS topic can send " +
        "messages to it.")
    log.Println(strings.Repeat("-", 88))
}

```

```

var filterPolicy map[string][]string
if isFifoTopic {
    if ordinal == "first" {
        log.Println("Subscriptions to a FIFO topic can have filters.\n" +
            "If you add a filter to this subscription, then only the filtered messages\n" +
            "will be received in the queue.\n" +
            "For information about message filtering, see\n" +
            "\thttps://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n" +
            "For this example, you can filter messages by a \"tone\" attribute.")
    }

    wantFiltering := runner.questioner.AskBool(
        fmt.Sprintf("Do you want to filter messages that are sent to \"%v\"\n"+
            "from the %v topic? (y/n) ", queueName, topicName), "y")
    if wantFiltering {
        log.Println("You can filter messages by one or more of the following \"tone\" attributes.")

        var toneSelections []string
        askAboutTones := true
        for askAboutTones {
            toneIndex := runner.questioner.AskChoice(
                "Enter the number of the tone you want to filter by:\n", ToneChoices)
            toneSelections = append(toneSelections, ToneChoices[toneIndex])
            askAboutTones = runner.questioner.AskBool("Do you want to add another tone to the filter? (y/n) ", "y")
        }
        log.Printf("Your subscription will be filtered to only pass the following tones: %v\n", toneSelections)
        filterPolicy = map[string][]string{TONE_KEY: toneSelections}
    }
}

subscriptionArn, err := runner.snsActor.SubscribeQueue(ctx, topicArn, queueArn, filterPolicy)
if err != nil {
    panic(err)
}
log.Printf("The queue %v is now subscribed to the topic %v with the subscription ARN %v.\n",
    queueName, topicName, subscriptionArn)

return subscriptionArn, filterPolicy != nil

```



```

}

func (runner ScenarioRunner) PublishMessages(ctx context.Context, topicArn string,
isFifoTopic bool, contentBasedDeduplication bool, usingFilters bool) {
    var message string
    var groupId string
    var dedupId string
    var toneSelection string
    publishMore := true
    for publishMore {
        groupId = ""
        dedupId = ""
        toneSelection = ""
        message = runner.questioner.Ask("Enter a message to publish: ")
        if isFifoTopic {
            log.Println("Because you are using a FIFO topic, you must set a message group ID.
\n" +
                "All messages within the same group will be received in the order they were
published.")
            groupId = runner.questioner.Ask("Enter a message group ID: ")
            if !contentBasedDeduplication {
                log.Println("Because you are not using content-based deduplication,\n" +
                    "you must enter a deduplication ID.")
                dedupId = runner.questioner.Ask("Enter a deduplication ID: ")
            }
        }
        if usingFilters {
            if runner.questioner.AskBool("Add a tone attribute so this message can be
filtered? (y/n) ", "y") {
                toneIndex := runner.questioner.AskChoice(
                    "Enter the number of the tone you want to filter by:\n", ToneChoices)
                toneSelection = ToneChoices[toneIndex]
            }
        }

        err := runner.snsActor.Publish(ctx, topicArn, message, groupId, dedupId, TONE_KEY,
toneSelection)
        if err != nil {
            panic(err)
        }
        log.Println(("Your message was published.))

        publishMore = runner.questioner.AskBool("Do you want to publish another message?
(y/n) ", "y")
    }
}

```

```

    }
}

func (runner ScenarioRunner) PollForMessages(ctx context.Context, queueUrls
[]string) {
    log.Println("Polling queues for messages...")
    for _, queueUrl := range queueUrls {
        var messages []types.Message
        for {
            currentMsgs, err := runner.sqsActor.GetMessages(ctx, queueUrl, 10, 1)
            if err != nil {
                panic(err)
            }
            if len(currentMsgs) == 0 {
                break
            }
            messages = append(messages, currentMsgs...)
        }
        if len(messages) == 0 {
            log.Printf("No messages were received by queue %v.\n", queueUrl)
        } else if len(messages) == 1 {
            log.Printf("One message was received by queue %v:\n", queueUrl)

        } else {
            log.Printf("%v messages were received by queue %v:\n", len(messages), queueUrl)
        }
        for msgIndex, message := range messages {
            messageBody := MessageBody{}
            err := json.Unmarshal([]byte(*message.Body), &messageBody)
            if err != nil {
                panic(err)
            }
            log.Printf("Message %v: %v\n", msgIndex+1, messageBody.Message)
        }

        if len(messages) > 0 {
            log.Printf("Deleting %v messages from queue %v.\n", len(messages), queueUrl)
            err := runner.sqsActor.DeleteMessages(ctx, queueUrl, messages)
            if err != nil {
                panic(err)
            }
        }
    }
}

```

```
// RunTopicsAndQueuesScenario is an interactive example that shows you how to use
// the
// AWS SDK for Go to create and use Amazon SNS topics and Amazon SQS queues.
//
// 1. Create a topic (FIFO or non-FIFO).
// 2. Subscribe several queues to the topic with an option to apply a filter.
// 3. Publish messages to the topic.
// 4. Poll the queues for messages received.
// 5. Delete the topic and the queues.
//
// This example creates service clients from the specified sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
// example.
// This package can be found in the ../../demotools folder of this repo.
func RunTopicsAndQueuesScenario(
    ctx context.Context, sdkConfig aws.Config, questioner demotools.IQuestioner) {
    resources := Resources{}
    defer func() {
        if r := recover(); r != nil {
            log.Println("Something went wrong with the demo.\n" +
                "Cleaning up any resources that were created...")
            resources.Cleanup(ctx)
        }
    }()
    queueCount := 2

    log.Println(strings.Repeat("-", 88))
    log.Printf("Welcome to messaging with topics and queues.\n\n"+
        "In this scenario, you will create an SNS topic and subscribe %v SQS queues to the\n"+
        "topic. You can select from several options for configuring the topic and the\n"+
        "subscriptions for the queues. You can then post to the topic and see the results\n"+
        "in the queues.\n", queueCount)

    log.Println(strings.Repeat("-", 88))

    runner := ScenarioRunner{
        questioner: questioner,
        snsActor:   &actions.SnsActions{SnsClient: sns.NewFromConfig(sdkConfig)},
        sqsActor:   &actions.SqsActions{SqsClient: sqs.NewFromConfig(sdkConfig)},
    }
```

```

}
resources.snsActor = runner.snsActor
resources.sqsActor = runner.sqsActor

topicName, topicArn, isFifoTopic, contentBasedDeduplication :=
runner.CreateTopic(ctx)
resources.topicArn = topicArn
log.Println(strings.Repeat("-", 88))

log.Printf("Now you will create %v SQS queues and subscribe them to the topic.\n",
queueCount)
ordinals := []string{"first", "next"}
usingFilters := false
for _, ordinal := range ordinals {
    queueName, queueUrl := runner.CreateQueue(ctx, ordinal, isFifoTopic)
    resources.queueUrls = append(resources.queueUrls, queueUrl)

    _, filtering := runner.SubscribeQueueToTopic(ctx, queueName, queueUrl, topicName,
topicArn, ordinal, isFifoTopic)
    usingFilters = usingFilters || filtering
}

log.Println(strings.Repeat("-", 88))
runner.PublishMessages(ctx, topicArn, isFifoTopic, contentBasedDeduplication,
usingFilters)
log.Println(strings.Repeat("-", 88))
runner.PollForMessages(ctx, resources.queueUrls)

log.Println(strings.Repeat("-", 88))

wantCleanup := questioner.AskBool("Do you want to remove all AWS resources created
for this scenario? (y/n) ", "y")
if wantCleanup {
    log.Println("Cleaning up resources...")
    resources.Cleanup(ctx)
}

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

```

이 예시에서 사용되는 Amazon SNS 작업을 래핑하는 구조를 정의합니다.

```
import (
    "context"
    "encoding/json"
    "log"

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

// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
// actions
// used in the examples.
type SnsActions struct {
    SnsClient *sns.Client
}

// CreateTopic creates an Amazon SNS topic with the specified name. You can
// optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
// based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(ctx context.Context, topicName string,
    isFifoTopic bool, contentBasedDeduplication bool) (string, error) {
    var topicArn string
    topicAttributes := map[string]string{}
    if isFifoTopic {
        topicAttributes["FifoTopic"] = "true"
    }
    if contentBasedDeduplication {
        topicAttributes["ContentBasedDeduplication"] = "true"
    }
    topic, err := actor.SnsClient.CreateTopic(ctx, &sns.CreateTopicInput{
        Name:      aws.String(topicName),
        Attributes: topicAttributes,
    })
    if err != nil {
        log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
    } else {
```

```

    topicArn = *topic.TopicArn
}

return topicArn, err
}

// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(ctx context.Context, topicArn string) error {
    _, err := actor.SnsClient.DeleteTopic(ctx, &sns.DeleteTopicInput{
        TopicArn: aws.String(topicArn)})
    if err != nil {
        log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
    }
    return err
}

// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
// policy
// so that messages are only sent to the queue when the message has the specified
// attributes.
func (actor SnsActions) SubscribeQueue(ctx context.Context, topicArn string,
    queueArn string, filterMap map[string][]string) (string, error) {
    var subscriptionArn string
    var attributes map[string]string
    if filterMap != nil {
        filterBytes, err := json.Marshal(filterMap)
        if err != nil {
            log.Printf("Couldn't create filter policy, here's why: %v\n", err)
            return "", err
        }
        attributes = map[string]string{"FilterPolicy": string(filterBytes)}
    }
    output, err := actor.SnsClient.Subscribe(ctx, &sns.SubscribeInput{
        Protocol:      aws.String("sqs"),
        TopicArn:      aws.String(topicArn),
        Attributes:    attributes,
        Endpoint:      aws.String(queueArn),
        ReturnSubscriptionArn: true,
    })
}

```

```

    if err != nil {
        log.Printf("Couldn't subscribe queue %v to topic %v. Here's why: %v\n",
            queueArn, topicArn, err)
    } else {
        subscriptionArn = *output.SubscriptionArn
    }

    return subscriptionArn, err
}

// Publish publishes a message to an Amazon SNS topic. The message is then sent to
// all
// subscribers. When the topic is a FIFO topic, the message must also contain a
// group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
// value
// filter attribute can be specified so that the message can be filtered according
// to
// a filter policy.
func (actor SnsActions) Publish(ctx context.Context, topicArn string, message
    string, groupId string, dedupId string, filterKey string, filterValue string) error
{
    publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
    aws.String(message)}
    if groupId != "" {
        publishInput.MessageGroupId = aws.String(groupId)
    }
    if dedupId != "" {
        publishInput.MessageDeduplicationId = aws.String(dedupId)
    }
    if filterKey != "" && filterValue != "" {
        publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
            filterKey: {DataType: aws.String("String"), StringValue:
            aws.String(filterValue)},
        }
    }
    _, err := actor.SnsClient.Publish(ctx, &publishInput)
    if err != nil {
        log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn, err)
    }
    return err
}

```

이 예시에서 사용되는 Amazon SQS 작업을 래핑하는 구조를 정의합니다.

```
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
}

```

리소스를 정리합니다.

```

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)
  }
}
}
```

• API 세부 정보는 AWS SDK for Go API 참조의 다음 주제를 참조하세요.

- [CreateQueue](#)
- [CreateTopic](#)
- [DeleteMessageBatch](#)
- [DeleteQueue](#)
- [DeleteTopic](#)
- [GetQueueAttributes](#)
- [Publish](#)
- [ReceiveMessage](#)
- [SetQueueAttributes](#)
- [Subscribe](#)
- [Unsubscribe](#)

서버리스 예제

Amazon SQS 트리거에서 간접적으로 Lambda 함수 간접 호출

다음 코드 예제는 SQS 대기열에서 메시지를 받아 트리거된 이벤트를 수신하는 Lambda 함수를 구현하는 방법을 보여줍니다. 함수는 이벤트 파라미터에서 메시지를 검색하고 각 메시지의 내용을 로깅합니다.

SDK for Go V2

 Note

GitHub에 더 많은 내용이 있습니다. [서버리스 예제](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Go를 사용하여 Lambda로 SQS 이벤트를 사용합니다.

```
// 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)
}
```

Amazon SQS 트리거로 Lambda 함수에 대한 배치 항목 실패 보고

다음 코드 예제는 SQS 대기열에서 이벤트를 수신하는 Lambda 함수에 대한 부분 배치 응답을 구현하는 방법을 보여줍니다. 이 함수는 응답으로 배치 항목 실패를 보고하고 나중에 해당 메시지를 다시 시도하도록 Lambda에 신호를 보냅니다.

SDK for Go V2

Note

GitHub에 더 많은 내용이 있습니다. [서버리스 예제](#) 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Go를 사용하여 Lambda로 SQS 배치 항목 실패 보고

```
// 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 SDK for Go v2로 마이그레이션

Go 최소 버전

에는 최소 버전의 Go 1.23이 AWS SDK for Go 필요합니다. 최신 버전의 Go는 [다운로드](#) 페이지에서 다운로드할 수 있습니다. 각 Go 버전 릴리스에 대한 자세한 내용과 업그레이드에 필요한 관련 정보는 [릴리스 기록](#)을 참조하세요.

모듈화

Go 1.13에서 기본 개발 모드가 된 Go 모듈을 활용하도록 업데이트 AWS SDK for Go 되었습니다. SDK에서 제공하는 여러 패키지는 모듈화되었으며 각각 독립적으로 버전이 지정되고 릴리스됩니다. 이 변경을 통해 애플리케이션 종속성 모델링을 개선하고, SDK가 Go 모듈 버전 관리 전략을 따르는 새로운 기능을 제공할 수 있습니다.

다음은 SDK에서 제공하는 일부 Go 모듈 목록입니다.

모듈	설명
github.com/aws/aws-sdk-go-v2	SDK 코어
github.com/aws/aws-sdk-go-v2/config	공유 구성 로드
github.com/aws/aws-sdk-go-v2/credentials	AWS 자격 증명 공급자
github.com/aws/aws-sdk-go-v2/feature/ec2/imds	Amazon EC2 인스턴스 메타데이터 서비스 클라이언트

SDK의 서비스 클라이언트 및 상위 수준 유틸리티 모듈은 다음 가져오기 경로 아래에 중첩됩니다.

루트 가져오기	설명
github.com/aws/aws-sdk-go-v2/service/	서비스 클라이언트 모듈

루트 가져오기	설명
<code>github.com/aws/aws-sdk-go-v2/feature/</code>	Amazon S3 Transfer Manager와 같은 서비스를 위한 상위 수준 유틸리티

구성 로드

[세션](#) 패키지 및 관련 기능은 [구성](#) 패키지에서 제공하는 간소화된 구성 시스템으로 대체됩니다. `config` 패키지는 별도의 Go 모듈이며 `go get`을 사용하여 애플리케이션의 종속성에 포함될 수 있습니다.

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

[session.New](#), [session.NewSession](#), [NewSessionWithOptions](#) 및 [session.Must](#)는 [config.LoadDefaultConfig](#)로 마이그레이션되어야 합니다.

`config` 패키지에서 제공하는 여러 헬퍼 함수는 프로그래밍 방식으로 공유 구성 로드를 재정의하는 데 도움이 됩니다. 이러한 함수 이름에는 접두사 `With` 뒤에 재정의하는 옵션이 따릅니다. `session` 패키지 사용을 마이그레이션하는 방법의 몇 가지 예를 살펴보겠습니다.

공유 구성 로드와 관련한 자세한 내용은 [SDK 구성](#) 섹션을 참조하세요.

예제

NewSession에서 LoadDefaultConfig로 마이그레이션

다음 예제에서는 추가 인수 파라미터가 없는 `session.NewSession`의 사용이 `config.LoadDefaultConfig`로 마이그레이션되는 방법을 보여줍니다.

```
// V1 using NewSession

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

aws.Config 옵션을 사용하여 NewSession에서 마이그레이션

이 예제에서는 구성 로드 중에 `aws.Config` 값의 재정의의 마이그레이션하는 방법을 보여줍니다. 로드된 구성 값을 재정의하기 위해 하나 이상의 `config.With*` 헬퍼 함수를 `config.LoadDefaultConfig`에 제공할 수 있습니다. 이 예제에서는 AWS 리전이 [config.WithRegion](#) 헬퍼 함수를 `us-west-2` 사용하여 로 재정의됩니다.

```
// 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
}
```

NewSessionWithOptions에서 마이그레이션

이 예제에서는 구성 로드 중에 값의 재정의의 마이그레이션하는 방법을 보여줍니다. 로드된 구성 값을 재정의하기 위해 0개 이상의 `config.With*` 헬퍼 함수를 `config.LoadDefaultConfig`에 제공할 수 있습니다. 이 예제에서는 AWS SDK 공유 구성을 로드할 때 사용되는 대상 프로파일을 재정의하는 방법을 보여줍니다.

```
// 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
}
```

모의 테스트 및 *iface

*iface 패키지 및 인터페이스(예: [s3iface.S3API](#))가 제거되었습니다. 이러한 인터페이스 정의는 서비스가 새 작업을 추가할 때마다 중단되므로 안정적이지 않습니다.

*iface의 사용은 사용 중인 서비스 작업에 대해 범위가 지정된 호출자 정의 인터페이스로 대체되어야 합니다.

```
// 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)
}

```

자세한 내용은 [AWS SDK for Go v2를 사용한 유닛 테스트](#) 섹션을 참조하세요.

자격 증명 및 자격 증명 공급자

[aws/credentials](#) 패키지 및 관련 자격 증명 공급자가 [자격 증명](#) 패키지 위치로 재배치되었습니다. `credentials` 패키지는 `go get`을 사용하여 검색하는 Go 모듈입니다.

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

AWS SDK for Go v2 릴리스는 AWS 자격 증명 공급자를 업데이트하여 자격 증명을 검색하기 AWS 위한 일관된 인터페이스를 제공합니다. 각 공급자는 AWS SDK for Go v1 [credentials.Value](#) 유형과 유사한 `aws.Credentials`를 반환하는 메서드를 정의하는 `aws.CredentialsProvider` 인터페이스를 구현합니다. `Retrieve (aws.Credentials, error)` <https://pkg.go.dev/github.com/aws/aws-sdk-go-v2/aws#Credentials>

자격 증명 캐싱이 발생할 수 있도록 `aws.CredentialsProvider` 객체를 [aws.CredentialsCache](#)로 래핑해야 합니다. [NewCredentialsCache](#)를 사용하여 `aws.CredentialsCache` 객체를 구성합니다. 기본적으로 `config.LoadDefaultConfig`에서 구성된 자격 증명은 `aws.CredentialsCache`로 래핑됩니다.

다음 표에는 AWS SDK for Go 자격 증명 공급자의 v1에서 v2로의 위치 변경 사항이 AWS 나와 있습니다.

이름	V1 가져오기	V2 가져오기
Amazon EC2 IAM 역할 자격 증명	<code>github.com/aws/aws-sdk-go/aws/credentials/ec2rolecreds</code>	<code>github.com/aws/aws-sdk-go-v2/credentials/ec2rolecreds</code>
엔드포인트 자격 증명	<code>github.com/aws/aws-sdk-go/aws/credentials/endpointcreds</code>	<code>github.com/aws/aws-sdk-go-v2/credentials/endpointcreds</code>
프로세스 자격 증명	<code>github.com/aws/aws-sdk-go/aws/credentials/processcreds</code>	<code>github.com/aws/aws-sdk-go-v2/credentials/processcreds</code>
AWS Security Token Service	<code>github.com/aws/aws-sdk-go/aws/credentials/stscreds</code>	<code>github.com/aws/aws-sdk-go-v2/credentials/stscreds</code>

정적 자격 증명

[credentials.NewStaticCredentials](#)를 사용하여 프로그래밍 방식으로 정적 자격 증명을 구성하는 애플리케이션은 [credentials.NewStaticCredentialsProvider](#)를 사용해야 합니다.

예제

```
// 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
}
```

Amazon EC2 IAM 역할 자격 증명

[New](#)를 사용하려면 [NewCredentials](#) 및 [NewCredentialsWithClient](#) 사용을 마이그레이션해야 합니다.

ec2rolecreds 패키지의 ec2rolecreds.New는 [ec2rolecreds.Options](#)의 기능 옵션을 입력으로 사용하므로 사용할 특정 Amazon EC2 인스턴스 메타데이터 서비스 클라이언트를 재정의하거나 자격 증명 만료 기간을 재정의할 수 있습니다.

예제

```
// 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
}
```

엔드포인트 자격 증명

[New](#)를 사용하려면 [NewCredentialsClient](#) 및 [NewProviderClient](#) 사용을 마이그레이션해야 합니다.

endpointcreds 패키지의 New 함수는 HTTP 또는 HTTPS 엔드포인트의 URL을 포함하는 문자열 인수를 사용하여 자격 증명을 검색하고 [endpointcreds.Options](#)의 기능 옵션을 사용하여 자격 증명 공급자를 변경하고 특정 구성 설정을 재정의합니다.

프로세스 자격 증명

[NewProvider](#) 또는 [NewProviderCommand](#)를 사용하려면 [NewCredentials](#), [NewCredentialsCommand](#) 및 [NewCredentialsTimeout](#) 사용을 마이그레이션해야 합니다.

processcreds 패키지의 NewProvider 함수는 호스트 환경의 셸에서 실행할 명령인 문자열 인수와 자격 증명 공급자를 변경하고 특정 구성 설정을 재정의하는 [옵션](#)의 기능 옵션을 사용합니다.

NewProviderCommand는 하나 이상의 명령줄 인수를 사용하거나 특정 실행 환경 요구 사항이 있을 수 있는 보다 복잡한 프로세스 명령을 정의하는 [NewCommandBuilder](#) 인터페이스를 구현합니다. [DefaultNewCommandBuilder](#)는 이 인터페이스를 구현하고 여러 명령줄 인수가 필요한 프로세스에 대한 명령 빌더를 정의합니다.

예제

```
// 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 자격 증명

AssumeRole

[NewAssumeRoleProvider](#)를 사용하려면 [NewCredentials](#) 및 [NewCredentialsWithClient](#) 사용을 마이그레이션해야 합니다.

stscreds 패키지의 NewAssumeRoleProvider 함수는 [sts.Client](#)로 호출해야 하며 역할 AWS Identity and Access Management ARN은 제공된 sts.Client의 구성된 자격 증명에서 수입해야 합니다. [AssumeRoleOptions](#)의 기능 옵션 세트를 제공하여 공급자의 다른 선택적 설정을 수정할 수도 있습니다.

예제

```
// V1

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

[NewWebIdentityRoleProvider](#)를 사용하려면 [NewWebIdentityCredentials](#), [NewWebIdentityRoleProvider](#), [NewWebIdentityRoleProviderWithToken](#) 사용을 마이그레이션해야 합니다.

stscreds 패키지의 NewWebIdentityRoleProvider 함수는 [sts.Client](#)로 호출해야 하며, AWS Identity and Access Management 역할 ARN은 제공된 sts.Client 구성된 자격 증명과 OAuth 2.0 또는 OpenID Connect ID 토큰을 제공하기 위한 [IdentityTokenRetriever](#) 구현을 사용하여 수입해야 합니다. [IdentityTokenFile](#)은 애플리케이션의 호스트 파일 시스템에 있는 파일에서 웹 자격 증명 토큰을 제공하는 데 사용할 수 있는 IdentityTokenRetriever입니다. [WebIdentityRoleOptions](#)의 기능 옵션 세트를 제공하여 공급자의 다른 선택적 설정을 수정할 수도 있습니다.

예제

```
// V1

import "github.com/aws/aws-sdk-go-v2/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
}

```

서비스 클라이언트

AWS SDK for Go 는 `github.com/aws/aws-sdk-go-v2/service` 가져오기 경로 아래에 중첩된 서비스 클라이언트 모듈을 제공합니다. 각 서비스 클라이언트는 각 서비스의 고유 식별자를 사용하여 Go 패키지에 포함됩니다. 다음 표에는 AWS SDK for Go의 서비스 가져오기 경로의 몇 가지 예가 나와 있습니다.

서비스 이름	V1 가져오기 경로	V2 가져오기 경로
Amazon S3	<code>github.com/aws/aws-sdk-go/service/s3</code>	<code>github.com/aws/aws-sdk-go-v2/service/s3</code>
Amazon DynamoDB	<code>github.com/aws/aws-sdk-go/service/dynamodb</code>	<code>github.com/aws/aws-sdk-go-v2/service/dynamodb</code>

서비스 이름	V1 가져오기 경로	V2 가져오기 경로
Amazon CloudWatch Logs	github.com/aws/aws-sdk-go/service/cloudwatchlogs	github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs

각 서비스 클라이언트 패키지는 독립적으로 버전이 지정된 Go 모듈입니다. 애플리케이션의 종속성으로 서비스 클라이언트를 추가하려면 서비스의 가져오기 경로와 함께 `go get` 명령을 사용합니다. 예를 들어 종속성에 Amazon S3 클라이언트를 추가하려면 다음을 사용합니다.

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

클라이언트 구성

클라이언트 패키지의 `New` 또는 생성자 함수를 AWS SDK for Go 사용하여 클라이언트를 `NewFromConfig` 구성할 수 있습니다. AWS SDK for Go v1에서 마이그레이션할 때는 `NewFromConfig` 변형을 사용하는 것이 좋습니다. 그러면의 값을 사용하여 새 서비스 클라이언트가 반환됩니다 `aws.Config`. `aws.Config` 값은 `config.LoadDefaultConfig`를 사용하여 SDK 공유 구성을 로드하는 동안 생성됩니다. 서비스 클라이언트 생성에 대한 자세한 내용은 [AWS SDK for Go v2를 AWS 서비스와 함께 사용](#) 섹션을 참조하세요.

예제 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)
```

예 2: 클라이언트 설정 재정의

```
// 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"
})

```

엔드포인트

[엔드포인트](#) 패키지는 더 이상 AWS SDK for Go에 존재하지 않습니다. 이제 각 서비스 클라이언트는 클라이언트 패키지 내에 필요한 AWS 엔드포인트 메타데이터를 포함합니다. 이렇게 하면 애플리케이션에서 사용하지 않는 서비스에 대한 엔드포인트 메타데이터를 더 이상 포함하지 않아 컴파일된 애플리케이션의 전체 바이너리 크기가 줄어듭니다.

또한 각 서비스는 이제 `EndpointResolverV2`에서 엔드포인트 확인을 위한 자체 인터페이스를 노출합니다. 각 API는 서비스 `EndpointParameters`에 대해 고유한 파라미터 세트를 가져옵니다. 이 파라미터의 값은 작업이 간접적으로 호출될 때 다양한 위치에서 SDK가 소싱합니다.

기본적으로 서비스 클라이언트는 구성된 AWS 리전을 사용하여 대상 리전의 서비스 엔드포인트를 확인합니다. 애플리케이션에 사용자 지정 엔드포인트가 필요한 경우 `aws.Config` 구조의 `EndpointResolverV2` 필드에 사용자 지정 동작을 지정할 수 있습니다. 애플리케이션이 사용자 지정 [endpoints.Resolver](#)를 구현하는 경우 이 새로운 서비스별 인터페이스를 준수하도록 마이그레이션해야 합니다.

엔드포인트 및 사용자 지정 해석기 구현에 대한 자세한 정보는 [클라이언트 엔드포인트 구성](#) 섹션을 참조하세요.

Authentication

는 코드 카탈로그 및 S3 Express One Zone과 같은 최신 AWS 서비스 기능을 사용할 수 있는 고급 인증 동작을 AWS SDK for Go 지원합니다. 또한 이 동작은 클라이언트별로 사용자 지정할 수 있습니다.

API 작업 간접 호출

서비스 클라이언트 작업 메서드 수가 크게 줄었습니다. `<OperationName>Request`, `<OperationName>WithContext` 및 `<OperationName>` 메서드는 모두 단일 작업 메서드인 `<OperationName>`으로 통합되었습니다.

예제

다음 예제에서는 Amazon S3 `PutObject` 작업에 대한 호출을 AWS SDK for Go v1에서 v2로 마이그레이션하는 방법을 보여줍니다.

```
// 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
})
```

서비스 데이터 형식

작업의 최상위 입력 및 출력 유형은 서비스 클라이언트 패키지에서 찾을 수 있습니다. 주어진 작업의 입력 및 출력 유형은 <OperationName>Input 및 <OperationName>Output 패턴을 따릅니다.

다. 여기서 `OperationName`은 간접적으로 호출하려는 작업의 이름입니다. 예를 들어 Amazon S3 `PutObject` 작업의 입력 및 출력 셰이프는 각각 [PutObjectInput](#) 및 [PutObjectOutput](#)입니다.

입력 및 출력 유형을 제외한 다른 모든 서비스 데이터 유형은 서비스 클라이언트 패키지 가져오기 경로 계층 구조 아래에 있는 `types` 패키지로 마이그레이션되었습니다. 예를 들어 [s3.AccessControlPolicy](#) 유형은 이제 [types.AccessControlPolicy](#)에 있습니다.

열거 값

SDK는 이제 모든 API 열거 필드에 대해 유형화된 환경을 제공합니다. 이제 서비스 API 참조 설명서에서 복사한 문자열 리터럴 값을 사용하는 대신 서비스 클라이언트의 `types` 패키지에 있는 구체적인 유형 중 하나를 사용할 수 있습니다. 예를 들어 Amazon S3 `PutObjectInput` 작업에 객체에 적용할 ACL을 제공할 수 있습니다. AWS SDK for Go v1에서 이 파라미터는 `*string` 유형이었습니다. AWS SDK for Go에서 이 파라미터는 이제 [type.ObjectCannedACL](#)입니다. `types` 패키지는 이 필드에 할당할 수 있는 유효한 열거 값에 대해 생성된 상수를 제공합니다. 예를 들어 [type.ObjectCannedACLPrivate](#)은 '프라이빗' 미리 준비된 ACL 값의 상수입니다. 이 값은 애플리케이션 내에서 문자열 상수를 관리하는 대신 사용할 수 있습니다.

포인터 파라미터

AWS SDK for Go v1 필수 포인터 참조는 모든 입력 파라미터에 대해 서비스 작업에 전달됩니다. AWS SDK for Go v2는 가능한 경우 입력 값을 포인터로 전달할 필요가 없으므로 대부분의 서비스에 대한 경험을 간소화했습니다. 이 변경으로 인해 많은 서비스 클라이언트 작업에서 애플리케이션이 더 이상 `uint8`, `uint16`, `uint32`, `int8`, `int16`, `int32`, `float32`, `float64`, `bool` 유형에 대한 포인터 참조를 전달할 필요가 없습니다. 마찬가지로 슬라이스 및 맵 요소 유형이 그에 따라 업데이트되어 해당 요소를 포인터 참조로 전달해야 하는지 여부를 반영합니다.

[aws](#) 패키지에는 Go 기본 제공 유형에 대한 포인터를 생성하기 위한 헬퍼 함수가 포함되어 있습니다. 이러한 헬퍼를 사용하여 Go 유형에 대한 포인터 유형 생성을 보다 쉽게 처리해야 합니다. 마찬가지로 이러한 유형의 포인터 값을 안전하게 참조 해제하기 위한 도우미 메서드가 제공됩니다. 예를 들어 [aws.String](#) 함수는 `string`에서 `*string`으로 변환됩니다. 반대로 [aws.ToString](#)은 `*string`에서 `string`으로 변환됩니다. 애플리케이션을 AWS SDK for Go v1에서 v2로 업그레이드할 때 포인터 유형에서 비포인터 변형으로 변환하기 위한 헬퍼 사용을 마이그레이션해야 합니다. 예를 들어 [aws.StringValue](#)를 `aws.ToString`으로 업데이트해야 합니다.

오류 유형

AWS SDK for Go는 [Go 1.13에 도입](#)된 오류 래핑 기능을 최대한 활용합니다. 오류 응답을 모델링하는 서비스는 클라이언트의 `types` 패키지에서 이러한 유형 중 하나로 인해 클라이언트 작업 오류

가 발생했는지 여부를 테스트하는 데 사용할 수 있는 유형을 생성했습니다. 예를 들어 Amazon S3 `GetObject` 작업은 존재하지 않는 객체 키를 검색하려고 하면 `NoSuchKey` 오류를 반환할 수 있습니다. [errors.As](#)를 사용하여 반환된 작업 오류가 [types.NoSuchKey](#) 오류인지 테스트할 수 있습니다. 서비스가 특정 유형의 오류를 모델링하지 않는 경우 [smithy.APIError](#) 인터페이스 유형을 활용하여 서비스의 반환된 오류 코드와 메시지를 검사할 수 있습니다. 이 기능은 v AWS SDK for Go 1의 [awserr.Error](#) 및 기타 [awserr](#) 기능을 대체합니다. 오류 처리에 대한 자세한 내용은 [AWS SDK for Go V2에서 오류 처리](#) 섹션을 참조하세요.

예제

```
// 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 ns *types.NoSuchKey
    if errors.As(err, &ns) {
        // 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
}

```

페이지네이터

서비스 작업 페이지네이터는 더 이상 서비스 클라이언트에서 메서드로 간접 호출되지 않습니다. 작업에 페이지네이터를 사용하려면 페이지네이터 생성자 방법 중 하나를 사용하여 작업에 페이지네이터를 구성해야 합니다. 예를 들어 Amazon S3 ListObjectsV2 작업에 페이지 매기기를 사용하려면 [s3.NewListObjectsV2Paginator](#)를 구성해야 합니다. 이 생성자는 메서드 HasMorePages 및 NextPage를 제공하는 [ListObjectsV2Paginator](#)를 반환하여 검색할 페이지가 더 있는지 확인하고 작업을 간접적으로 호출하여 각각 다음 페이지를 검색합니다. SDK 페이지네이터 사용에 대한 자세한 내용은 [작업 페이지네이터 사용](#)에서 확인할 수 있습니다.

AWS SDK for Go v1 페이지네이터에서 AWS SDK for Go v2에 상응하는 페이지로 마이그레이션하는 방법의 예를 살펴보겠습니다.

예제

```

// V1

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

```

```
// ...

client := s3.New(sess)

params := &s3.ListObjectsV2Input{
    // input parameters
}

totalObjects := 0
err := client.ListObjectsV2Pages(params, func(output *s3.ListObjectsV2Output, lastPage
bool) bool {
    totalObjects += len(output.Contents)
    return !lastPage
})
if err != nil {
    // handle error
}
fmt.Println("total objects:", totalObjects)
```

```
// V2

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

// ...

client := s3.NewFromConfig(cfg)

params := &s3.ListObjectsV2Input{
    // input parameters
}

totalObjects := 0
paginator := s3.NewListObjectsV2Paginator(client, params)
for paginator.HasMorePages() {
    output, err := paginator.NextPage(context.TODO())
    if err != nil {
        // handle error
    }
    totalObjects += len(output.Contents)
}
```

```
fmt.Println("total objects:", totalObjects)
```

Waiters

서비스 작업 웨이터는 더 이상 서비스 클라이언트에서 메서드로 간접 호출되지 않습니다. 웨이터를 사용하려면 먼저 원하는 웨이터 유형을 구성한 다음 대기 방법을 간접 호출합니다. 예를 들어 Amazon S3 버킷이 존재할 때까지 기다리려면 `BucketExists` 웨이터를 구성해야 합니다. [s3.NewBucketExistsWaiter](#) 생성자를 사용하여 [s3.BucketExistsWaiter](#)를 생성합니다. `s3.BucketExistsWaiter`는 버킷을 사용할 수 있을 때까지 기다리는 데 사용할 수 있는 `Wait` 메서드를 제공합니다.

미리 서명된 요청

V1 SDK는 SDK AWS 작업의 사전 서명을 기술적으로 지원하지만 서비스 수준에서 실제로 지원되는 항목을 정확하게 나타내지는 않습니다(실제로 대부분의 AWS 서비스 작업은 사전 서명을 지원하지 않음).

AWS SDK for Go 는 지원되는 사전 서명 가능 작업에 대해 특정 APIs가 있는 서비스 패키지의 특정 `PresignClient` 구현을 노출하여이 문제를 해결합니다.

참고: SDK v1에서 성공적으로 사용하던 작업에 대한 사전 서명 지원이 서비스에서 누락된 경우 [GitHub에 문제를 제출](#)하여 알려주세요.

서비스별 사전 서명 클라이언트를 사용하도록 [Presign](#) 및 [PresignRequest](#)의 사용을 변환해야 합니다.

다음 예제에서는 S3 `GetObject` 요청의 사전 서명을 마이그레이션하는 방법을 보여줍니다.

```
// 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)
}

```

요청 사용자 지정

모놀리식 [request.Request](#) API가 다시 세분화되었습니다.

작업 입력/출력

이제 작업 입력 및 출력 구조를 각각 포함하는 불투명 Request 필드 Params 및 Data에 특정 미들웨어 단계 내에서 입력/출력으로 액세스할 수 있습니다.

Request.Params 및 Request.Data를 참조하는 요청 핸들러를 미들웨어로 마이그레이션해야 합니다.

Params 마이그레이션

```

// V1

import (
    "github.com/aws/aws-sdk-go/aws"
    "github.com/aws/aws-sdk-go/aws/request"
    "github.com/aws/aws-sdk-go/aws/session"
    "github.com/aws/aws-sdk-go/service/s3"
)

func withPutObjectDefaultACL(acl string) request.Option {
    return func(r *request.Request) {
        in, ok := r.Params.(*s3.PutObjectInput)
        if !ok {
            return
        }
    }
}

```

```

        if in.ACL == nil {
            in.ACL = aws.String(acl)
        }
        r.Params = in
    }
}

func main() {
    sess := session.Must(session.NewSession())

    sess.Handlers.Validate.PushBack(withPutObjectDefaultACL(s3.ObjectCannedACLBucketOwnerFullControl))

    // ...
}

```

```

// V2

import (
    "context"

    "github.com/aws/aws-sdk-go-v2/service/s3"
    "github.com/aws/aws-sdk-go-v2/service/s3/types"
    "github.com/aws/smithy-go/middleware"
    smithyhttp "github.com/aws/smithy-go/transport/http"
)

type withPutObjectDefaultACL struct {
    acl types.ObjectCannedACL
}

// implements middleware.InitializeMiddleware, which runs BEFORE a request has
// been serialized and can act on the operation input
var _ middleware.InitializeMiddleware = (*withPutObjectDefaultACL)(nil)

func (*withPutObjectDefaultACL) ID() string {
    return "withPutObjectDefaultACL"
}

func (m *withPutObjectDefaultACL) HandleInitialize(ctx context.Context, in
    middleware.InitializeInput, next middleware.InitializeHandler) (
    out middleware.InitializeOutput, metadata middleware.Metadata, err error,
) {
    input, ok := in.Parameters.(*s3.PutObjectInput)
}

```



```

    if !ok {
        return next.HandleInitialize(ctx, in)
    }

    if len(input.ACL) == 0 {
        input.ACL = m.acl
    }
    in.Parameters = input
    return next.HandleInitialize(ctx, in)
}

// create a helper function to simplify instrumentation of our middleware
func WithPutObjectDefaultACL(acl types.ObjectCannedACL) func (*s3.Options) {
    return func(o *s3.Options) {
        o.APIOptions = append(o.APIOptions, func (s *middleware.Stack) error {
            return s.Initialize.Add(&withPutObjectDefaultACL{acl: acl},
                middleware.After)
        })
    }
}

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

    svc := s3.NewFromConfig(cfg,
        WithPutObjectDefaultACL(types.ObjectCannedACLBucketOwnerFullControl))
    // ...
}

```

Data 마이그레이션

```

// 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)
    // ...
}

```

HTTP 요청/응답

이제 Request의 HTTPRequest 및 HTTPResponse 필드가 특정 미들웨어 단계에 노출됩니다. 미들웨어는 전송에 구매받지 않으므로 미들웨어 입력 또는 출력에서 유형 어설션을 수행하여 기본 HTTP 요청 또는 응답을 공개해야 합니다.

Request.HTTPRequest 및 Request.HTTPResponse를 참조하는 요청 핸들러를 미들웨어로 마이그레이션해야 합니다.

HTTPRequest 마이그레이션

```
// 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", "..."))
    // ...
}

```

핸들러 단계

SDK v2 미들웨어 단계는 v1 핸들러 단계의 후속 단계입니다.

다음 표는 V1 핸들러 단계를 V2 미들웨어 스택 내의 동등한 위치에 대략적으로 매핑한 것입니다.

v1 핸들러 이름	v2 미들웨어 단계
Validate	초기화
빌드	직렬화
Sign	마무리
Send	해당 사항 없음(1)
ValidateResponse	역직렬화
Unmarshal	역직렬화
UnmarshalMetadata	역직렬화
UnmarshalError	역직렬화
재시도	"Retry" 미들웨어(2) 이후 완료
AfterRetry	"Retry" 미들웨어 전, <code>next.HandleFinalize()</code> 후에 완료(2, 3)
CompleteAttempt	완료, 단계 종료
완료	초기화, 단계 시작, <code>next.HandleInitialize()</code> 후(3)

(1) v1의 Send 단계는 사실상 v2의 래핑된 HTTP 클라이언트 왕복입니다. 이 동작은 클라이언트 옵션의 `HTTPClient` 필드에 의해 제어됩니다.

(2) 완료 단계에서 미들웨어 이후의 모든 "Retry" 미들웨어는 재시도 루프의 일부가 됩니다.

(3) 작업 시 미들웨어 '스택'은 반복적으로 데코레이션된 핸들러 함수에 내장됩니다. 각 핸들러는 체인의 다음 핸들러를 직접 호출할 책임이 있습니다. 이는 암시적으로 다음 단계가 직접 호출된 후에도 미들웨어 단계가 조치를 취할 수 있음을 의미합니다.

예를 들어 스택 상단에 있는 초기화 단계의 경우 다음 핸들러를 직접 호출한 후 조치를 취하는 미들웨어를 초기화하면 요청이 끝날 때 효과적으로 작동합니다.

```
// 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
}
```

특성

Amazon EC2 인스턴스 메타데이터 서비스

는 Amazon EC2 인스턴스에서 애플리케이션을 실행할 때 로컬 IMDS를 쿼리하는 데 사용할 수 있는 Amazon EC2 인스턴스 메타데이터 서비스(IMDS) 클라이언트를 AWS SDK for Go 제공합니다. IMDS 클라이언트는 다음을 사용하여 애플리케이션에 추가할 수 있는 별도의 Go 모듈입니다.

```
go get github.com/aws/aws-sdk-go-v2/feature/ec2/imds
```

클라이언트 생성자 및 메서드 작업이 다른 SDK 서비스 클라이언트의 설계와 일치하도록 업데이트되었습니다.

예제

```
// V1

import "github.com/aws/aws-sdk-go/aws/ec2metadata"

// ...

client := ec2metadata.New(sess)

region, err := client.Region()
if err != nil {
    // handle error
}
```

```
// V2

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

// ...

client := imds.NewFromConfig(cfg)

region, err := client.GetRegion(context.TODO())
if err != nil {
    // handle error
}
```

Amazon S3 Transfer Manager

Amazon S3 전송 관리자는 객체의 업로드 및 다운로드를 동시에 관리하는 데 사용할 수 있습니다. 이 패키지는 서비스 클라이언트 가져오기 경로 외부의 Go 모듈에 있습니다. 이 모듈은 `go get github.com/aws/aws-sdk-go-v2/feature/s3/manager`를 사용하여 검색할 수 있습니다.

[s3.NewUploader](#) 및 [s3.NewUploaderWithClient](#)가 업로드 매니저 클라이언트를 생성하기 위한 생성자 메서드 [manager.NewUploader](#)로 대체되었습니다.

[s3.NewDownloader](#) 및 [s3.NewDownloaderWithClient](#)는 다운로드 매니저 클라이언트를 생성하기 위한 단일 생성자 메서드 [manager.NewDownloader](#)로 대체되었습니다.

Amazon CloudFront 서명 유틸리티

는 서비스 클라이언트 가져오기 경로 외부의 Go 모듈에 Amazon CloudFront 서명 유틸리티를 AWS SDK for Go 제공합니다. 이 모듈은 `go get`을 사용하여 검색할 수 있습니다.

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

Amazon S3 암호화 클라이언트

부터 AWS SDK for Go Amazon S3 암호화 클라이언트는 [AWS 암호화 도구](#) 아래에 있는 별도의 모듈입니다. Go용 S3 암호화 클라이언트의 최신 버전인 3.x는 이제 [GitHub](#)에서 사용할 수 있습니다. 이 모듈은 `go get`을 사용하여 검색할 수 있습니다.

```
go get github.com/aws/amazon-s3-encryption-client-go/v3
```

별도의 `EncryptionClient(v1, v2)` 및 `DecryptionClient(v1, v2)` API는 암호화 및 복호화 기능을 모두 노출하는 단일 클라이언트인 [S3EncryptionClientV3](#)로 대체되었습니다.

AWS SDK for Go의 다른 서비스 클라이언트와 마찬가지로 작업 APIs가 압축되었습니다.

- `GetObject`, `GetObjectRequest` 및 `GetObjectWithContext` 복호화 API는 [GetObject](#)로 대체되었습니다.
- `PutObject`, `PutObjectRequest` 및 `PutObjectWithContext` 암호화 API는 [PutObject](#)로 대체되었습니다.

암호화 클라이언트의 3.x 메이저 버전으로 마이그레이션하는 방법을 알아보려면 [이 가이드](#)를 참조하세요.

서비스 사용자 지정 변경 사항

Amazon S3

AWS SDK for Go v1에서 v2로 마이그레이션할 때 알아야 할 중요한 변경 사항에는 고객 제공 키(SSE-C)를 사용한 서버 측 암호화에 `SSECustomerKey` 사용되는의 처리가 포함됩니다. AWS SDK for Go v1에서는 `Base64SSECustomerKey`로의 인코딩이 SDK에 의해 내부적으로 처리되었습니다. SDK v2에서는 이 자동 인코딩이 제거되었으며, 이제 SDK에 전달하기 전에 `SSECustomerKey`를 Base64에 수동으로 인코딩해야 합니다.

조정 예:

```
// 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
```

AWS SDK for Go의 보안

AWS에서 클라우드 보안은 가장 중요합니다. AWS 고객은 보안에 가장 보안에 민감한 조직의 요구 사항에 부합하도록 빌드된 데이터 센터 및 네트워크 아키텍처의 혜택을 누릴 수 있습니다.

보안은 AWS와 사용자의 공동 책임입니다. [공동 책임 모델](#)은 이 사항을 클라우드의 보안 및 클라우드 내 보안으로 설명합니다.

- 클라우드의 보안 - AWS는 AWS 클라우드에서 AWS 서비스를 실행하는 인프라를 보호합니다. AWS는 또한 안전하게 사용할 수 있는 서비스를 제공합니다. 서드 파티 감사자는 [AWS 규정 준수 프로그램](#)의 일환으로 보안 효과를 정기적으로 테스트하고 검증합니다. AWS SDK for Go에 적용되는 규정 준수 프로그램에 대한 자세한 내용은 [규정 준수 프로그램의 범위에 속하는 AWS 서비스](#)를 참조하세요.
- 클라우드의 보안 - 귀하의 책임은 귀하가 사용하는 AWS 서비스에 의해 결정됩니다. 또한 귀하는 귀하의 데이터 민감도, 귀하의 요구 사항, 관련 법률 및 규정을 비롯한 기타 요소에 대해서도 책임이 있습니다.

이 설명서는 AWS SDK for Go 사용 시 공동 책임 모델을 적용하는 방법을 이해하는 데 도움이 됩니다. 다음 주제에서는 보안 및 규정 준수 목표를 충족하도록 AWS SDK for Go을(를) 구성하는 방법을 보여줍니다. 또한 AWS SDK for Go 리소스를 모니터링하고 보호하는 데 도움이 되는 다른 AWS 서비스 사용 방법을 알아봅니다.

주제

- [의 데이터 보호AWS SDK for Go](#)
- [의 규정 준수 검증AWS SDK for Go](#)
- [AWS SDK for Go의 복원성](#)

의 데이터 보호AWS SDK for Go

AWS [공동 책임 모델](#)은 AWS SDK for Go의 데이터 보호에 적용됩니다. 이 모델에서 설명하는 것처럼 AWS는 모든 AWS 클라우드를 실행하는 글로벌 인프라를 보호할 책임이 있습니다. 사용자는 인프라에서 호스팅되는 콘텐츠를 관리해야 합니다. 사용하는 AWS 서비스의 보안 구성과 관리 태스크에 대한 책임도 사용자에게 있습니다. 데이터 프라이버시에 대한 자세한 내용은 [데이터 프라이버시 FAQ](#)를 참조하세요. 유럽의 데이터 보호에 대한 자세한 내용은 AWS 보안 블로그의 [AWS 공동 책임 모델 및 GDPR](#) 블로그 게시물을 참조하세요.

데이터를 보호하려면 AWS 계정 자격 증명을 보호하고 AWS IAM Identity Center 또는 AWS Identity and Access Management(IAM)를 통해 개별 사용자 계정을 설정하는 것이 좋습니다. 이렇게 하면 개별 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정에 다중 인증(MFA)을 사용하세요.
- SSL/TLS를 사용하여 AWS 리소스와 통신하세요. TLS 1.2는 필수이며 TLS 1.3을 권장합니다.
- AWS CloudTrail로 API 및 사용자 활동 로깅을 설정하세요. AWS 활동 캡처에 CloudTrail 추적을 사용하는 방법에 대한 자세한 내용은 AWS CloudTrail 사용 설명서의 [CloudTrail 추적 작업](#)을 참조하세요.
- AWS 암호화 솔루션을 AWS 서비스 내의 모든 기본 보안 컨트롤과 함께 사용하세요.
- Amazon S3에 저장된 민감한 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용하세요.
- 명령줄 인터페이스 또는 API를 통해 AWS에 액세스할 때 FIPS 140-3 검증된 암호화 모듈이 필요한 경우, FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 [Federal Information Processing Standard\(FIPS\) 140-3](#)을 참조하세요.

고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 이름 필드와 같은 자유 형식 텍스트 필드에 입력하지 않는 것이 좋습니다. 여기에는 AWS SDK for Go 또는 기타 AWS 서비스에서 콘솔, API, AWS CLI 또는 AWS SDK를 사용하여 작업하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 텍스트 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버로 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 자격 증명 정보를 URL에 포함해서는 안 됩니다.

의 규정 준수 검증AWS SDK for Go

AWS 서비스가 특정 규정 준수 프로그램의 범위에 포함되는지 알아보려면 [규정 준수 프로그램 제공 범위 내 AWS 서비스](#)를 참조하고 관심 있는 규정 준수 프로그램을 선택하십시오. 일반적인 정보는 [AWS 규정 준수 프로그램](#)을 참조하세요.

AWS Artifact(을)를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다. 자세한 내용은 [AWS Artifact에서 보고서 다운로드](#)를 참조하세요.

AWS 서비스 사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률 및 규정에 따라 결정됩니다. AWS 서비스 사용 시 규정 준수 책임에 대한 자세한 내용은 [AWS 보안 설명서](#)를 참조하세요.

AWS SDK for Go의 복원성

AWS 글로벌 인프라는 AWS 리전 및 가용 영역을 중심으로 구축됩니다. AWS 리전에서는 물리적으로 분리되고 격리된 다수의 가용 영역을 제공하며 이러한 가용 영역은 짧은 대기 시간, 높은 처리량 및 높은 중복성을 갖춘 네트워크에 연결되어 있습니다. 가용 영역을 사용하면 중단 없이 영역 간에 자동으로 장애 극복 조치가 이루어지는 애플리케이션 및 데이터베이스를 설계하고 운영할 수 있습니다. 가용 영역은 기존의 단일 또는 다중 데이터 센터 인프라보다 가용성, 내결함성, 확장성이 뛰어납니다.

AWS 리전 및 가용 영역에 대한 자세한 내용은 [AWS 글로벌 인프라](#) 섹션을 참조하세요.

AWS SDK for Go v2 개발자 가이드용 문서 기록

다음 표에서는 AWS SDK for Go v2에 대한 문서 릴리스를 설명합니다.

변경 사항	설명	날짜
HTTP 인터셉터	HTTP 인터셉터, 미들웨어 대비 이점, 사용 가능한 후크 및 등록 방법을 다루는 새 섹션입니다.	2025년 9월 4일
체크섬을 통한 데이터 무결성 보호	자동 체크섬 계산에 대한 세부 정보로 콘텐츠가 업데이트되었습니다.	2025년 1월 16일
최초 릴리스입니다.	AWS SDK for Go v2 개발자 안내서 최초 출시.	2024년 10월 31일

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.