aws

개발자 가이드

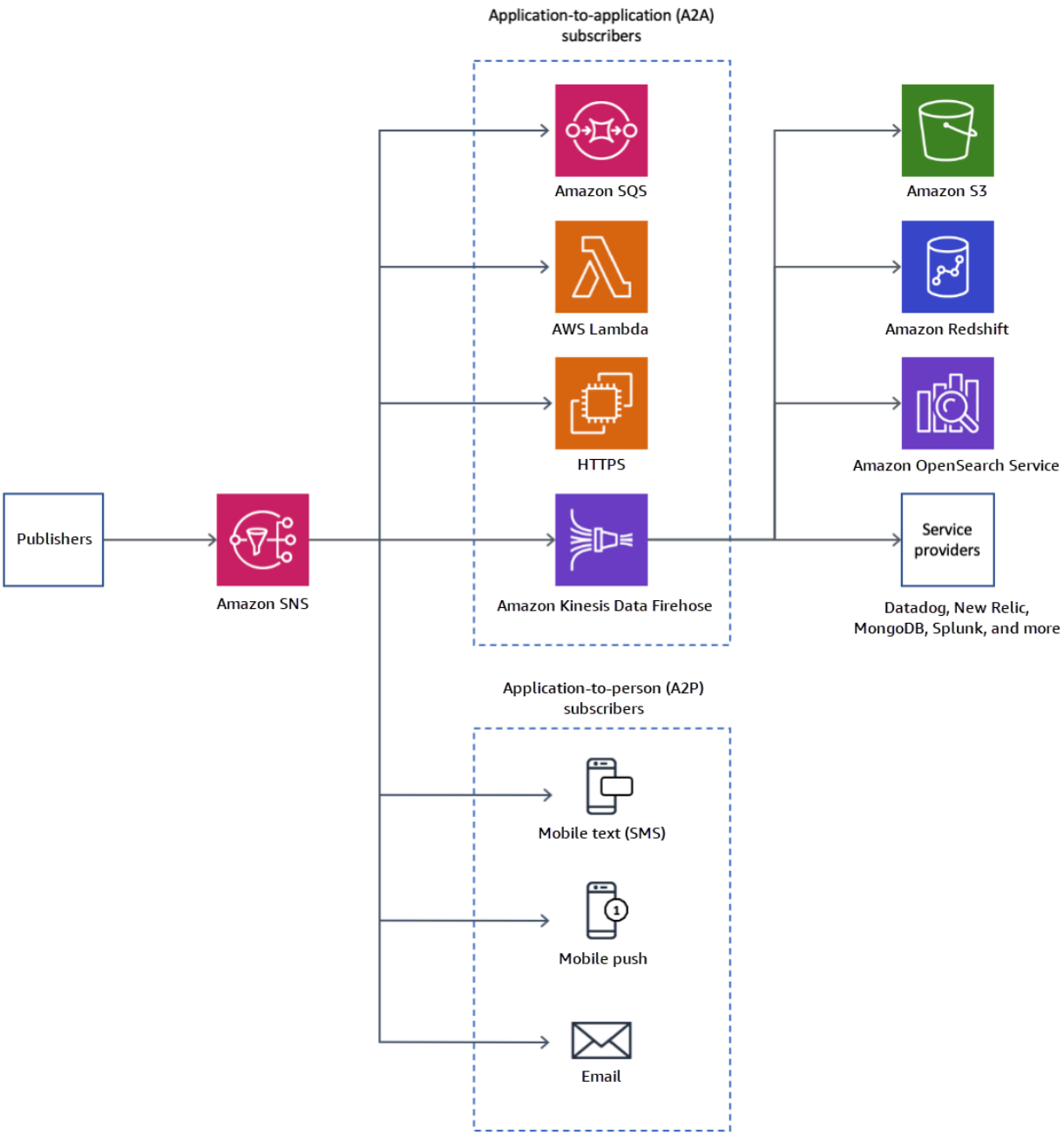
Amazon Simple Notification Service

Amazon Simple Notification Service: 개발자 가이드

Table of Contents

Amazon SNS란 무엇인가요?

Amazon Simple Notification Service(Amazon SNS)는 게시자에서 구독자(생산자 및 소비자라고도 함)로 메시지를 전송하는 관리형 서비스입니다. 게시자는 논리적 액세스 지점 및 커뮤니케이션 채널인 주제에 메시지를 전송하여 구독자와 비동기식으로 통신합니다. 클라이언트는 Amazon Data Firehose, Amazon SQS AWS Lambda, HTTP, 이메일, 모바일 푸시 알림, 모바일 문자 메시지 (SMS) 등 지원되는 엔드포인트 유형을 사용하여 SNS 주제를 구독하고 게시된 메시지를 수신할 수 있습니다.



주제

- 특징 및 기능
- 관련 서비스
- Amazon SNS 액세스

- Amazon SNS 요금
- 일반적인 Amazon SNS 시나리오
- AWS SDK와 함께 Amazon SNS 사용

특징 및 기능

Amazon SNS는 다음과 같은 특징 및 기능을 제공합니다.

- 메시지 pplication-to-application

  pplication-to-application 메시징은 Amazon Data Firehose 전송 스트림, Lambda 함수, Amazon SQS 대기열, HTTP/S 엔드포인트, 이벤트 포크 파이프라인과 같은 구독자를 지원합니다. AWS 자세한 정보는 애플리케이션 간(A2A) 메시징을 참조하세요.
- pplication-to-person A. 알림

  pplication-to-person 알림은 모바일 애플리케이션, 휴대폰 번호, 이메일 주소와 같은 사용자 알림을 구독자에게 제공합니다. 자세한 정보는 애플리케이션 대 개인(A2P) 메시징을 참조하세요.
- 표준 및 FIFO 주제

  FIFO 주제를 사용하여 엄격한 메시지 순서를 보장하고, 메시지 그룹을 정의하고, 메시지 중복을 방지할 수 있습니다. FIFO 대기열과 표준 대기열을 모두 사용하여 FIFO 주제를 구독할 수 있습니다. 자세한 정보는 메시지 순서 지정 및 중복 제거(FIFO 주제)을 참조하세요.

  메시지 전송 순서와 잠재적 메시지 중복이 중요하지 않은 경우 표준 주제를 사용합니다. 지원되는 모든 전송 프로토콜은 표준 주제를 구독할 수 있습니다.
- 메시지 내구성

  Amazon SNS는 메시지 내구성을 제공하기 위해 함께 작동하는 다양한 전략을 사용합니다.
  - 게시된 메시지는 지리적으로 분리된 여러 서버 및 데이터 센터에 저장됩니다.
  - 구독된 엔드포인트를 사용할 수 없는 경우 Amazon SNS는 전송 재시도 정책을 실행합니다.
  - 전송 재시도 정책이 종료되기 전에 전송되지 않은 메시지를 보존하기 위해 배달 못한 편지 대기열을 만들 수 있습니다.
- 메시지 아카이브, 재생 및 분석

  Firehose 전송 스트림을 SNS 주제에 구독하는 등 다양한 방법으로 Amazon SNS를 통해 메시지를 보관할 수 있습니다. 이를 통해 Amazon Simple Storage Service (Amazon S3) 버킷, Amazon Redshift 테이블 등과 같은 분석 엔드포인트에 알림을 보낼 수 있습니다. 또한 Amazon SNS FIFO 주

제는 코드 없는 인플레이스 메시지 아카이브로 메시지 아카이브 및 재생을 지원하므로 주제 소유자가 주제 내에 메시지를 저장(또는 아카이브)할 수 있습니다. 그런 다음 주제 구독자는 구독한 엔드포인트로 아카이브된 메시지를 다시 가져올(또는 재생할) 수 있습니다. 자세한 내용은 FIFO 주제의 메시지 아카이브 및 재생 섹션을 참조하세요.

- 메시지 속성

  메시지 속성을 사용하여 메시지에 대한 임의 메타데이터를 제공할 수 있습니다. the section called “메시지 속성”.

- 메시지 필터링

  기본적으로 각 구독자는 주제에 게시된 모든 메시지를 수신합니다. 메시지의 하위 세트만 수신하려면 구독자는 주제 구독에 필터 정책을 할당해야 합니다. 구독자는 필터 정책 범위를 정의하여 페이로드 기반 또는 속성 기반 필터링을 활성화할 수도 있습니다. 필터 정책 범위의 기본값은 `MessageAttributes`입니다. 수신 메시지 속성이 필터 정책 속성과 일치하면 메시지가 구독된 엔드포인트로 전송됩니다. 그렇지 않으면 메시지가 필터링되어 제외됩니다. 필터 정책 범위가 `MessageBody`인 경우 필터 정책 속성을 페이로드와 일치시킵니다. 자세한 정보는 메시지 필터링을 참조하세요.

- 메시지 보안

  서버 측 암호화는 에서 제공하는 암호화 키를 사용하여 Amazon SNS 주제에 저장된 메시지의 콘텐츠를 보호합니다. AWS KMS자세한 정보는 the section called “저장 시 암호화”을 참조하세요.

  Amazon SNS와 Virtual Private Cloud(VPC) 간에 프라이빗 연결을 설정할 수도 있습니다. 자세한 정보는 the section called “인터네트워크 트래픽 개인 정보 보호”에서 확인하세요.

관련 서비스

Amazon SNS에서 다음 서비스를 사용할 수 있습니다.

- Amazon SQS는 내구력 있고 가용성이 뛰어난 보안 호스팅 대기열을 제공하며 이를 통해 분산 소프트웨어 시스템과 구성 요소를 통합 및 분리할 수 있습니다. Amazon SQS는 다음과 같은 방식으로 Amazon SNS와 관련됩니다.
  - Amazon SNS는 전송할 수 없는 메시지에 대해 Amazon SQS 기반의 배달 못한 편지 대기열을 제공합니다.
  - Amazon SQS 대기열에서 Amazon SNS 주제를 구독할 수 있습니다.
  - Amazon SQS FIFO 대기열 또는 표준 대기열에서 Amazon SNS FIFO 주제를 구독할 수 있습니다. Amazon SQS FIFO 대기열만이 메시지가 중복되지 않고 순서대로 수신되도록 보장합니다.

- AWS Lambda는 새 정보에 신속하게 응답하는 애플리케이션을 구축할 수 있도록 해줍니다. 고가용성 컴퓨팅 인프라에서 Lambda 함수의 애플리케이션 코드를 실행합니다. 자세한 정보는 AWS Lambda 개발자 안내서를 참조하세요. Lambda 함수에서 SNS 주제를 구독할 수 있습니다.
- AWS Identity and Access Management (IAM) 을 사용하면 사용자의 AWS 리소스 액세스를 안전하게 제어할 수 있습니다. IAM을 사용하여 Amazon SNS 주제를 사용할 수 있는 사람(인증), 사용할 수 있는 주제 및 사용 방법(권한 부여)을 제어합니다. 자세한 정보는 Amazon SNS로 자격 증명 기반 정책 사용을 참조하세요.
- AWS CloudFormation AWS 리소스를 모델링하고 설정할 수 있습니다. Amazon SNS 주제 및 구독을 포함하여 원하는 AWS 리소스를 설명하는 템플릿을 만드십시오. AWS CloudFormation 이러한 리소스를 자동으로 프로비저닝하고 구성합니다. 자세한 정보는 AWS CloudFormation 사용 설명서를 참조하세요.

Amazon SNS 액세스

Amazon SNS 콘솔, 명령줄 도구 또는 AWS SDK를 사용하여 SNS 주제 및 구독을 구성하고 관리할 수 있습니다.

- 이 Amazon SNS 콘솔은 주제 및 구독 생성, 메시지 송수신, 이벤트 및 로그 모니터링을 위한 편리한 사용자 인터페이스를 제공합니다.
- AWS Command Line Interface (AWS CLI) 를 사용하면 고급 구성 및 자동화 사용 사례를 위해 Amazon SNS API에 직접 액세스할 수 있습니다. 자세한 정보는 AWS CLI로 Amazon SNS 사용을 참조하세요.
- AWS 다양한 언어로 SDK를 제공합니다. 자세한 정보는 SDK 및 도구 키트를 참조하세요.

Amazon SNS 요금

Amazon SNS에는 선결제 비용이 없습니다. 게시하는 메시지 수, 전송하는 알림 수, 주제 및 구독 관리를 위한 추가 API 호출에 따라 요금을 지불합니다. 전송 요금은 엔드포인트 유형에 따라 다릅니다. Amazon SNS 프리 티어를 통해 무료로 시작할 수 있습니다.

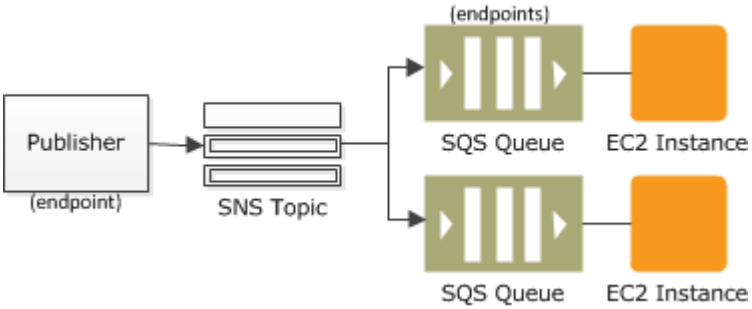
자세한 내용은 Amazon SNS 요금을 참조하세요.

일반적인 Amazon SNS 시나리오

애플리케이션 통합

팬아웃 시나리오는 SNS 주제에 게시된 메시지가 Firehose 전송 스트림, Amazon SQS 대기열, HTTP(S) 엔드포인트 및 Lambda 함수와 같은 여러 엔드포인트에 복제되어 푸시되는 경우입니다. 따라서 평행한 비동시적 처리가 가능합니다.

예를 들어, 사용자는 제품에 대한 주문이 생성될 때 SNS 주제에 메시지를 전송하는 애플리케이션을 개발할 수 있습니다. 그러면 해당 SNS 주제를 구독하는 SQS 대기열은 새 주문에 대해 동일한 알림을 수신합니다. SQS 대기열 중 하나에 연결된 Amazon Elastic Compute Cloud(Amazon EC2) 서버 인스턴스는 주문 처리 또는 이행을 처리할 수 있습니다. 또한 수신된 모든 주문을 분석하기 위해 다른 Amazon EC2 서버 인스턴스를 데이터 웨어하우스에 연결할 수 있습니다.



또한 팬아웃을 사용하여 테스트 환경과 함께 프로덕션 환경으로 전송된 데이터를 복제할 수 있습니다. 기존 예를 확대해보면, 사용자는 다른 SQS 대기열에서 새로운 수신 주문에 대해 동일한 SNS 주제를 구독할 수 있습니다. 그러면 이 새로운 SQS 대기열을 테스트 환경에 연결함으로써 사용자는 개선을 계속하면서 프로덕션 환경에서 수신한 데이터를 사용하여 애플리케이션을 테스트할 수 있습니다.

> **Important**
>
> 프로덕션 데이터를 테스트 환경에 보내기 전에 데이터 프라이버시 및 보안을 고려해야 합니다.

자세한 정보는 다음 리소스를 참조하세요.

- 팬아웃에서 Firehose로의 전송 스트림
- Lambda 함수로 팬아웃
- Amazon SQS 대기열로 팬아웃
- HTTP(S) 엔드포인트로 팬아웃

- Amazon SNS와 AWS 컴퓨팅, 스토리지, 데이터베이스, 네트워킹 서비스를 사용한 이벤트 기반 컴퓨팅

애플리케이션 알림

애플리케이션 및 시스템 알림은 미리 정의된 임계값에 의해 트리거되는 알림입니다. Amazon SNS에서는 SMS와 이메일을 통해 지정된 사용자에게 이러한 알림을 보낼 수 있습니다. 예를 들어 Amazon EC2 Auto Scaling 그룹에 대한 특정 변경, Amazon S3 버킷에 새 파일이 업로드되거나, Amazon에서 측정치 임계값이 위반되는 등의 이벤트가 발생하면 즉시 알림을 받을 수 있습니다. CloudWatch 자세한 내용은 Amazon CloudWatch 사용 설명서의 Amazon SNS 알림 설정을 참조하십시오.

사용자 알림

Amazon SNS에서는 개인 또는 그룹에 푸시 이메일 메시지와 문자 메시지(SMS 메시지)를 전송할 수 있습니다. 예를 들어 사용자는 전자 상거래 주문 확인을 사용자 알림으로 보낼 수 있습니다. Amazon SNS를 사용하여 SMS 메시지를 전송하는 방법에 대한 자세한 정보는 모바일 문자 메시지(SMS)에서 확인하세요.

모바일 푸시 알림

모바일 푸시 알림을 통해 메시지를 모바일 앱으로 바로 전송할 수 있습니다. 예를 들어 Amazon SNS를 사용하여 앱에 업데이트 알림을 전송할 수 있습니다. 알림 메시지는 업데이트를 다운로드 및 설치하기 위한 링크를 포함할 수 있습니다. Amazon SNS를 사용하여 푸시 알림 메시지를 전송하는 방법에 대한 자세한 정보는 모바일 푸시 알림에서 확인하세요.

AWS SDK와 함께 Amazon SNS 사용

AWS 소프트웨어 개발 키트 (SDK) 는 널리 사용되는 여러 프로그래밍 언어에 사용할 수 있습니다. 각 SDK는 개발자가 선호하는 언어로 애플리케이션을 쉽게 구축할 수 있도록 하는 API, 코드 예시 및 설명서를 제공합니다.

SDK 설명서	코드 예시
AWS SDK for C++	AWS SDK for C++ 코드 예제
AWS CLI	AWS CLI 코드 예제
AWS SDK for Go	AWS SDK for Go 코드 예제

SDK 설명서	코드 예시
AWS SDK for Java	AWS SDK for Java 코드 예제
AWS SDK for JavaScript	AWS SDK for JavaScript 코드 예제
AWS SDK for Kotlin	AWS SDK for Kotlin 코드 예제
AWS SDK for .NET	AWS SDK for .NET 코드 예제
AWS SDK for PHP	AWS SDK for PHP 코드 예제
AWS Tools for PowerShell	PowerShell 코드 예제를 위한 도구
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) 코드 예제
AWS SDK for Ruby	AWS SDK for Ruby 코드 예제
AWS SDK for Rust	AWS SDK for Rust 코드 예제
AWS SDK for SAP ABAP	AWS SDK for SAP ABAP 코드 예제
AWS SDK for Swift	AWS SDK for Swift 코드 예제

Amazon SNS 관련 예는 AWS SDK를 사용하는 Amazon SNS의 코드 예제에서 확인하세요.

> **가용성 예**
>
> 필요한 예제를 찾을 수 없습니까? 이 페이지 하단의 피드백 제공 링크를 사용하여 코드 예시를 요청하세요.

Amazon SNS 이벤트 소스 및 대상

Amazon SNS는 수많은 AWS 소스에서 이벤트 기반 알림을 수신하고 알림을 애플리케이션 간(A2A) 및 애플리케이션 간(A2P) 대상으로 팬아웃할 수 있습니다. 이 섹션에서는 지원되는 이벤트 소스 및 대상을 나열하고 자세한 내용을 볼 수 있는 링크를 제공합니다.

주제

Amazon SNS 이벤트 소스

이 페이지에는 AWS 제품 범주별로 그룹화된 Amazon SNS 주제에 이벤트를 게시할 수 있는 AWS 서비스가 나열되어 있습니다.

> **Note**
>
> Amazon SNS는 2020년 10월에 FIFO 주제를 도입했습니다. 현재, 대부분의 AWS 서비스는 SNS 표준 주제에서만 이벤트 전송을 지원합니다.

분석 서비스

AWS 서비스	Amazon SNS에서 사용하면 얻을 수 있는 혜택
Amazon Athena – 표준 SQL을 사용해 Amazon S3의 데이터를 분석할 수 있습니다.	제어 제한을 초과하면 알림을 수신합니다. 자세한 정보는 Amazon Athena 사용 설명서의 데이터 사용량 제어 제한 설정을 참조하세요.
AWS Data Pipeline – 데이터 이동 및 변환을 자동화할 수 있습니다.	파이프라인 구성 요소의 상태에 대한 알림을 수신합니다. 자세한 내용은 AWS Data Pipeline 개발자 안내서의 SnsAlarm을 참조하세요.
Amazon Redshift – 데이터 웨어하우스의 설정, 운영 및 조정 작업을 모두 관리합니다.	Amazon Redshift 이벤트에 대한 알림을 수신합니다. 자세한 정보는 Amazon Redshift 관리 안

AWS 서비스	Amazon SNS에서 사용하면 얻을 수 있는 혜택
	내서의 Amazon Redshift 이벤트 알림을 참조하세요.

애플리케이션 통합 서비스

AWS 서비스	Amazon SNS에서 사용하면 얻을 수 있는 혜택
Amazon EventBridge — 자체 애플리케이션, SaaS software-as-a-service (애플리케이션) 및 AWS 서비스에서 실시간 데이터 스트림을 전달하고 해당 데이터를 Amazon SNS를 포함한 대상으로 라우팅합니다. EventBridge 이전에는 이벤트라고 불렸습니다. CloudWatch	이벤트 알림을 받습니다. EventBridge 자세한 내용은 Amazon EventBridge 사용 설명서의 Amazon EventBridge 대상을 참조하십시오.
AWS Step Functions – 이를 통해 AWS Lambda 기능과 기타 AWS 서비스를 결합하여 비즈니스 크리티컬 애플리케이션을 구축할 수 있습니다.	Step Functions 이벤트에 대한 알림을 수신합니다. 자세한 정보는 AWS Step Functions 개발자 안내서의 Step Functions로 Amazon SNS 호출을 참조하세요.

Billing and Cost Management 서비스

AWS 서비스	Amazon SNS에서 사용하면 얻을 수 있는 혜택
AWS Billing and Cost Management – 비용을 모니터링하고 청구서를 결제하는 데 도움이 되는 기능을 제공합니다.	예산 알림, 가격 변경 알림 및 이상 알림을 수신합니다. 자세한 정보는 AWS Billing 사용 설명서에서 다음 페이지를 참조하세요. • 예산 알림을 위한 Amazon SNS 주제 생성 • 알림 설정 • AWS 비용 이상 탐지로 비정상적인 지출 탐지

비즈니스 애플리케이션 서비스

AWS 서비스	Amazon SNS에서 사용하면 얻을 수 있는 혜택
Amazon Chime- 조직 내부 및 외부에서 회의, 채팅 및 비즈니스 통화를 수행할 수 있습니다.	중요한 회의 이벤트 알림을 수신합니다. 자세한 정보는 Amazon Chime 개발자 안내서의 Amazon Chime SDK 이벤트 알림을 참조하세요.

컴퓨팅 서비스

AWS 서비스	Amazon SNS에서 사용하면 얻을 수 있는 혜택
Amazon EC2 Auto Scaling – 애플리케이션 로드를 처리하는 데 사용할 수 있는 Amazon Elastic Compute Cloud(Amazon EC2) 인스턴스를 올바른 수로 확보하도록 지원합니다.	Auto Scaling 그룹 내 Amazon EC2 인스턴스를 시작하거나 종료하면 알림을 수신합니다. 자세한 정보는 Amazon EC2 Auto Scaling 사용 설명서에서 Auto Scaling 그룹 조정 시 Amazon SNS 알림 수신을 참조하세요.
EC2 Image Builder — 특정 IT 표준을 충족하는 소프트웨어 및 설정으로 사전 설치 up-to-date 및 사전 구성된 사용자 지정되고 안전한 서버 이미지의 생성, 관리 및 배포를 자동화합니다.	구축이 완료되면 알림을 수신합니다. 자세한 정보는 AWS 컴퓨팅 블로그의 EC2 Image Builder 파이프라인에서 최신 서버 이미지 추적을 참조하세요.
AWS Elastic Beanstalk – 애플리케이션에 대한 용량 프로비저닝, 로드 밸런싱 및 조정에 대한 세부 정보를 처리하고 애플리케이션 상태 모니터링을 제공합니다.	애플리케이션에 영향을 미치는 중요 이벤트에 대한 알림을 수신합니다. 자세한 정보는 AWS Elastic Beanstalk 개발자 안내서의 Amazon SNS를 통한 Elastic Beanstalk 환경 알림을 참조하세요.
AWS Lambda – 서버를 프로비저닝하거나 관리할 필요 없이 코드를 실행할 수 있습니다.	SNS 주제를 Lambda 배달 못한 편지 대기열 또는 Lambda 대상으로 설정하여 함수 출력 데이터를 수신합니다. 자세한 정보는 AWS Lambda 개발자 안내서의 비동기식 호출을 참조하세요.

AWS 서비스	Amazon SNS에서 사용하면 얻을 수 있는 혜택
Amazon Lightsail – 개발자가 AWS를 이용해 웹 사이트 또는 웹 애플리케이션을 구축하는 데 사용할 수 있도록 도와줍니다.	인스턴스, 데이터베이스 또는 로드 밸런서 중 하나에 대한 지표가 지정된 임계값을 초과하면 알림을 수신합니다. 자세한 정보는 Amazon Lightsail 개발자 안내서의 Amazon Lightsail에서 알림 연락처 추가를 참조하세요.

컨테이너 서비스

AWS 서비스	Amazon SNS에서 사용하면 얻을 수 있는 혜택
Amazon EKS Distro – 애플리케이션이 배포되는 위치에 관계없이 안정적이고 안전한 클러스터를 생성할 수 있습니다.	Amazon EKS Distro로 생성된 클러스터에 대한 업데이트 및 보안 패치를 추적합니다. 자세한 정보는 Amazon EKS Distro 소개 - Amazon EKS에서 사용하는 오픈 소스 Kubernetes 배포를 참조하세요.
Amazon Elastic Container Service(Amazon ECS) – 클러스터에서 컨테이너를 실행, 중지 및 관리할 수 있습니다.	새로운 Amazon ECS 최적화 AMI를 사용할 수 있을 때 알림을 수신합니다. 자세한 정보는 Amazon Elastic Container Service 개발자 안내서의 Amazon ECS 최적화 AMI 업데이트 알림 구독을 참조하세요.

고객 참여 서비스

AWS 서비스	Amazon SNS에서 사용하면 얻을 수 있는 혜택
Amazon Connect – 옴니채널 클라우드 고객 센터를 설정하여 고객과 협력하도록 지원합니다.	알림 및 검증을 수신합니다. 자세한 정보는 Amazon Connect 관리자 가이드의 Amazon Connec와 함께 사용할 경우 AWS의 강점을 참조하세요.

AWS 서비스	Amazon SNS에서 사용하면 얻을 수 있는 혜택
Amazon Pinpoint – 이메일, SMS 및 음성 메시지, 푸시 알림을 보내 고객의 참여를 유도할 수 있습니다.	고객의 메시지를 수신할 수 있는 양방향 SMS를 구성합니다. 자세한 정보는 Amazon Pinpoint 사용 설명서의 Amazon Pinpoint에서 양방향 SMS 메시징 사용을 참조하세요.
Amazon Simple Email Service(Amazon SES) – 사용자의 이메일 주소와 도메인을 사용해 이메일을 보내고 받기 위한 경제적인 방법을 제공합니다.	반송 메일, 불만 사항 및 전송에 대한 알림을 수신합니다. 자세한 정보는 Amazon Simple Email Service 개발자 안내서의 Amazon SES에 대한 Amazon SNS 알림 구성을 참조하세요.

데이터베이스 서비스

AWS 서비스	Amazon SNS에서 사용하면 얻을 수 있는 혜택
AWS Database Migration Service – 온프레미스 데이터베이스에서 AWS 클라우드로 데이터를 마이그레이션합니다.	복제 인스턴스가 생성되거나 삭제되는 등의 AWS DMS 이벤트가 발생할 때 알림을 수신합니다. 자세한 정보는 AWS Database Migration Service 사용 설명서에서 AWS Database Migration Service의 이벤트 및 알림 작업을 참조하세요.
Amazon DynamoDB – 이 완전관리형 NoSQL 데이터베이스 서비스에서 원활한 확장성과 함께 빠르고 예측 가능한 성능을 제공합니다.	유지 관리 이벤트가 발생할 때 알림을 수신합니다. 자세한 정보는 Amazon DynamoDB 개발자 안내서의 DAX 클러스터 설정 사용자 지정을 참조하세요.
Amazon ElastiCache — 크기 조정이 가능하고 비용 효율적인 고성능 인메모리 캐시를 제공하는 동시에 분산 캐시 환경 배포 및 관리와 관련된 복잡성을 제거합니다.	중요한 이벤트가 발생하면 알림을 수신합니다. 자세한 ElastiCache 내용은 Memcached Amazon 사용 설명서의 이벤트 알림 및 Amazon SNS를 참조하십시오.
Amazon Neptune – 고도로 연결된 데이터 세트에서 작동하는 애플리케이션을 구축하고 실행할 수 있습니다.	Neptune 이벤트가 발생하면 알림을 수신합니다. 자세한 정보는 Neptune 사용 설명서의 Neptune 이벤트 알림 사용을 참조하세요.

AWS 서비스	Amazon SNS에서 사용하면 얻을 수 있는 혜택
Amazon Redshift – 데이터 웨어하우스의 설정, 운영 및 조정 작업을 모두 관리합니다.	Amazon Redshift 이벤트에 대한 알림을 수신합니다. 자세한 정보는 Amazon Redshift 관리 안내서의 Amazon Redshift 이벤트 알림을 참조하세요.
Amazon Relational Database Service – AWS 클라우드에서 관계형 데이터베이스를 더 쉽게 설치, 운영 및 확장할 수 있도록 합니다.	Amazon RDS 이벤트에 대한 알림을 수신합니다. 자세한 정보는 Amazon RDS 사용 설명서의 Amazon RDS 이벤트 알림 사용을 참조하세요.

개발자 도구 서비스

AWS 서비스	Amazon SNS에서 사용하면 얻을 수 있는 혜택
AWS CodeBuild – 소스 코드를 컴파일하고 단위 테스트를 실행하며 배포 준비가 완료된 결과물을 생성합니다.	구축이 성공 또는 실패하거나 한 구축 단계에서 다른 구축 단계로 이동할 때 알림을 수신합니다. 자세한 내용은 CodeBuild AWS CodeBuild사용 설명서의 빌드 알림 샘플을 참조하십시오.
AWS CodeCommit – 클라우드에서 자산을 비공개로 저장하고 관리하기 위한 버전 제어 기능을 제공합니다.	CodeCommit 리포지토리 이벤트에 대한 알림을 받습니다. 자세한 정보는 AWS CodeCommit 사용 설명서의 예: Amazon SNS 주제에 대한 AWS CodeCommit 트리거 생성을 참조하세요.
AWS CodeDeploy – Amazon EC2 인스턴스, 온프레미스 인스턴스, 서버리스 Lambda 함수 또는 Amazon ECS 서비스로 애플리케이션 배포를 자동화합니다.	CodeDeploy 배포 또는 인스턴스 이벤트에 대한 알림을 받습니다. 자세한 내용은 AWS CodeDeploy사용 설명서의 CodeDeploy 이벤트 트리거 만들기를 참조하십시오.
Amazon CodeGuru — 라이브 애플리케이션에서 런타임 성능 데이터를 수집하고 애플리케이션 성능을 미세 조정하는 데 도움이 되는 권장 사항을 제공합니다.	이상 현상이 발생하면 알림을 수신합니다. 자세한 내용은 Amazon CodeGuru 사용 설명서의 예외 항목 및 권장 사항 보고서 사용을 참조하십시오.

AWS 서비스	Amazon SNS에서 사용하면 얻을 수 있는 혜택
AWS CodePipeline – 소프트웨어 변경 사항을 지속적으로 릴리스하는 데 필요한 단계를 자동화합니다.	승인 작업에 대한 알림을 수신합니다. 자세한 내용은 사용 설명서의 승인 조치 관리를 참조하십시오. CodePipeline AWS CodePipeline
AWS CodeStar – AWS에 대한 소프트웨어 개발 프로젝트를 생성, 관리 및 작업합니다.	사용하는 리소스에서 발생하는 이벤트에 관한 알림을 수신합니다. 자세한 정보는 개발자 도구 콘솔 사용 설명서의 알림에 대한 Amazon SNS 주제 구성을 참조하세요.

프런트 엔드 웹 및 모바일 서비스

AWS 서비스	Amazon SNS에서 사용하면 얻을 수 있는 혜택
Amazon Pinpoint – 이메일, SMS 및 음성 메시지, 푸시 알림을 보내 고객의 참여를 유도할 수 있습니다.	고객의 메시지를 수신할 수 있는 양방향 SMS를 구성합니다. 자세한 정보는 Amazon Pinpoint 사용 설명서의 Amazon Pinpoint에서 양방향 SMS 메시징 사용을 참조하세요.

게임 개발 서비스

AWS 서비스	Amazon SNS에서 사용하면 얻을 수 있는 혜택
Amazon GameLift — 게임 서버의 배포, 운영 및 확장을 위한 완전 관리형 서비스를 포함하여 클라우드에서 세션 기반 멀티플레이어 게임 서버를 호스팅하기 위한 솔루션을 제공합니다.	매치메이킹 및 대기열 이벤트 알림을 수신합니다. 자세한 정보는 다음 페이지를 참조하세요. • 매치메이킹 알림에 대해서는 Amazon GameLift FlexMatch 개발자 FlexMatch 안내서의 이벤트 알림 설정을 참조하십시오. • 대기열 알림에 대해서는 Amazon GameLift 개발자 안내서의 게임 세션 배치를 위한 이벤트 알림 설정을 참조하십시오.

사물 인터넷 서비스

AWS 서비스	Amazon SNS에서 사용하면 얻을 수 있는 혜택
AWS IoT Core – 사용자의 IoT 디바이스를 다른 디바이스 및 AWS 클라우드 서비스에 연결하는 클라우드 서비스를 제공합니다.	AWS IoT Core 이벤트에 대한 알림을 수신합니다. 자세한 정보는 AWS IoT 개발자 안내서의 Amazon SNS 규칙 생성을 참조하세요.
AWS IoT Device Defender – 디바이스 구성을 감사하고 연결된 디바이스를 모니터링하여 비정상적인 동작을 감지하고 보안 위험을 완화할 수 있습니다.	디바이스가 동작을 위반하면 경보를 수신합니다. 자세한 정보는 AWS IoT 개발자 안내서의 AWS IoT Device Defender 감지 사용 방법을 참조하세요.
AWS IoT Events – 장비 또는 디바이스 집합에서 오류 또는 작동 변경 사항을 모니터링하고 이러한 이벤트가 발생할 때 작업을 트리거할 수 있습니다.	AWS IoT Events 이벤트에 대한 알림을 수신합니다. 자세한 정보는 AWS IoT Events 개발자 안내서의 Amazon Simple Notification Service를 참조하세요.
AWS IoT Greengrass – AWS를 물리적 디바이스로 확장하여 관리, 분석 및 내구성 있는 스토리지를 위해 클라우드를 계속 사용하면서 생성된 데이터에 대해 로컬로 작업할 수 있도록 합니다.	AWS IoT Greengrass 이벤트에 대한 알림을 수신합니다. 자세한 정보는 AWS IoT Greengrass Version 1 개발자 안내서의 SNS 커넥터를 참조하세요.

기계 학습 서비스

AWS 서비스	Amazon SNS에서 사용하면 얻을 수 있는 혜택
Amazon CodeGuru — 라이브 애플리케이션에서 런타임 성능 데이터를 수집하고 애플리케이션 성능을 미세 조정하는 데 도움이 되는 권장 사항을 제공합니다.	이상 현상이 발생하면 알림을 수신합니다. 자세한 내용은 Amazon CodeGuru 사용 설명서의 예외 항목 및 권장 사항 보고서 사용을 참조하십시오.
Amazon DevOps Guru — 기계 학습을 사용하여 운영 애플리케이션의 성능을 개선하는 데 도움이 되는 운영 통찰력을 생성합니다.	인사이트 및 확인 사항을 전달합니다. 자세한 내용은 AWS관리 및 거버넌스 블로그에서 PagerDuty Amazon DevOps Guru를 통해 대기

AWS 서비스	Amazon SNS에서 사용하면 얻을 수 있는 혜택
	중인 팀에 ML 기반 운영 인사이트 제공을 참조하십시오.
Amazon Lookout for Metrics – 데이터에서 이상을 찾고 근본 원인을 파악하여 신속하게 조치를 취할 수 있습니다.	이상에 대한 알림을 수신합니다. 자세한 정보는 Amazon Lookout for Metrics 개발자 안내서의 Lookout for Metrics와 함께 Amazon SNS 사용을 참조하세요.
Amazon Rekognition – 애플리케이션에 이미지 및 비디오 분석을 추가할 수 있습니다.	요청 결과에 대한 알림을 수신합니다. 자세한 정보는 Amazon Rekognition 개발자 안내서의 참조: 비디오 분석 결과 알림을 참조하세요.
Amazon SageMaker — 데이터 과학자와 개발자가 기계 학습 모델을 구축 및 교육한 다음 프로덕션 준비가 완료된 호스팅 환경에 직접 배포할 수 있도록 지원합니다.	데이터 객체에 레이블이 지정되면 알림을 수신합니다. 자세한 내용은 Amazon SageMaker 개발자 안내서의 스트리밍 레이블 지정 작업 생성을 참조하십시오.

관리 및 거버넌스 서비스

AWS 서비스	Amazon SNS에서 사용하면 얻을 수 있는 혜택
AWS Chatbot— 소프트웨어 개발 팀이 Amazon Chime 및 Slack 채팅방을 사용하여 클라우드의 운영 이벤트를 모니터링하고 이에 대응할 수 있도록 합니다. DevOps AWS	채팅룸에 알림을 전달합니다. 자세한 정보는 AWS Chatbot 관리자 가이드의 AWS Chatbot 설정을 참조하세요.
AWS CloudFormation – AWS 인프라 배포를 예상한 대로 반복해서 생성 및 프로비저닝할 수 있습니다.	스택이 생성되고 업데이트되면 알림을 수신합니다. 자세한 정보는 AWS CloudFormation 사용 설명서의 AWS CloudFormation 스택 옵션 설정을 참조하세요.
AWS CloudTrail – 사용자의 AWS 계정 활동에 대한 이벤트 기록을 제공합니다.	Amazon S3 버킷에 새 로그 파일을 CloudTrail 게시할 때 알림을 받습니다. 자세한 내용은 AWS CloudTrail사용 설명서의 Amazon SNS 알림 구성을 참조하십시오. CloudTrail

AWS 서비스	Amazon SNS에서 사용하면 얻을 수 있는 혜택
Amazon CloudWatch — 실행 중인 AWS 리소스와 애플리케이션을 AWS 실시간으로 모니터링합니다.	경보 상태가 변경되면 알림을 수신합니다. 자세한 내용은 Amazon 사용 설명서의 Amazon CloudWatch 경보 사용을 참조하십시오. CloudWatch
AWS Config – AWS 계정에 있는 AWS 리소스의 구성을 자세히 보여줍니다.	리소스가 업데이트되거나 AWS Config에서 리소스에 대해 사용자 지정 또는 관리형 규칙을 평가할 때 알림을 수신합니다. 자세한 정보는 AWS Config 개발자 안내서의 AWS Config에서 SNS 주제로 전송되는 알림 및 구성 항목 변경 알림 예를 참조하세요.
AWS Control Tower – 안전하고 규정을 준수하는 다중 계정 AWS 환경을 설정하고 관리할 수 있습니다.	알림을 사용하여 랜딩 존 내 드리프트를 방지하고 규정 준수 알림을 수신합니다. 자세한 정보는 AWS Control Tower 사용 설명서의 Amazon Simple Notification Service를 통해 알림 추적을 참조하세요.
AWS License Manager – AWS 및 온프레미스 환경 전반에 걸쳐 소프트웨어 공급 업체의 소프트웨어 라이선스를 중앙 관리하는 데 도움이 됩니다.	License Manager 알림 및 경고를 수신합니다. 자세한 내용은 License Manager 사용 설명서의 License Manager의 설정 및 AWS관리 및 거버넌스 블로그의 AWS License Manager 알림 ServiceNow 인시던트 생성을 참조하십시오.
AWS Service Catalog - IT 관리자가 승인된 제품으로 구성된 포트폴리오를 생성 및 관리하고 최종 사용자에게 배포할 수 있으며, 최종 사용자는 개인화된 포털에서 필요한 제품에 액세스할 수 있습니다.	스택 이벤트에 대한 알림을 수신합니다. 자세한 정보는 서비스 카탈로그 관리자 가이드의 AWS Service Catalog notification constraints(알림 제약 조건)를 참조하세요.
AWS Systems Manager - AWS에서 인프라를 확인하고 제어할 수 있습니다.	명령의 상태에 대한 알림을 수신합니다. 자세한 정보는 AWS Systems Manager 사용 설명서의 Amazon SNS 알림을 사용하여 Systems Manager 상태 변경 모니터링을 참조하세요.

미디어 서비스

AWS 서비스	Amazon SNS에서 사용하면 얻을 수 있는 혜택
Amazon Elastic Transcoder – Amazon S3에 저장한 미디어 파일을 소비자 재생 디바이스에 필요한 형식의 미디어 파일로 변환할 수 있습니다.	작업 상태가 변경되면 알림을 수신합니다. 자세한 정보는 Amazon Elastic Transcoder 개발자 안내서의 작업 상태 알림을 참조하세요.

마이그레이션 및 전송 서비스

AWS 서비스	Amazon SNS에서 사용하면 얻을 수 있는 혜택
AWS Application Discovery Service - 온프레미스 서버에 대한 사용 및 구성 데이터를 수집하여 AWS 클라우드로의 마이그레이션을 계획하도록 지원합니다.	AWS CloudTrail을 통해 이벤트에 대한 알림을 수신합니다. 자세한 정보는 Application Discovery Service 사용 설명서의 AWS CloudTrail로 Application Discovery Service API 호출 로깅을 참조하세요.
AWS Database Migration Service – 온프레미스 데이터베이스에서 AWS 클라우드로 데이터를 마이그레이션합니다.	복제 인스턴스가 생성되거나 삭제되는 등의 AWS DMS 이벤트가 발생할 때 알림을 수신합니다. 자세한 정보는 AWS Database Migration Service 사용 설명서에서 AWS Database Migration Service의 이벤트 및 알림 작업을 참조하세요.
AWS Snowball— 물리적 스토리지 디바이스를 사용하여 Amazon S3와 온사이트 데이터 스토리지 위치 간에 대량의 데이터를 faster-than-internet 빠른 속도로 전송합니다.	Snowball 작업에 대한 알림을 수신합니다. 자세한 정보는 다음 자료를 참조하세요. • AWS Snowball 사용 설명서의 Snowball 알림 • 5단계: AWS Snowball Edge 개발자 안내서에서 알림 기본 설정 선택 • 5단계: AWS Snowcone 사용 설명서에서 알림 기본 설정 선택

네트워킹 및 콘텐츠 전송 서비스

AWS 서비스	Amazon SNS에서 사용하면 얻을 수 있는 혜택
Amazon API Gateway — 규모에 상관없이 자체 REST 및 WebSocket API를 생성하고 배포할 수 있습니다.	API Gateway 엔드포인트에 게시된 메시지를 수신합니다. 자세한 정보는 API Gateway 개발자 안내서의 자습서: AWS 통합으로 API Gateway REST API 빌드를 참조하세요.
Amazon CloudFront —.html, .css, .php, 이미지 및 미디어 파일과 같은 정적 및 동적 웹 콘텐츠의 배포 속도를 높입니다.	지정된 지표에 따른 경보가 발생하면 알림을 받습니다. CloudFront 자세한 내용은 Amazon CloudFront 개발자 안내서의 알림 수신을 위한 알람 설정을 참조하십시오.
AWS Direct Connect – 표준 이더넷 광섬유 케이블을 통해 내부 네트워크를 AWS Direct Connect 위치에 연결할 수 있습니다.	AWS Direct Connect 연결의 상태를 모니터링하는 경보가 상태를 변경하면 알림을 수신합니다. 자세한 내용은 AWS Direct Connect사용 설명서의 AWS Direct Connect 연결 모니터링을 위한 CloudWatch 경보 생성을 참조하십시오.
Elastic Load Balancing – 둘 이상의 가용 영역에서 Amazon EC2 인스턴스, 컨테이너, IP 주소 등 여러 대상에 걸쳐 수신되는 트래픽을 자동으로 분산합니다.	로드 밸런서 이벤트에 대해 생성한 경보의 알림을 수신합니다. 자세한 내용은 클래식 로드 밸런서 사용 설명서의 로드 밸런서에 대한 CloudWatch 경보 생성을 참조하십시오.
Amazon Route 53 – 도메인 등록, DNS 라우팅 및 상태 확인을 제공합니다.	상태 확인 상태가 비정상일 때 알림을 수신합니다. 자세한 정보는 Amazon Route 53 개발자 안내서의 상태 확인 상태가 비정상일 때 Amazon SNS 알림을 수신하려면(콘솔)을 참조하세요.
Amazon Virtual Private Cloud(Amazon VPC) – 사용자가 정의한 가상 네트워크로 AWS 리소스를 시작할 수 있습니다.	인터페이스 엔드포인트에서 발생하는 특정 이벤트에 대한 알림을 수신합니다. 자세한 정보는 Amazon VPC 사용 설명서의 엔드포인트 서비스에 대한 알림 생성 및 관리를 참조하세요.

보안, 자격 증명 및 규정 준수 서비스

AWS 서비스	Amazon SNS에서 사용하면 얻을 수 있는 혜택
AWS Directory Service – 다른 AWS 서비스에서 Microsoft Active Directory(AD)를 사용할 수 있는 몇 가지 방법을 제공합니다.	디렉터리 상태가 바뀔 때 이메일 또는 텍스트(SMS) 메시지를 수신합니다. 자세한 정보는 AWS Directory Service 관리 안내서의 디렉터리 상태 알림 구성을 참조하세요.
Amazon GuardDuty — 사용자 AWS 환경에서 예상치 못한 잠재적 무단 또는 악의적인 활동을 식별하는 데 도움이 되는 지속적인 보안 모니터링을 제공합니다.	새로 발표하는 결과 유형, 기존 결과 유형에 대한 업데이트 및 기타 기능 변경에 대한 최신 알림을 수신합니다. 자세한 내용은 Amazon GuardDuty 사용 GuardDuty 설명서의 공지 SNS 구독 주제를 참조하십시오.
Amazon Inspector - Amazon EC2 인스턴스의 네트워크 액세스 가능성과 해당 인스턴스에서 실행되는 애플리케이션의 보안 상태를 테스트합니다.	Amazon Inspector 이벤트에 대한 알림을 수신합니다. 자세한 정보는 Amazon Inspector 사용 설명서의 Amazon Inspector 알림에 대한 SNS 주제 설정을 참조하세요.
AWS Security Hub - AWS 보안 검사를 자동화하고 보안 알림을 중앙 집중화합니다.	추가, 편집 또는 사용 중지된 AWS Security Hub 컨트롤 또는 표준에 대한 알림을 포함하여 AWS Security Hub 공지 사항에 대한 알림을 받으세요. 자세한 내용은 Amazon SNS로 AWS Security Hub 공지 사항 구독을 참조하세요.

서버리스 서비스

AWS 서비스	Amazon SNS에서 사용하면 얻을 수 있는 혜택
Amazon DynamoDB – 이 완전관리형 NoSQL 데이터베이스 서비스에서 원활한 확장성과 함께 빠르고 예측 가능한 성능을 제공합니다.	유지 관리 이벤트가 발생할 때 알림을 수신합니다. 자세한 정보는 Amazon DynamoDB 개발자 안내서의 DAX 클러스터 설정 사용자 지정을 참조하세요.

AWS 서비스	Amazon SNS에서 사용하면 얻을 수 있는 혜택
Amazon EventBridge — 자체 애플리케이션, SaaS software-as-a-service (애플리케이션) 및 AWS 서비스에서 실시간 데이터 스트림을 전달하고 해당 데이터를 Amazon SNS를 포함한 대상으로 라우팅합니다. EventBridge 이전에는 이벤트라고 불렸습니다. CloudWatch	이벤트 알림을 받습니다. EventBridge 자세한 내용은 Amazon EventBridge 사용 설명서의 Amazon EventBridge 대상을 참조하십시오.
AWS Lambda – 서버를 프로비저닝하거나 관리할 필요 없이 코드를 실행할 수 있습니다.	SNS 주제를 Lambda 배달 못한 편지 대기열 또는 Lambda 대상으로 설정하여 함수 출력 데이터를 수신합니다. 자세한 정보는 AWS Lambda 개발자 안내서의 비동기식 호출을 참조하세요.

스토리지 서비스

AWS 서비스	Amazon SNS에서 사용하면 얻을 수 있는 혜택
AWS Backup – 클라우드 및 온프레미스에서 AWS 서비스 전반에 걸친 데이터 백업을 중앙 집중화하고 자동화할 수 있습니다.	AWS Backup 이벤트에 대한 알림을 수신합니다. 자세한 정보는 AWS Backup 개발자 안내서의 Amazon SNS를 사용하여 AWS Backup 이벤트 추적을 참조하세요.
Amazon Elastic File System – Amazon EC2 인스턴스를 위한 파일 스토리지를 제공합니다.	Amazon EFS 이벤트에 대해 생성한 경보의 알림을 수신합니다. 자세한 정보는 Amazon Elastic File System 사용 설명서의 자동 모니터링 도구를 참조하세요.
Amazon S3 Glacier – 자주 사용되지 않는 데이터를 위한 스토리지를 제공합니다.	작업 완료 시 메시지가 SNS 주제에 전송되도록 볼트의 알림 구성을 설정합니다. 자세한 정보는 Amazon S3 Glacier 개발자 안내서의 Amazon S3 Glacier에서 저장소 알림 구성을 참조하세요.
Amazon Simple Storage Service(Amazon S3) – 객체 스토리지를 제공합니다.	Amazon S3 버킷에 변경 사항이 발생하거나 드물게 객체가 대상 리전에 복제되지 않는 경우 알림을 수신합니다. 자세한 정보는 연습: 알림용 버킷 구성(SNS 주제 또는 SQS 대기열) 및

AWS 서비스	Amazon SNS에서 사용하면 얻을 수 있는 혜택
	Amazon Simple Storage Service 사용 설명서의 복제 지표 및 Amazon S3 이벤트 알림으로 진행률 모니터링을 참조하세요.
AWS Snowball— 물리적 스토리지 디바이스를 사용하여 Amazon S3와 온사이트 데이터 스토리지 위치 간에 대량의 데이터를 faster-than-internet 빠른 속도로 전송합니다.	Snowball 작업에 대한 알림을 수신합니다. 자세한 정보는 다음 자료를 참조하세요. • AWS Snowball 사용 설명서의 Snowball 알림 • 5단계: AWS Snowball Edge 개발자 안내서에서 알림 기본 설정 선택 • 5단계: AWS Snowcone 사용 설명서에서 알림 기본 설정 선택

추가 이벤트 소스

소스(Source)	Amazon SNS에서 사용하면 얻을 수 있는 혜택
AWS 일일 기능 업데이트	Amazon SNS 주제를 통해 AWS 릴리스 및 업데이트에 대한 세부 정보를 적시에 받을 수 있습니다. 이러한 릴리스에는 Service AWS Quotas와 AWS 서비스 통합된 Amazon VPC 엔드포인트, Amazon EC2 인스턴스 유형, Amazon 인스턴스 유형, Amazon Nimble Studio 인스턴스 유형, SageMaker Amazon RDS 데이터베이스 엔진 버전, Amazon MSK 아파치 카프카 버전 등이 포함됩니다AWS 리전. AWS 서비스 자세한 내용은 AWS 뉴스 블로그의 Amazon SNS를 통한 AWS 일일 기능 업데이트 구독을 참조하십시오.
AWS IP 주소 범위	Amazon SNS 주제를 통해 AWS IP 범위 변경에 대한 알림을 수신합니다. 자세한 내용은 Amazon Web Services 일반 참조의 AWS IP 주

소스(Source)	Amazon SNS에서 사용하면 얻을 수 있는 혜택
	소 범위 알림 및 AWS 뉴스 블로그의 Amazon SNS를 통한 AWS 퍼블릭 IP 주소 변경 구독을 참조하세요.

이벤트 기반 컴퓨팅에 대한 자세한 정보는 다음과 같은 소스를 참조하세요.

- 이벤트 기반 아키텍처란 무엇인가요?
- AWS 컴퓨팅 블로그의 Amazon SNS 및 AWS 컴퓨팅, 스토리지, 데이터베이스 및 네트워킹 서비스를 사용한 이벤트 기반 컴퓨팅
- AWS 컴퓨팅 블로그의 AWS 이벤트 포크 파이프라인으로 이벤트 기반 아키텍처 강화

Amazon SNS 이벤트 대상

이 페이지에는 이벤트에 대한 정보를 받을 수 있는 모든 대상이 application-to-application (A2A) 메시징과 application-to-person (A2P) 알림별로 그룹화되어 나열됩니다.

> **Note**
>
> Amazon SNS는 2020년 10월에 FIFO 주제를 도입했습니다. 현재, 대부분의 AWS 서비스는 SNS 표준 주제에서만 이벤트 수신을 지원합니다. Amazon SQS는 SNS 표준 및 FIFO 주제 모두에서 이벤트 수신을 지원합니다.

A2A 대상

이벤트 대상	Amazon SNS에서 사용하면 얻을 수 있는 혜택
아마존 데이터 파이어호스	아카이브 및 분석을 위해 전송 스트림에 이벤트를 전송합니다. 전송 스트림을 통해 Amazon Simple Storage Service (Amazon S3), Amazon Redshift, Amazon 서비스 (서비스) 와 OpenSearch 같은 대상 또는 데이터독, 뉴렐릭, MongoDB, Splunk와 같은 타사 목적지로 이벤트를 전송할 수 있습니다. AWS OpenSearch 자세

이벤트 대상	Amazon SNS에서 사용하면 얻을 수 있는 혜택
	한 설명은 팬아웃에서 Firehose로의 전송 스트림 섹션을 참조하세요.
AWS Lambda	사용자 지정 비즈니스 로직의 실행을 트리거하는 함수에 이벤트를 전달합니다. 자세한 설명은 Lambda 함수로 팬아웃 섹션을 참조하세요.
Amazon SQS	애플리케이션 통합을 위해 대기열에 이벤트를 전송합니다. 자세한 설명은 Amazon SQS 대기열로 팬아웃 섹션을 참조하세요.
AWS Event Fork Pipelines	이벤트 백업 및 저장, 이벤트 검색 및 분석 또는 이벤트 재생 파이프라인에 이벤트를 전달합니다. 자세한 설명은 AWS Event Fork Pipelines로 팬아웃 섹션을 참조하세요.
HTTP/S	외부 Webhook에 이벤트를 전달합니다. 자세한 설명은 HTTP(S) 엔드포인트로 팬아웃 섹션을 참조하세요.

A2P 대상

이벤트 대상	Amazon SNS에서 사용하면 얻을 수 있는 혜택
SMS	휴대폰에 이벤트를 문자 메시지로 전달할 수 있습니다. 자세한 설명은 모바일 문자 메시지(SMS) 섹션을 참조하세요.
이메일	이벤트를 받은 편지함에 전자 메일 메시지로 전달합니다. 자세한 설명은 이메일 알림 섹션을 참조하세요.
플랫폼 엔드포인트	기본 푸시 알림으로 휴대폰에 이벤트를 전달합니다. 자세한 설명은 모바일 푸시 알림 섹션을 참조하세요.

이벤트 대상	Amazon SNS에서 사용하면 얻을 수 있는 혜택
AWS Chatbot	Amazon Chime 채팅룸 또는 Slack 채널로 이벤트를 전달하세요. 자세한 정보는 AWS Chatbot 관리자 가이드에서 다음 페이지를 참조하세요. • Amazon AWS Chatbot Chime으로 설정하기 • AWS Chatbot슬랙으로 설정하기 • 다른 AWS 서비스와 함께 AWS Chatbot 사용
PagerDuty	대기 중인 팀에 운영 인사이트를 제공합니다. 자세한 내용은 AWS관리 및 거버넌스 블로그에서 PagerDuty Amazon DevOps Guru를 통해 대기 중인 팀에 ML 기반 운영 인사이트 제공을 참조하십시오.

Note

두 가지 기본 AWS 이벤트 및 사용자 지정 이벤트를 모두 채팅 앱으로 전송할 수 있습니다.

- 기본 AWS 이벤트 — AWS Chatbot을 사용하여 Amazon SNS 주제를 통해 기본 AWS 이벤트를 Amazon Chime 및 Slack으로 보냅니다. 지원되는 네이티브 AWS 이벤트 세트에는AWS Billing and Cost Management,, AWS Health AWS CloudFormation CloudWatch, Amazon 등의 이벤트가 포함됩니다. 자세한 정보는 AWS Chatbot 관리자 가이드의 다른 서비스와 함께 AWS Chatbot 사용을 참조하세요.
- 사용자 지정 이벤트 – Amazon SNS 주제를 통해 사용자 지정 이벤트를 Amazon Chime, Slack 및 Microsoft Teams로 보낼 수도 있습니다. 이렇게 하려면 SNS 주제에 사용자 지정 이벤트를 게시하여 구독된 Lambda 함수에 이벤트를 전달합니다. 그런 다음 Lambda 함수는 채팅 앱의 Webhook를 사용하여 수신자에게 이벤트를 전달합니다. 자세한 정보는 Webhook를 사용하여 Amazon Chime, Slack 또는 Microsoft Teams에 Amazon SNS 메시지를 게시하려면 어떻게 해야 합니까?를 참조하세요.

Amazon SNS에 대한 액세스 설정

Amazon SNS를 처음 사용하려면 먼저 다음 단계를 완료해야 합니다.

주제

1단계: AWS 계정 및 IAM 사용자 생성

서비스에 액세스하려면 먼저 AWS 서비스를 AWS 계정생성해야 합니다. 를 사용하여 활동 및 사용 보고서를 보고 인증 및 액세스를 관리할 수 있습니다. AWS 계정

가입해 보세요. AWS 계정

계정이 없는 경우 다음 단계를 완료하여 계정을 만드세요. AWS 계정

가입하려면 AWS 계정

1. https://portal.aws.amazon.com/billing/signup을 여세요.
2. 온라인 지시 사항을 따르세요.

   등록 절차 중에는 전화를 받고 키패드로 인증 코드를 입력하는 과정이 있습니다.

   에 AWS 계정가입하면 AWS 계정 루트 사용자가 생성됩니다. 루트 사용자에게는 계정의 모든 AWS 서비스 및 리소스 액세스 권한이 있습니다. 보안 모범 사례는 사용자에게 관리 액세스 권한을 할당하고, 루트 사용자만 사용하여 루트 사용자 액세스 권한이 필요한 작업을 수행하는 것입니다.

AWS 가입 절차가 완료된 후 확인 이메일을 보냅니다. 언제든지 https://aws.amazon.com/으로 가서 내 계정(My Account)을 선택하여 현재 계정 활동을 보고 계정을 관리할 수 있습니다.

관리 액세스 권한이 있는 사용자 생성

가입한 후에는 AWS 계정 루트 사용자를 보호하고 관리자 사용자를 AWS IAM Identity Center활성화하고 생성하여 일상적인 작업에 루트 사용자를 사용하지 않도록 하십시오. AWS 계정

AWS 계정 루트 사용자를 보호하세요.

1. Root user를 선택하고 AWS 계정 이메일 주소를 입력하여 계정 AWS Management Console소유자로 로그인합니다. 다음 페이지에서 비밀번호를 입력합니다.

   루트 사용자를 사용하여 로그인하는 데 도움이 필요하면AWS 로그인 사용 설명서의 루트 사용자로 로그인을 참조하세요.
2. 루트 사용자의 다중 인증(MFA)을 활성화합니다.

   지침은 IAM 사용 설명서의 AWS 계정 루트 사용자 (콘솔) 에 대한 가상 MFA 디바이스 활성화를 참조하십시오.

관리 액세스 권한이 있는 사용자 생성

1. IAM Identity Center를 활성화합니다.

   지침은 AWS IAM Identity Center 사용 설명서의 AWS IAM Identity Center설정을 참조하세요.
2. IAM Identity Center에서 사용자에게 관리 액세스 권한을 부여합니다.

   를 ID 소스로 사용하는 방법에 대한 자습서는 사용 설명서의 기본값으로 IAM Identity Center 디렉터리사용자 액세스 구성을 참조하십시오. IAM Identity Center 디렉터리 AWS IAM Identity Center

관리 액세스 권한을 가진 사용자로 로그인

- IAM IDentity Center 사용자로 로그인하려면 IAM IDentity Center 사용자를 생성할 때 이메일 주소로 전송된 로그인 URL을 사용합니다.

  IAM Identity Center 사용자를 사용하여 로그인하는 데 도움이 필요하면 사용 설명서의 AWS 액세스 포털 로그인을 참조하십시오.AWS 로그인

추가 사용자에게 액세스 권한 할당

1. IAM Identity Center에서 최소 권한 적용 모범 사례를 따르는 권한 세트를 생성합니다.

   지침은AWS IAM Identity Center 사용 설명서의 권한 세트 생성을 참조하세요.
2. 사용자를 그룹에 할당하고, 그룹에 Single Sign-On 액세스 권한을 할당합니다.

   지침은AWS IAM Identity Center 사용 설명서의 그룹 추가를 참조하세요.

다음 단계

이제 Amazon SNS를 시작할 준비를 마쳤으므로 주제를 생성하고, 주제 구독을 생성하고, 주제에 메시지를 게시하고, 구독 및 주제를 삭제하는 작업을 시작합니다.

Amazon SNS 시작하기

이 섹션에서는 Amazon SNS를 익숙하게 사용할 수 있도록 Amazon SNS 콘솔을 사용하여 주제, 구독 및 메시지를 관리하는 방법을 설명합니다.

주제

필수 조건

시작하기 전에 Amazon SNS에 대한 액세스 설정의 단계를 완료해야 합니다.

1단계: 주제 생성

1. Amazon SNS 콘솔에 로그인합니다.
2. 왼쪽 탐색 창에서 주제를 선택합니다.
3. 주제 페이지에서 주제 생성을 선택합니다.
4. 기본적으로 콘솔은 FIFO 주제를 만듭니다. 표준을 선택합니다.
5. 세부 정보 섹션에서 주제 이름 (예:) 을 입력합니다 *MyTopic*.
6. 양식의 끝으로 스크롤하고 주제 생성을 선택합니다.

   콘솔에서 새 주제의 세부 정보 페이지가 열립니다.

2단계: 주제 구독을 생성

1. 왼쪽의 탐색 창에서 구독을 선택합니다.
2. 구독 페이지에서 구독 생성을 선택합니다.
3. 구독 생성 페이지에서 주제 ARN 필드를 선택하여 AWS 계정에서 주제 목록을 확인합니다.

4. 이전 단계에서 생성한 주제를 선택합니다.
5. 프로토콜에서 이메일을 선택합니다.
6. 엔드포인트에 알림을 받는 데 사용할 수 있는 이메일 주소를 입력합니다.
7. 구독 생성을 선택합니다.

   콘솔에서 새 구독의 세부 정보 페이지가 열립니다.
8. 이메일 받은 편지함을 확인하고 AWS 알림의 이메일에서 구독 확인을 선택합니다. 발신자 ID는 일반적으로 "no-reply@sns.amazonaws.com"입니다.
9. Amazon SNS가 웹 브라우저를 열고 구독 ID와 함께 구독 확인을 표시합니다.

3단계: 주제에 메시지 게시

1. 왼쪽 탐색 창에서 주제를 선택합니다.
2. 주제 페이지에서 이전에 생성한 주제를 선택한 다음 메시지 게시를 선택합니다.

   콘솔에서 주제에 메시지 게시 페이지가 열립니다.
3. (선택 사항) 메시지 세부 정보 섹션에서 주제를 입력합니다. 예를 들면 다음과 같습니다.

   ```
   Hello from Amazon SNS!
   ```
4. 메시지 본문 섹션에서 모든 전송 프로토콜에 대해 동일한 페이로드를 선택한 후 다음과 같은 메시지 본문을 입력합니다.

   ```
   Publishing a message to an SNS topic.
   ```
5. 메시지 게시를 선택합니다.

   메시지가 주제에 게시되고 콘솔에서 주제의 세부 정보 페이지가 열립니다.
6. 이메일 받은 편지함을 확인하고 게시된 메시지와 함께 Amazon SNS에서 보낸 이메일을 받았는지 확인합니다.

4단계: 구독 및 주제 삭제

1. 탐색 창에서 구독을 선택합니다.
2. 구독 페이지에서 확인된 구독을 선택한 다음 삭제를 선택합니다.

> **Note**
>
> 보류 중인 확인은 삭제할 수 없습니다. 48시간이 지나면 Amazon SNS에서 해당 정보를 자동으로 삭제합니다.

3. 구독 삭제 대화 상자에서 삭제를 선택합니다.

   구독이 삭제됩니다.
4. 탐색 창에서 주제(Topics)를 선택합니다.
5. 주제 페이지에서 주제를 선택한 다음 삭제를 선택합니다.

> **Important**
>
> 주제를 삭제하면, 해당 주제에 대한 모든 구독도 삭제됩니다.

6. 주제 삭제 ***MyTopic***대화 상자에서 `delete me` 를 입력한 다음 삭제를 선택합니다.

   주제가 삭제됩니다.

다음 단계

구독이 있는 주제를 만들고 주제에 메시지를 보냈으므로 이제 다음 작업을 시도해 볼 수 있습니다.

- AWS 개발자 센터를 살펴봅니다.
- 보안 섹션에서 데이터를 보호하는 방법에 대해 알아봅니다.
- 주제에 대해 서버 측 암호화를 사용합니다.
- 암호화된 Amazon Simple Queue Service(Amazon SQS) 대기열에서 구독하는 주제에 대해 서버 측 암호화를 사용합니다.
- AWS Event Fork Pipelines에서 주제를 구독합니다.

Amazon SNS 구성

Amazon SNS 콘솔을 사용하여 Amazon SNS 주제 및 구독을 생성하고 구성할 수 있습니다. Amazon SNS에 대한 자세한 정보는 Amazon SNS란 무엇인가요?에서 확인하세요.

주제

- Amazon SNS 주제 생성
- Amazon SNS 주제 구독
- Amazon SNS 주제 및 구독 삭제
- Amazon SNS 주제 태그 지정

Amazon SNS 주제 생성

Amazon SNS 주제는 커뮤니케이션 채널 역할을 하는 논리적 액세스 포인트입니다. 주제를 사용하면 여러 엔드포인트 (예: Amazon SQS AWS Lambda, HTTP/S 또는 이메일 주소) 를 그룹화할 수 있습니다.

해당 메시지를 필요로 하는 다른 여러 시스템과 연동하는 메시지 생산자 시스템(예: 전자 상거래 웹 사이트)의 메시지를 브로드캐스트하기 위해 생산자 시스템에 대해 주제를 생성할 수 있습니다.

가장 먼저 이루어지고 가장 흔한 Amazon SNS 태스크는 주제를 생성하는 것입니다. 이 페이지에서는 AWS Management Console,, 를 사용하여 주제를 생성하는 AWS SDK for Java방법을 보여줍니다. AWS SDK for .NET

주제가 생성되는 동안 주제 유형(표준 또는 FIFO)을 선택하고 주제의 이름을 지정합니다. 주제를 생성한 후에는 주제 유형이나 이름을 변경할 수 없습니다. 다른 모든 구성 선택 사항은 주제 생성 중에 선택할 수 있는 사항이며 나중에 편집할 수 있습니다.

> **Important**
>
> 개인 식별 정보(PII)나 기타 기밀 정보 또는 민감한 정보를 주제 이름에 추가하지 마십시오. 주제 이름은 CloudWatch 로그를 비롯한 다른 Amazon Web Services에서 액세스할 수 있습니다. 주제 이름은 개인 데이터나 민감한 데이터에 사용하기 위한 것이 아닙니다.

주제

- 를 사용하여 주제를 만들려면 AWS Management Console
- SDK를 AWS 사용하여 주제를 만들려면

를 사용하여 주제를 만들려면 AWS Management Console

1. Amazon SNS 콘솔에 로그인합니다.
2. 다음 중 하나를 수행하십시오.
   - AWS 계정 이전에 생성한 주제가 없는 경우 홈 페이지에서 Amazon SNS 설명을 읽어보십시오.
   - AWS 계정 이전에 주제를 생성한 적이 있는 경우 탐색 패널에서 주제를 선택합니다.
3. 주제 페이지에서 주제 생성을 선택합니다.
4. 주제 생성 페이지의 세부 정보 섹션에서 다음을 수행합니다.
   a. 유형에서 주제 유형(표준 또는 FIFO을 선택합니다.
   b. 주제의 이름을 입력합니다. FIFO 주제의 경우 이름 끝에 .fifo를 추가합니다.
   c. (선택 사항) 주제의 표시 이름을 입력합니다.

      Important

      이메일 엔드포인트를 구독할 때 Amazon SNS 주제 표시 이름과 보내는 이메일 주소(예: no-reply@sns.amazonaws.com)의 문자 수를 합친 것이 UTF-8 320자를 초과해서는 안 됩니다. Amazon SNS 주제의 표시 이름을 구성하기 전에 서드 파티 인코딩 도구를 사용하여 전송하는 주소의 길이를 확인할 수 있습니다.

   d. (선택 사항) FIFO 주제의 경우 콘텐츠 기반 메시지 중복 제거를 선택하여 기본 메시지 중복 제거를 활성화할 수 있습니다. 자세한 정보는 FIFO 주제에 대한 메시지 중복 제거에서 확인하세요.
5. (선택 사항) 암호화 섹션을 확장하고 다음을 수행합니다. 자세한 정보는 저장 시 암호화을 참조하세요.
   a. 암호화 활성을 선택합니다.
   b. AWS KMS 키를 지정합니다. 자세한 정보는 주요 용어을 참조하세요.

      각 KMS 유형에 대해 설명(Description), 계정(Account), KMS ARN이 표시됩니다.

> **Important**
>
> 해당 KMS의 소유자가 아니거나 `kms:ListAliases` 및 `kms:DescribeKey` 권한이 없는 계정으로 로그인하는 경우 Amazon SNS 콘솔에서 해당 KMS에 대한 정보를 볼 수 없습니다.
> KMS의 소유자에게 이 권한을 부여해 달라고 요청해야 합니다. 자세한 정보는 AWS Key Management Service 개발자 안내서의 AWS KMS API 권한: 작업 및 리소스 참조를 참조하세요.

- Amazon SNS용 AWS 관리형 KMS (기본값) 별칭/AWS/sns가 기본적으로 선택됩니다.

  > **Note**
  >
  > 다음 사항에 유의하십시오:
  >
  > - 를 사용하여 주제에 대해 Amazon SNS용 AWS 관리형 KMS를 처음으로 지정하면 Amazon SNS용 AWS 관리형 KMS가 AWS KMS 생성됩니다. AWS Management Console
  > - 또는 SSE가 활성화된 주제에 대해 처음으로 `Publish` 작업을 사용할 때 Amazon SNS용 AWS 관리형 KMS가 AWS KMS 생성됩니다.

- AWS 계정의 사용자 지정 KMS를 사용하려면 KMS 키 필드를 선택한 다음 목록에서 사용자 지정 KMS를 선택합니다.

  > **Note**
  >
  > 사용자 지정 KMS 생성에 대한 지침은 AWS Key Management Service 개발자 안내서의 키 생성을 참조하세요.

- 사용자 계정 또는 AWS 다른 계정의 사용자 지정 KMS AWS ARN을 사용하려면 KMS 키 필드에 해당 ARN을 입력합니다.

6. (선택 사항) 기본적으로 주제 소유자만 주제에 게시하거나 주제를 구독할 수 있습니다. 추가 액세스 권한을 구성하려면 액세스 정책 섹션을 확장합니다. 자세한 정보는 Amazon SNS의 Identity and Access Management 및 Amazon SNS 액세스 제어의 예제 사례에서 확인하세요.

> **Note**
>
> 콘솔을 사용하여 주제를 생성하는 경우 기본 정책은 `aws:SourceOwner` 조건 키를 사용합니다. 이 키는 `aws:SourceAccount`와 비슷합니다.

7. (선택 사항) Amazon SNS가 실패한 메시지 전송을 재시도하는 방식을 구성하려면 전송 재시도 정책(HTTP/S) 섹션을 확장합니다. 자세한 정보는 Amazon SNS 메시지 전송 재시도을 참조하세요.
8. (선택 사항) Amazon SNS가 메시지 전송을 로깅하는 방법을 구성하려면 CloudWatch 전송 상태 로깅 섹션을 확장하십시오. 자세한 정보는 Amazon SNS 메시지 전송 상태을 참조하세요.
9. (선택 사항) 메타데이터 태그를 주제에 추가하려면 태그 섹션을 확장하고, 키와 값(옵션)을 입력한 다음 태그 추가를 선택합니다. 자세한 정보는 Amazon SNS 주제 태그 지정을 참조하세요.
10. 주제 생성을 선택합니다.

    주제가 생성되고 ***MyTopic***페이지가 표시됩니다.

    주제 이름, ARN, 표시 이름 (선택 사항), 주제 소유자 AWS 계정 ID가 세부 정보 섹션에 표시됩니다.
11. 다음 예와 같이 주제 ARN을 클립보드에 복사합니다.

```
arn:aws:sns:us-east-2:123456789012:MyTopic
```

SDK를 AWS 사용하여 주제를 만들려면

AWS SDK를 사용하려면 사용자 인증 정보로 구성해야 합니다. 자세한 정보는 AWS SDK 및 도구 참조 가이드의 공유 구성 및 자격 증명 파일을 참조하세요.

다음 코드 예제는 사용 `CreateTopic` 방법을 보여줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 내용이 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

특정 이름으로 주제를 생성합니다.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example shows how to use Amazon Simple Notification Service
    /// (Amazon SNS) to add a new Amazon SNS topic.
    /// </summary>
    public class CreateSNSTopic
    {
        public static async Task Main()
        {
            string topicName = "ExampleSNSTopic";

            IAmazonSimpleNotificationService client = new
 AmazonSimpleNotificationServiceClient();

            var topicArn = await CreateSNSTopicAsync(client, topicName);
            Console.WriteLine($"New topic ARN: {topicArn}");
        }

        /// <summary>
        /// Creates a new SNS topic using the supplied topic name.
        /// </summary>
        /// <param name="client">The initialized SNS client object used to
        /// create the new topic.</param>
        /// <param name="topicName">A string representing the topic name.</param>
        /// <returns>The Amazon Resource Name (ARN) of the created topic.</
returns>
        public static async Task<string>
 CreateSNSTopicAsync(IAmazonSimpleNotificationService client, string topicName)
        {
            var request = new CreateTopicRequest
            {
                Name = topicName,
            };

            var response = await client.CreateTopicAsync(request);

            return response.TopicArn;
        }
```

```
    }
```

이름과 특정 FIFO 및 중복 제거 속성을 사용하여 새 주제를 생성합니다.

```
    /// <summary>
    /// Create a new topic with a name and specific FIFO and de-duplication
 attributes.
    /// </summary>
    /// <param name="topicName">The name for the topic.</param>
    /// <param name="useFifoTopic">True to use a FIFO topic.</param>
    /// <param name="useContentBasedDeduplication">True to use content-based de-
duplication.</param>
    /// <returns>The ARN of the new topic.</returns>
    public async Task<string> CreateTopicWithName(string topicName, bool
 useFifoTopic, bool useContentBasedDeduplication)
    {
        var createTopicRequest = new CreateTopicRequest()
        {
            Name = topicName,
        };

        if (useFifoTopic)
        {
            // Update the name if it is not correct for a FIFO topic.
            if (!topicName.EndsWith(".fifo"))
            {
                createTopicRequest.Name = topicName + ".fifo";
            }

            // Add the attributes from the method parameters.
            createTopicRequest.Attributes = new Dictionary<string, string>
            {
                { "FifoTopic", "true" }
            };
            if (useContentBasedDeduplication)
            {
                createTopicRequest.Attributes.Add("ContentBasedDeduplication",
 "true");
            }
        }
```

```
        var createResponse = await
 _amazonSNSClient.CreateTopicAsync(createTopicRequest);
        return createResponse.TopicArn;
    }
```

- API 세부 정보는 AWS SDK for .NET API CreateTopic참조를 참조하십시오.

C++

SDK for C++

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
//! Create an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param topicName: An Amazon SNS topic name.
  \param topicARNResult: String to return the Amazon Resource Name (ARN) for the
 topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::createTopic(const Aws::String &topicName,
                              Aws::String &topicARNResult,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::CreateTopicRequest request;
    request.SetName(topicName);

    const Aws::SNS::Model::CreateTopicOutcome outcome =
 snsClient.CreateTopic(request);

    if (outcome.IsSuccess()) {
        topicARNResult = outcome.GetResult().GetTopicArn();
        std::cout << "Successfully created an Amazon SNS topic " << topicName
                  << " with topic ARN '" << topicARNResult
```

```
                  << "'." << std::endl;

    }
    else {
        std::cerr << "Error creating topic " << topicName << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
        topicARNResult.clear();
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API CreateTopic참조를 참조하십시오.

CLI

AWS CLI

SNS 주제를 생성하려면

다음 `create-topic` 예제에서는 `my-topic`이라는 SNS 주제를 생성합니다.

```
aws sns create-topic \
    --name my-topic
```

출력:

```
{
    "ResponseMetadata": {
        "RequestId": "1469e8d7-1642-564e-b85d-a19b4b341f83"
    },
    "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
}
```

자세한 내용은 AWS 명령줄 인터페이스 사용 설명서의 Amazon SQS 및 Amazon SNS에서 AWS 명령줄 인터페이스 사용을 참조하십시오.

- API 세부 정보는 AWS CLI 명령 CreateTopic참조를 참조하십시오.

Go

SDK for Go V2

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// 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(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(context.TODO(), &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 CreateTopic참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicName>

                Where:
```

```
                    topicName - The name of the topic to create (for example,
 mytopic).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                    .name(topicName)
                    .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API CreateTopic참조를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

별도의 모듈에서 클라이언트를 생성하고 내보냅니다.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
import { CreateTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicName - The name of the topic to create.
 */
export const createTopic = async (topicName = "TOPIC_NAME") => {
  const response = await snsClient.send(
    new CreateTopicCommand({ Name: topicName }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '087b8ad2-4593-50c4-a496-d7e90b82cf3e',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
```

```
  //   TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:TOPIC_NAME'
  // }
  return response;
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API CreateTopic참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createSNSTopic(topicName: String): String {

    val request = CreateTopicRequest {
        name = topicName
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn.toString()
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 CreateTopic.

PHP

SDK for PHP

> **Note**
>
> 자세한 내용은 여기에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Create a Simple Notification Service topics in your AWS account at the
 requested region.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topicname = 'myTopic';

try {
    $result = $SnSclient->createTopic([
        'Name' => $topicname,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 자세한 정보는 AWS SDK for PHP 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for PHP API CreateTopic참조를 참조하십시오.

Python

SDK for Python(Boto3)

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def create_topic(self, name):
        """
        Creates a notification topic.

        :param name: The name of the topic to create.
        :return: The newly created topic.
        """
        try:
            topic = self.sns_resource.create_topic(Name=name)
            logger.info("Created topic %s with ARN %s.", name, topic.arn)
        except ClientError:
            logger.exception("Couldn't create topic %s.", name)
            raise
        else:
            return topic
```

- API에 대한 자세한 내용은 파이썬용AWS SDK (Boto3) API 레퍼런스를 참조하십시오 CreateTopic.

Ruby

SDK for Ruby

> **Note**
>
> 자세한 내용은 여기에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# This class demonstrates how to create an Amazon Simple Notification Service
 (SNS) topic.
class SNSTopicCreator
  # Initializes an SNS client.
  #
  # Utilizes the default AWS configuration for region and credentials.
  def initialize
    @sns_client = Aws::SNS::Client.new
  end

  # Attempts to create an SNS topic with the specified name.
  #
  # @param topic_name [String] The name of the SNS topic to create.
  # @return [Boolean] true if the topic was successfully created, false
 otherwise.
  def create_topic(topic_name)
    @sns_client.create_topic(name: topic_name)
    puts "The topic '#{topic_name}' was successfully created."
    true
  rescue Aws::SNS::Errors::ServiceError => e
    # Handles SNS service errors gracefully.
    puts "Error while creating the topic named '#{topic_name}': #{e.message}"
    false
  end
end
```

```
# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_name = "YourTopicName" # Replace with your topic name
  sns_topic_creator = SNSTopicCreator.new

  puts "Creating the topic '#{topic_name}'..."
  unless sns_topic_creator.create_topic(topic_name)
    puts "The topic was not created. Stopping program."
    exit 1
  end
end
```

- 자세한 정보는 AWS SDK for Ruby 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for Ruby API CreateTopic참조를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
async fn make_topic(client: &Client, topic_name: &str) -> Result<(), Error> {
    let resp = client.create_topic().name(topic_name).send().await?;

    println!(
        "Created topic with ARN: {}",
        resp.topic_arn().unwrap_or_default()
    );

    Ok(())
}
```

- API에 대한 자세한 내용은 Rust용AWS SDK API 레퍼런스를 참조하십시오 CreateTopic.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 자세한 내용은 여기에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   TRY.
       oo_result = lo_sns->createtopic( iv_name = iv_topic_name ). " oo_result
is returned for testing purposes. "
       MESSAGE 'SNS topic created' TYPE 'I'.
     CATCH /aws1/cx_snstopiclimitexcdex.
       MESSAGE 'Unable to create more topics. You have reached the maximum
number of topics allowed.' TYPE 'E'.
   ENDTRY.
```

- API 세부 정보는 SAP용AWS SDK ABAP API 참조를 참조하십시오 CreateTopic.

Amazon SNS 주제 구독

주제에 게시된 메시지를 수신하려면 엔드포인트에서 해당 주제를 구독해야 합니다. 엔드포인트에서 주제를 구독하면 엔드포인트는 연결된 주제에 게시된 메시지를 수신하기 시작합니다.

> **Note**
>
> HTTP(S) 엔드포인트, 이메일 주소 및 기타 AWS 계정의 AWS 리소스는 메시지를 수신하기 전에 구독을 확인해야 합니다.

엔드포인트에서 Amazon SNS 주제를 구독하려면

1. Amazon SNS 콘솔에 로그인합니다.
2. 왼쪽의 탐색 창에서 구독을 선택합니다.
3. 구독 페이지에서 구독 생성을 선택합니다.

4. 구독 생성 페이지의 세부 정보 섹션에서 다음을 수행합니다.

   a. 주제 ARN에서 주제의 Amazon 리소스 이름(ARN)을 선택합니다. 이 값은 Amazon SNS 주제를 만들 때 생성된 AWS ARN(예: `arn:aws:sns:us-east-2:123456789012:your_topic`)입니다.

   b. 프로토콜에서 엔드포인트 유형을 선택합니다. 사용 가능한 엔드포인트 유형은 다음과 같습니다.

      - HTTP/HTTPS
      - Email/Email-JSON
      - 아마존 데이터 파이어호스
      - Amazon SQS

      > **Note**
      > SNS FIFO 주제를 구독하려면 이 옵션을 선택합니다.

      - AWS Lambda
      - 플랫폼 애플리케이션 엔드포인트
      - SMS

   c. 엔드포인트에 이메일 주소 또는 Amazon SQS 대기열의 ARN과 같은 엔드포인트 값을 입력합니다.

   d. Firehose 엔드포인트만 해당: 구독 역할 ARN의 경우 Firehose 전송 스트림에 쓰기 위해 만든 IAM 역할의 ARN을 지정하십시오. 자세한 설명은 Amazon SNS 주제에 대한 Firehose 전송 스트림을 구독하기 위한 사전 요구 사항 섹션을 참조하세요.

   e. (선택 사항) Firehose, Amazon SQS, HTTP/S 엔드포인트의 경우 원시 메시지 전송을 활성화할 수도 있습니다. 자세한 설명은 Amazon SNS 원시 메시지 전송 섹션을 참조하세요.

   f. (선택 사항) 필터 정책을 구성하려면 구독 필터 정책 섹션을 확장합니다. 자세한 설명은 Amazon SNS 구독 필터 정책 섹션을 참조하세요.

   g. (선택 사항) 페이로드 기반 필터링을 활성화하려면 `Filter Policy Scope`를 `MessageBody`로 구성하십시오. 자세한 설명은 Amazon SNS 구독 필터 정책 범위 섹션을 참조하세요.

   h. (선택 사항) 구독에 대한 배달 못한 편지 대기열을 구성하려면 리드라이브 정책(배달 못한 편지 대기열) 섹션을 확장합니다. 자세한 설명은 Amazon SNS 배달 못한 편지 대기열(DLQ) 섹션을 참조하세요.

i. 구독 생성을 선택합니다.

콘솔에서 구독을 만들고 구독의 세부 정보 페이지를 엽니다.

Amazon SNS 주제 및 구독 삭제

주제가 삭제되면 관련 구독도 비동기적으로 삭제됩니다. 고객은 여전히 이러한 구독에 액세스할 수 있지만 동일한 이름을 사용하여 주제를 다시 만들더라도 구독은 더 이상 주제와 연결되지 않습니다.

구독자가 삭제된 주제에 메시지를 게시하려고 하면 게시자는 해당 주제가 존재하지 않는다는 오류 메시지를 받게 됩니다. 마찬가지로 삭제된 주제를 구독하려고 하면 오류 메시지가 표시됩니다.

확인이 보류 중인 구독은 삭제할 수 없습니다. 48시간 후 Amazon SNS는 확인되지 않은 구독을 자동으로 삭제합니다.

주제

- 를 사용하여 Amazon SNS 주제 또는 구독을 삭제하려면 AWS Management Console
- AWS SDK를 사용하여 구독 및 주제를 삭제하려면

를 사용하여 Amazon SNS 주제 또는 구독을 삭제하려면 AWS Management Console

를 사용하여 주제를 삭제하려면 AWS Management Console

1. Amazon SNS 콘솔에 로그인합니다.
2. 왼쪽의 탐색 창에서 주제를 선택합니다.
3. 주제 페이지에서 주제를 선택한 다음, 삭제를 선택합니다.
4. 삭제 대화 상자에 `delete me`를 입력한 후 삭제를 선택합니다.

   콘솔에서 주제가 삭제됩니다.

를 사용하여 구독을 삭제하려면 AWS Management Console

1. Amazon SNS 콘솔에 로그인합니다.
2. 왼쪽의 탐색 창에서 구독을 선택합니다.
3. 구독 페이지에서 상태가 확인됨인 구독을 선택한 다음 삭제를 선택합니다.

4. 구독 삭제 대화 상자에서 삭제를 선택합니다.

   콘솔에서 구독이 삭제됩니다.

AWS SDK를 사용하여 구독 및 주제를 삭제하려면

AWS SDK를 사용하려면 사용자 인증 정보로 구성해야 합니다. 자세한 정보는 AWS SDK 및 도구 참조 가이드의 공유 구성 및 자격 증명 파일을 참조하세요.

다음 코드 예제는 `DeleteTopic`의 사용 방법을 보여 줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> 자세한 내용은 에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

주제 ARN으로 주제를 삭제합니다.

```
/// <summary>
/// Delete a topic by its topic ARN.
/// </summary>
/// <param name="topicArn">The ARN of the topic.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteTopicByArn(string topicArn)
{
    var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
        new DeleteTopicRequest()
        {
            TopicArn = topicArn
        });
    return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- API 세부 정보는 AWS SDK for .NET API DeleteTopic참조를 참조하십시오.

C++

SDK for C++

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
//! Delete an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::deleteTopic(const Aws::String &topicARN,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::DeleteTopicRequest request;
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::DeleteTopicOutcome outcome =
 snsClient.DeleteTopic(request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted the Amazon SNS topic " << topicARN <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting topic " << topicARN << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API DeleteTopic참조를 참조하십시오.

CLI

AWS CLI

SNS 주제를 삭제하려면

다음 `delete-topic` 예제에서는 지정된 SNS 주제를 삭제합니다.

```
aws sns delete-topic \
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic"
```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 DeleteTopic참조를 참조하십시오.

Go

SDK for Go V2

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// 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(topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(context.TODO(), &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 DeleteTopic참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:     <topicArn>

                Where:
                   topicArn - The ARN of the topic to delete.
                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        System.out.println("Deleting a topic with name: " + topicArn);
        deleteSNSTopic(snsClient, topicArn);
        snsClient.close();
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                    .topicArn(topicArn)
                    .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API DeleteTopic참조를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

별도의 모듈에서 클라이언트를 생성하고 내보냅니다.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
import { DeleteTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic to delete.
 */
export const deleteTopic = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new DeleteTopicCommand({ TopicArn: topicArn }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'a10e2886-5a8f-5114-af36-75bd39498332',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
```

```
  // }
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API DeleteTopic참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteSNSTopic(topicArnVal: String) {

    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was successfully deleted.")
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 DeleteTopic.

PHP

SDK for PHP

> **Note**
>
> 자세한 내용은 여기에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Deletes an SNS topic and all its subscriptions.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->deleteTopic([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- API 세부 정보는 AWS SDK for PHP API DeleteTopic참조를 참조하십시오.

Python

SDK for Python(Boto3)

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def delete_topic(topic):
        """
        Deletes a topic. All subscriptions to the topic are also deleted.
        """
        try:
            topic.delete()
            logger.info("Deleted topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't delete topic %s.", topic.arn)
            raise
```

- API에 대한 자세한 내용은 파이썬용AWS SDK (Boto3) API 레퍼런스를 참조하십시오 DeleteTopic.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 자세한 내용은 여기에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
TRY.
    lo_sns->deletetopic( iv_topicarn = iv_topic_arn ).
    MESSAGE 'SNS topic deleted.' TYPE 'I'.
  CATCH /aws1/cx_snsnotfoundexception.
    MESSAGE 'Topic does not exist.' TYPE 'E'.
ENDTRY.
```

- API 세부 정보는 SAP용AWS SDK ABAP API 참조를 참조하십시오 DeleteTopic.

Amazon SNS 주제 태그 지정

Amazon SNS는 Amazon SNS 주제에 대한 태그 지정을 지원합니다. 이를 통해 주제와 관련된 비용을 추적 및 관리하고, AWS Identity and Access Management(IAM) 정책에서 향상된 보안을 제공하고, 수천 개의 주제를 쉽게 검색하거나 필터링할 수 있습니다. 태그 지정을 사용하면 AWS Resource Groups를 통해 Amazon SNS 주제를 관리할 수 있습니다. Resource Groups에 대한 자세한 내용은 AWS Resource Groups 사용 설명서를 참조하세요.

주제

- 비용 할당을 위한 태그 지정
- 액세스 제어용 태그 지정
- 리소스 검색 및 필터링을 위한 태그 지정
- Amazon SNS 주제 태그 구성

비용 할당을 위한 태그 지정

비용 할당을 위해 Amazon SNS 주제를 구성하고 식별하려면 주제의 목적을 식별하는 태그를 추가할 수 있습니다. 이 기능은 주제가 많을 때 특히 유용합니다. 비용 할당 태그를 사용하여 비용 구조를 반영하도록 AWS 청구서를 구성할 수 있습니다. 이렇게 하려면 태그 키와 값이 포함된 AWS 계정 청구서를 가져오도록 등록합니다. 자세한 내용은 AWS Billing and Cost Management 사용 설명서에서 월별 비용 할당 보고서 설정을 참조하세요.

예를 들어 다음과 같이 Amazon SNS 주제의 비용 센터 및 목적을 나타내는 태그를 추가할 수 있습니다.

리소스	키	값
주제 1	비용 센터	43289
	애플리케이션	주문 처리
주제 2	비용 센터	43289
	애플리케이션	결제 처리
주제 3	비용 센터	76585
	애플리케이션	보관

이 태그 지정 체계에서는 동일한 비용 센터에서 관련된 작업을 수행하는 2개의 주제를 그룹화할 수 있고, 관련이 없는 활동은 다른 비용 할당 태그를 사용해 태그 지정할 수 있습니다.

액세스 제어용 태그 지정

AWS Identity and Access Management는 태그를 기반으로 리소스에 대한 액세스를 제어하는 작업을 지원합니다. 리소스에 태그를 지정한 후 태그 기반 액세스를 관리하기 위해 IAM 정책의 조건 요소에 리소스 태그에 대한 정보를 제공합니다. Amazon SNS 콘솔 또는 AWS SDK를 사용하여 리소스에 태그를 지정하는 방법에 대한 자세한 내용은 태그 구성을 참조하세요.

IAM 자격 증명에 대한 액세스를 제한할 수 있습니다. 예를 들어 키가 `environment`이고 값이 `production`인 태그를 포함하는 모든 Amazon SNS 주제에 대한 `Publish` 및 `PublishBatch` 액세스는 제한하면서 다른 모든 Amazon SNS 주제에 대한 액세스는 허용할 수 있습니다. 아래 예에서 정책은 `production`으로 태그가 지정된 주제에 메시지를 게시하는 기능을 제한하면서 `development`로

태그가 지정된 주제에 메시지를 게시할 수 있도록 허용합니다. 자세한 내용은 IAM 사용 설명서의 태그를 사용한 액세스 제어를 참조하세요.

> **Note**
>
> Publish에 대한 IAM 권한을 설정하면 Publish 및 PublishBatch 모두에 대한 권한이 설정됩니다.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Deny",
    "Action": [
 "sns:Publish"
    ],
    "Resource": "arn:aws:sns:*:*:*",
    "Condition": {
      "StringEquals": {
        "aws:ResourceTag/environment": "production"
      }
    }
  },
  {
    "Effect": "Allow",
    "Action": [
      "sns:Publish"
    ],
    "Resource": "arn:aws:sns:*:*:*",
    "Condition": {
      "StringEquals": {
        "aws:ResourceTag/environment": "development"
      }
    }
  }]
}
```

리소스 검색 및 필터링을 위한 태그 지정

AWS 계정에는 수만 개의 Amazon SNS 주제가 있을 수 있습니다(자세한 내용은 Amazon SNS 할당량 참조). 주제에 태그를 지정하여 주제를 검색하거나 필터링하는 프로세스를 간소화할 수 있습니다.

예를 들어 프로덕션 환경과 관련된 수백 개의 주제가 있을 수 있습니다. 이러한 주제를 개별적으로 검색할 필요 없이 지정된 태그가 있는 모든 주제를 쿼리할 수 있습니다.

```
import com.amazonaws.services.resourcegroups.AWSResourceGroups;
import com.amazonaws.services.resourcegroups.AWSResourceGroupsClientBuilder;
import com.amazonaws.services.resourcegroups.model.QueryType;
import com.amazonaws.services.resourcegroups.model.ResourceQuery;
import com.amazonaws.services.resourcegroups.model.SearchResourcesRequest;
import com.amazonaws.services.resourcegroups.model.SearchResourcesResult;

public class Example {
    public static void main(String[] args) {
        // Query Amazon SNS Topics with tag "keyA" as "valueA"
        final String QUERY = "{\"ResourceTypeFilters\":[\"AWS::SNS::Topic\"],
\"TagFilters\":[{\"Key\":\"keyA\", \"Values\":[\"valueA\"]}]}";

        // Initialize ResourceGroup client
        AWSResourceGroups awsResourceGroups = AWSResourceGroupsClientBuilder
            .standard()
            .build();

        // Query all resources with certain tags from ResourceGroups
        SearchResourcesResult result = awsResourceGroups.searchResources(
            new SearchResourcesRequest().withResourceQuery(
                new ResourceQuery()
                .withType(QueryType.TAG_FILTERS_1_0)
                .withQuery(QUERY)
            ));
        System.out.println("SNS Topics with certain tags are " +
 result.getResourceIdentifiers());
    }
}
```

Amazon SNS 주제 태그 구성

이 페이지에서는 AWS Management Console, AWS SDK 및 AWS CLI를 사용하여 Amazon SNS 주제에 대한 태그를 구성하는 방법을 보여줍니다.

> **Important**
>
> 개인 식별 정보(PII)나 기타 기밀 정보 또는 민감한 정보를 태그에 추가하지 않습니다. 태그는 결제를 포함하여 다른 Amazon Web Services에서 액세스할 수 있습니다. 태그는 개인 데이터나 민감한 데이터에 사용하기 위한 것이 아닙니다.

주제

- 를 사용하여 Amazon SNS 주제에 대한 태그를 나열, 추가 및 제거합니다. AWS Management Console
- AWS SDK를 사용하여 주제에 태그 추가
- Amazon SNS API 작업을 사용하여 태그 관리
- ABAC를 지원하는 API 작업

를 사용하여 Amazon SNS 주제에 대한 태그를 나열, 추가 및 제거합니다. AWS Management Console

1. Amazon SNS 콘솔에 로그인합니다.
2. 탐색 창에서 주제(Topics)를 선택합니다.
3. 주제 페이지에서 주제를 선택한 다음 편집을 선택합니다.
4. 태그 섹션을 확장합니다.

   주제에 추가된 태그가 나열됩니다.
5. 주제 태그를 수정합니다.
   - 태그를 추가하려면 태그 추가(Add tag)를 선택하고 키(Key) 및 값(Value)을 입력합니다(선택 사항).
   - 태그를 제거하려면 키-값 페어 옆에 있는 태그 제거를 선택합니다.
6. 변경 사항 저장를 선택합니다.

AWS SDK를 사용하여 주제에 태그 추가

AWS SDK를 사용하려면 사용자 인증 정보로 구성해야 합니다. 자세한 정보는 AWS SDK 및 도구 참조 가이드의 공유 구성 및 자격 증명 파일을 참조하세요.

다음 코드 예제는 사용 TagResource 방법을 보여줍니다.

CLI

AWS CLI

주제에 태그를 추가하려면

다음 tag-resource 예제에서는 지정된 Amazon SNS 주제에 메타데이터 태그를 추가합니다.

```
aws sns tag-resource \
    --resource-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --tags Key=Team,Value=Alpha
```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 TagResource참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.Tag;
import software.amazon.awssdk.services.sns.model.TagResourceRequest;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class AddTags {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn>

                Where:
                   topicArn - The ARN of the topic to which tags are added.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        addTopicTags(snsClient, topicArn);
        snsClient.close();
    }

    public static void addTopicTags(SnsClient snsClient, String topicArn) {
        try {
            Tag tag = Tag.builder()
                    .key("Team")
                    .value("Development")
                    .build();

            Tag tag2 = Tag.builder()
                    .key("Environment")
                    .value("Gamma")
                    .build();

            List<Tag> tagList = new ArrayList<>();
            tagList.add(tag);
            tagList.add(tag2);
```

```
            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(topicArn)
                    .tags(tagList)
                    .build();

            snsClient.tagResource(tagResourceRequest);
            System.out.println("Tags have been added to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API TagResource참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun addTopicTags(topicArn: String) {

    val tag = Tag {
        key = "Team"
        value = "Development"
    }

    val tag2 = Tag {
        key = "Environment"
        value = "Gamma"
    }

    val tagList = mutableListOf<Tag>()
    tagList.add(tag)
```

```
    tagList.add(tag2)

    val request = TagResourceRequest {
        resourceArn = topicArn
        tags = tagList
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.tagResource(request)
        println("Tags have been added to $topicArn")
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 TagResource.

Amazon SNS API 작업을 사용하여 태그 관리

Amazon SNS API를 사용하여 태그를 관리하려면 다음 API 작업을 사용합니다.

- ListTagsForResource
- TagResource
- UntagResource

ABAC를 지원하는 API 작업

다음은 속성 기반 액세스 제어(ABAC)를 지원하는 API 작업의 목록입니다. ABAC에 대한 자세한 내용은 ABAC의 용도를 참조하세요. AWS IAM 사용 설명서에서

- AddPermission
- ConfirmSubscription
- DeleteTopic
- GetDataProtectionPolicy
- GetSubscriptionAttributes
- GetTopicAttributes
- ListSubscriptionsByTopic
- ListTagsForResource

- Publish
- PublishBatch
- PutDataProtectionPolicy
- RemovePermission
- SetSubscriptionAttributes
- SetTopicAttributes
- Subscribe
- TagResource
- Unsubscribe
- UntagResource

메시지 순서 지정 및 중복 제거(FIFO 주제)

Amazon SNS FIFO(선입 선출) 주제와 Amazon SQS FIFO 대기열을 함께 사용하여 엄격한 메시지 순서 지정 및 메시지 중복 제거 기능을 제공할 수 있습니다. 이러한 각 서비스의 FIFO 기능은 함께 작동하여 거의 실시간으로 데이터 일관성이 필요한 분산 애플리케이션을 통합하는 완전 관리형 서비스 역할을 합니다. Amazon SQS 표준 대기열에서 Amazon SNS FIFO 주제를 구독하면 최선의 순서 지정 및 최소 한 번 전달이 이루어집니다.

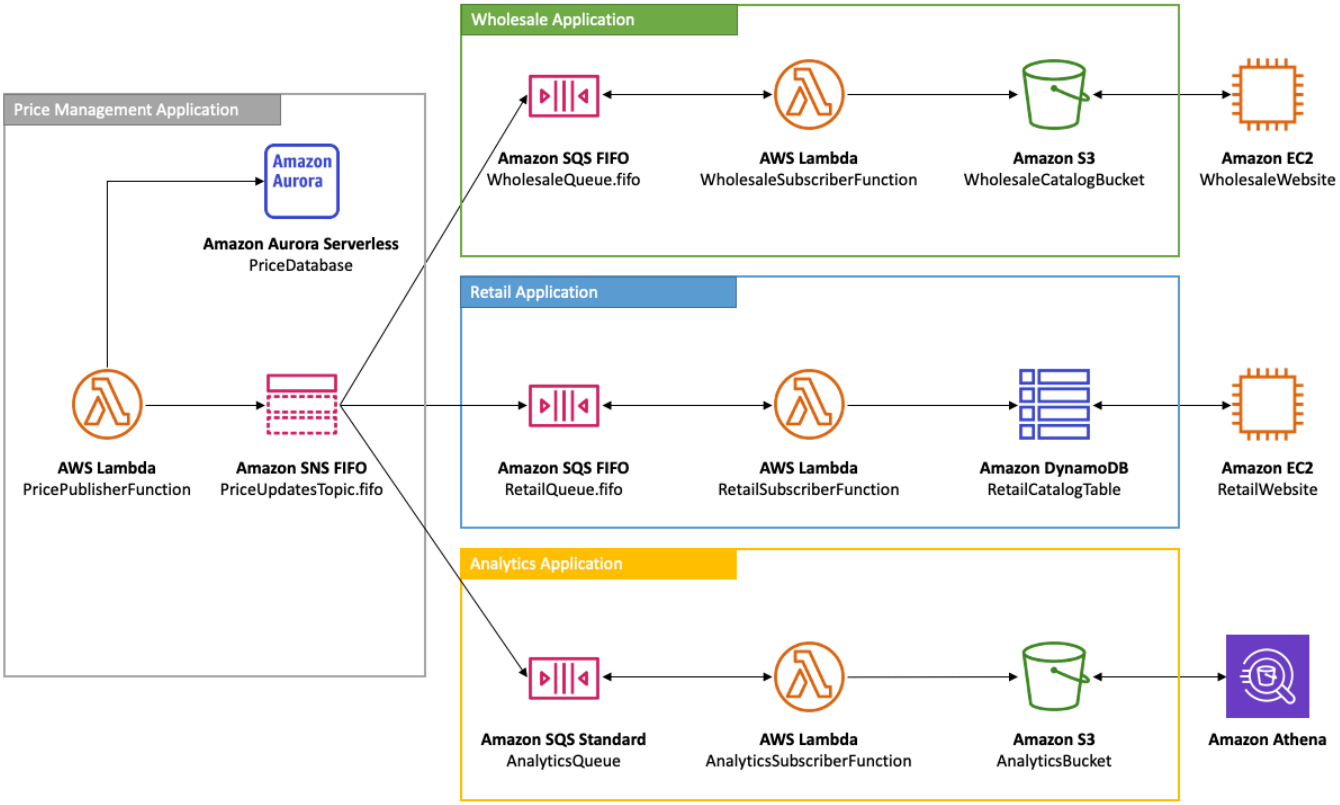
주제

FIFO 주제 예 사용 사례

다음 예에서는 Amazon SNS FIFO 주제 및 Amazon SQS 대기열을 사용하여 자동차 부품 제조업체에서 구축한 전자상거래 플랫폼을 설명합니다. 이 플랫폼은 다음 4가지의 서버리스 애플리케이션으로 구성됩니다.

- 인벤토리 관리자는 가격 관리 애플리케이션을 사용하여 재고가 있는 각 항목의 가격을 설정합니다. 이 회사에서는 환율 변동, 시장 수요, 판매 전략의 변화에 따라 제품 가격이 변동될 수 있습니다. 가격 관리 애플리케이션은 가격이 변경될 때마다 Amazon SNS FIFO 주제에 가격 업데이트를 게시하는 AWS Lambda 기능을 사용합니다.
- 도매 애플리케이션은 자동차 차체 상점과 자동차 제조업체가 회사의 자동차 부품을 대량으로 구입할 수 있는 웹사이트에 대한 백엔드를 제공합니다. 가격 변경 알림을 받기 위해 도매 애플리케이션은 해당 Amazon SQS FIFO 대기열에서 가격 관리 애플리케이션의 Amazon SNS FIFO 주제를 구독합니다.

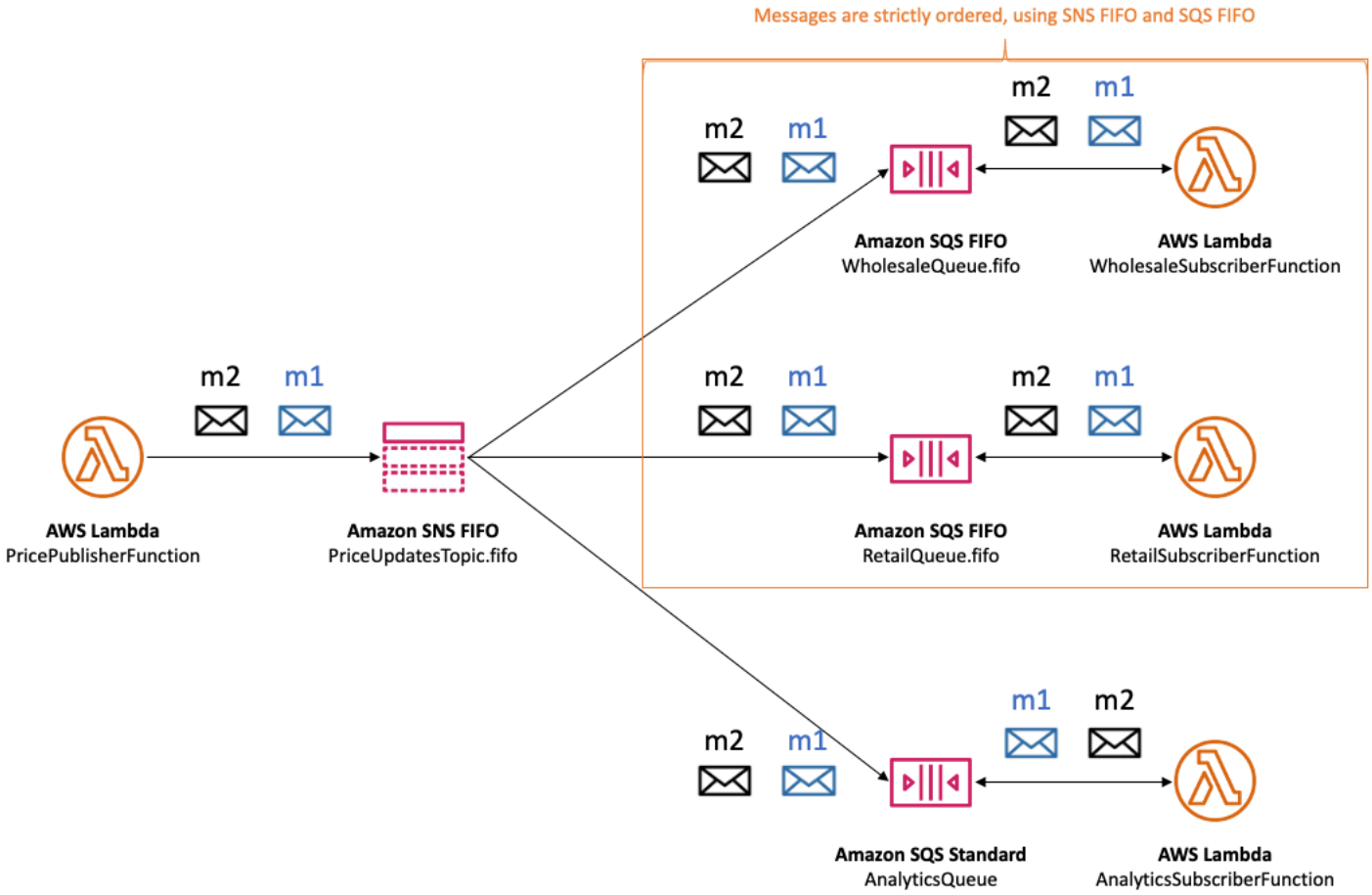
- 소매 애플리케이션은 자동차 소유자와 자동차 튜닝 애호가가 차량의 개별 자동차 부품을 구매할 수 있는 다른 웹사이트에 대한 백엔드를 제공합니다. 가격 변경 알림을 받기 위해 소매 애플리케이션도 해당 Amazon SQS FIFO 대기열에서 가격 관리 애플리케이션의 Amazon SNS FIFO 주제를 구독합니다.
- 가격 업데이트를 집계하여 Amazon S3 버킷에 저장하는 분석 애플리케이션을 통해 Amazon Athena에서 비즈니스 인텔리전스(BI) 목적으로 버킷을 쿼리할 수 있습니다. 가격 변경 알림을 받기 위해 분석 애플리케이션은 해당 Amazon SQS 표준 대기열에서 가격 관리 애플리케이션의 Amazon SNS FIFO 주제를 구독합니다. 다른 애플리케이션과 달리 분석 애플리케이션은 가격 업데이트의 순서를 엄격하게 지정할 필요가 없습니다.



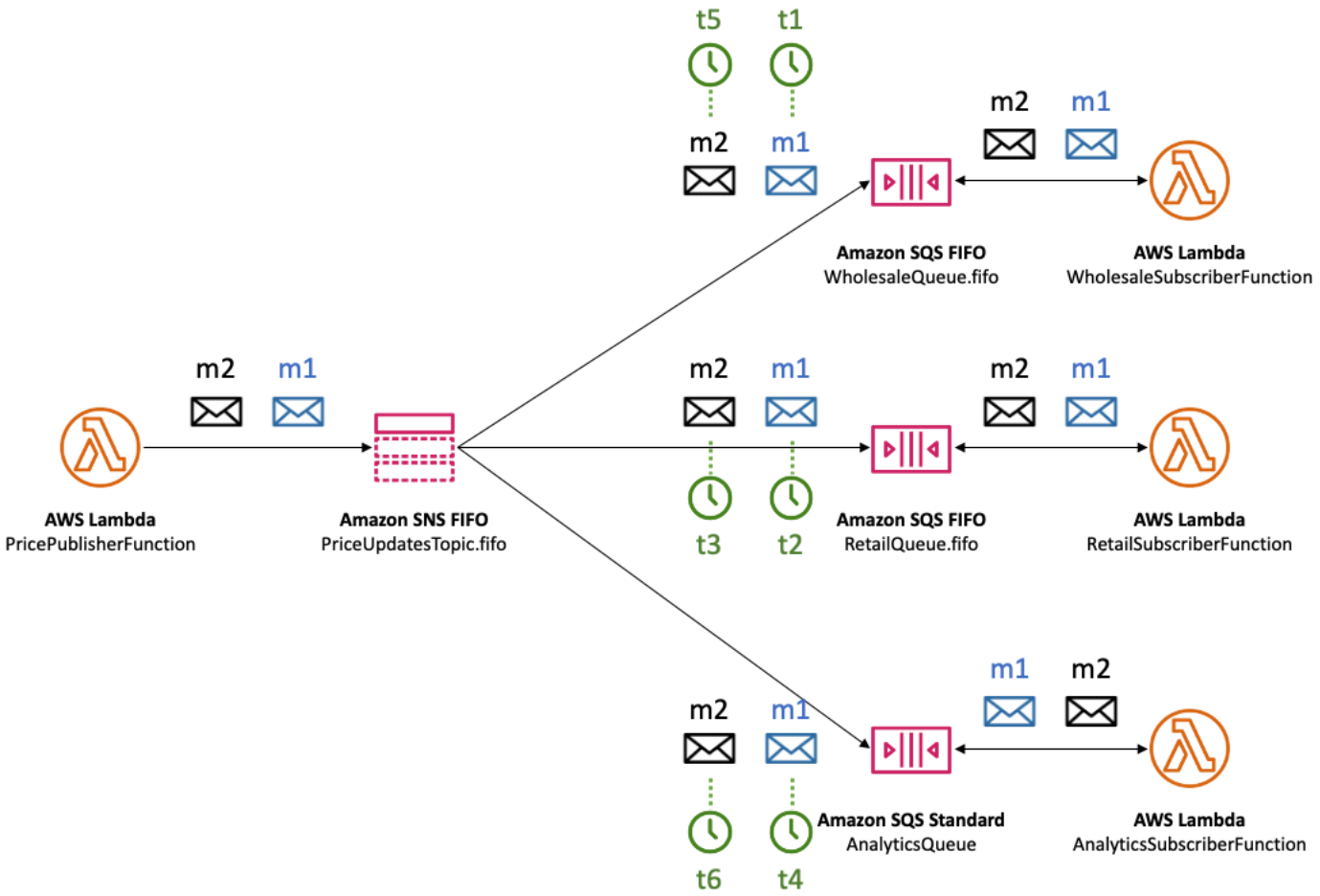
도매 및 소매 애플리케이션이 올바른 순서로 가격 업데이트를 수신하려면 가격 관리 애플리케이션이 엄격하게 순서가 지정된 메시지 배포 시스템을 사용해야 합니다. Amazon SNS FIFO 주제 및 Amazon SQS FIFO 대기열을 사용하면 중복 없이 순서대로 메시지를 처리할 수 있습니다. 자세한 내용은 FIFO 주제에 대한 메시지 정렬 세부 정보 섹션을 참조하세요. 이 사용 사례를 구현하는 코드 조각은 FIFO 주제에 대한 코드 예제에서 확인하세요.

FIFO 주제에 대한 메시지 정렬 세부 정보

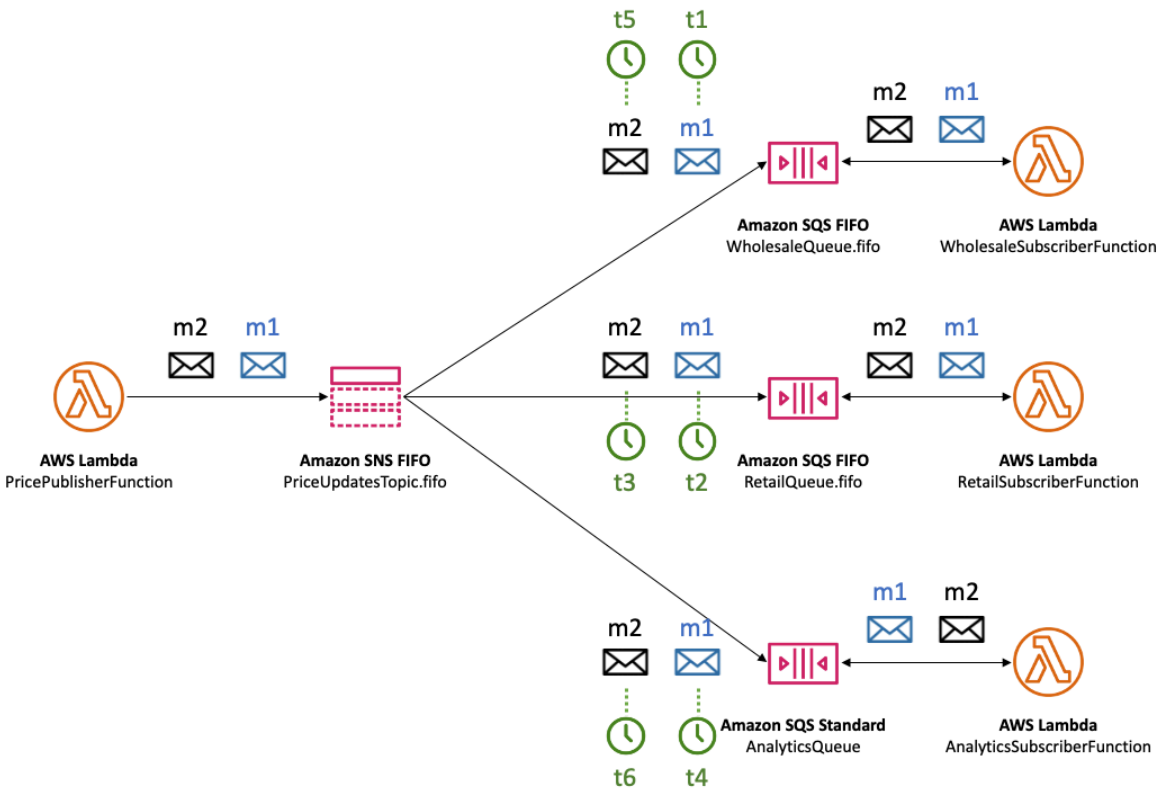
Amazon SNS FIFO 주제는 항상 메시지가 주제에 게시된 정확한 순서로 한 번만, 구독된 Amazon SQS FIFO 대기열에 메시지를 전달합니다. Amazon SQS FIFO 대기열을 구독하면 대기열 소비자가 메시지가 대기열로 전송되는 순서와 동일한 순서로 메시지를 수신하며 중복은 없습니다. 그러나 Amazon SQS 표준 대기열을 구독하면 대기열의 소비자가 메시지를 다른 순서로 두 번 이상 수신할 수 있습니다. 이를 통해 구독자와 게시자를 더욱 분리할 수 있어 구독자가 FIFO 주제 예 사용 사례를 기반으로 하는 다음 다이어그램에 나온 것처럼 메시지 소비 및 비용 최적화 측면에서 유연성을 누릴 수 있습니다.



구독자에 대한 묵시적인 정렬은 없습니다. 다음 예에서는 메시지 m1이 먼저 도매 구독자에게 전달된 다음 소매 구독자에게 전달되고 마지막으로 분석 구독자에게 전달됨을 보여줍니다. 메시지 m2는 소매 구독자에게 먼저 전달된 다음 도매 구독자에게 전달되고 마지막으로 분석 구독자에게 전달됩니다. 두 메시지가 구독자에게 다른 순서로 배달되지만 메시지 순서는 각 Amazon SQS FIFO 구독자에 대해 유지됩니다. 각 구독자는 다른 구독자와 분리되어 인식됩니다.

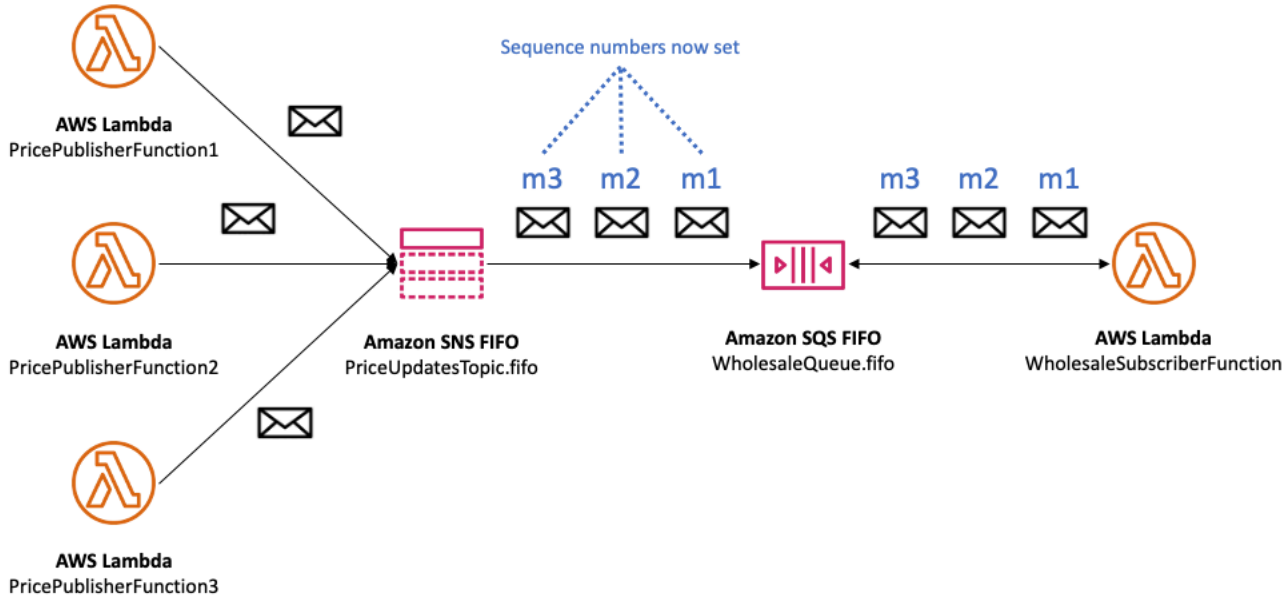


Amazon SQS 대기열 구독자에 연결할 수 없는 경우 동기화되지 않을 수 있습니다. 예를 들어, 도매 애플리케이션 대기열 소유자가 Amazon SNS 서비스 보안 주체가 대기열로 메시지를 전송하지 못하도록 하는 방식으로 Amazon SQS 대기열 정책을 실수로 변경했다고 가정해 보겠습니다. 이 경우 도매 대기열로의 가격 업데이트는 전달되지 않지만 소매 및 분석 대기열로의 전달은 성공하여 구독자가 동기화되지 않게 됩니다. 도매 애플리케이션 대기열 소유자가 대기열 정책을 수정하면 Amazon SNS는 구독 대기열로 메시지 전달을 재개합니다. 구독에 DLQ(Dead Letter Queue)가 구성되어 있지 않은 경우, 잘못 구성된 대기열을 대상으로 하는 주제에 게시된 모든 메시지는 삭제됩니다.



여러 애플리케이션(또는 동일한 애플리케이션 내의 여러 스레드)이 SNS FIFO 주제에 병렬로 메시지를 게시하도록 할 수 있습니다. 이렇게 하면 메시지 시퀀싱을 Amazon SNS 서비스에 효과적으로 위임하는 것입니다. 설정된 메시지 시퀀스를 확인하기 위해 시퀀스 번호를 확인할 수 있습니다.

시퀀스 번호는 Amazon SNS가 각 메시지에 할당하는 크고 비연속적인 숫자입니다. 시퀀스 번호의 길이는 128비트이며 각 메시지 그룹마다 계속 늘어납니다. 시퀀스 번호는 메시지 본문의 일부로 구독된 Amazon SQS 대기열에 전달됩니다. 그러나 원시 메시지 전달을 사용하면 Amazon SQS 대기열로 전달되는 메시지에 시퀀스 번호나 기타 Amazon SNS 메시지 메타데이터가 포함되지 않습니다.



Amazon SNS FIFO 주제는 메시지 그룹의 컨텍스트에서 정렬을 정의합니다. 자세한 내용은 FIFO 주제에 대한 메시지 그룹화 섹션을 참조하세요.

FIFO 주제에 대한 메시지 그룹화

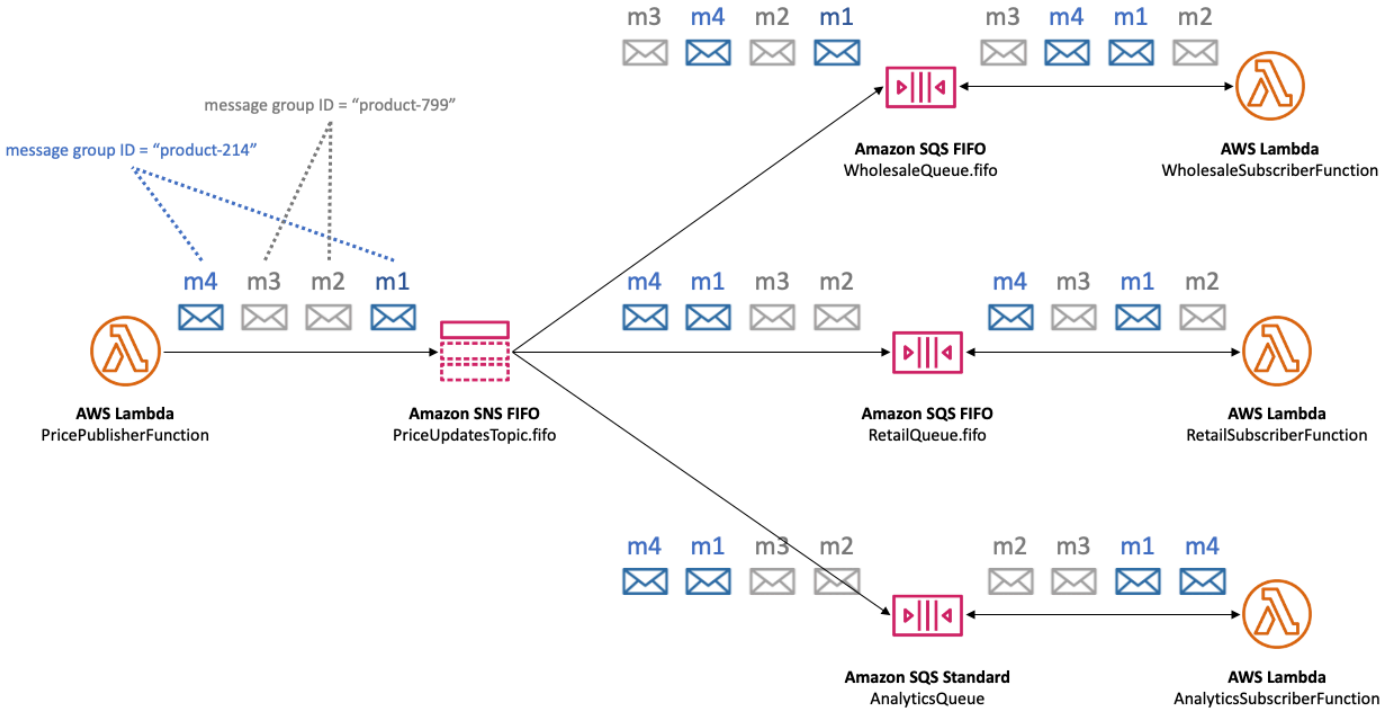
동일한 그룹에 속한 메시지는 그룹에 상대적인 엄격한 순서로 하나씩 처리됩니다.

Amazon SNS FIFO 주제에 메시지를 게시할 때 메시지 그룹 ID를 설정합니다. 그룹 ID는 메시지가 특정 메시지 그룹에 속하도록 지정하는 필수 토큰입니다. SNS FIFO 주제는 구독된 Amazon SQS FIFO 대기열에 그룹 ID를 전달합니다. SNS FIFO 주제 또는 SQS FIFO 대기열의 그룹 ID 수에는 제한이 없습니다. 메시지 그룹 ID는 Amazon SQS 표준 대기열로 전달되지 않습니다.

메시지 그룹과 구독 간에는 선호도가 없습니다. 따라서 모든 메시지 그룹에 게시된 메시지는 구독에 연결된 필터 정책에 따라 모든 구독 대기열에 전송됩니다. 자세한 정보는 FIFO 주제에 대한 메시지 전송 및 FIFO 주제에 대한 메시지 필터링에서 확인하세요.

자동차 부품 가격 관리 예 사용 사례에는 플랫폼에서 판매되는 각 제품에 대한 전용 메시지 그룹이 있습니다. 동일한 Amazon SNS FIFO 주제가 모든 가격 업데이트를 처리하는 데 사용됩니다. 가격 업데이트 순서는 단일 자동차 부품 제품 컨텍스트 내에서 유지되지만 복수의 제품에서는 유지되지 않습니다. 다음 다이어그램은 이 과정을 보여 줍니다. 메시지 그룹 ID가 product-214인 제품의 경우 m1 메시지는 항상 m4 메시지보다 먼저 처리됩니다. 이 시퀀스는 Amazon SNS FIFO에서 Amazon SQS FIFO까지 사용하는 워크플로 전체에서 보존됩니다. 마찬가지로 메시지 그룹 ID가 제품-799인 제품의 경우 워크플로에서 Amazon SNS FIFO와 Amazon SQS FIFO를 사용하는 한 m2 메시지가 m3 메시지보다 먼저 처리됩니다. 하지만 Amazon SQS 표준 대기열을 사용하면 메시지 순서가 더 이상 보장되지 않으

며 메시지 그룹이 존재하지 않습니다. product-214 및 product-799 메시지 그룹은 서로 독립적이므로 해당 메시지의 순서를 지정하는 방법 사이에는 관계가 없습니다.



성능 향상을 위한 메시지 그룹 ID별 데이터 배포

전송 처리량을 최적화하기 위해 Amazon SNS FIFO 주제는 서로 다른 메시지 그룹의 메시지를 병렬로 전송하지만 메시지 순서는 각 메시지 그룹 내에서 엄격하게 유지됩니다. 각 메시지 그룹은 초당 최대 300개의 메시지를 전송할 수 있습니다. 따라서 단일 주제에 대한 처리량을 높이려면 많은 수의 개별 메시지 그룹 ID를 사용하도록 합니다. Amazon SNS FIFO 주제는 다양한 메시지 그룹을 활용하여 더 많은 수의 병렬 파티션에 메시지를 자동으로 배포합니다.

> **Note**
>
> Amazon SNS FIFO 주제는 그룹 수에 관계없이 메시지 그룹 ID 전체에 메시지를 균일하게 배포하도록 최적화되어 있습니다. AWS에서는 성능 최적화를 위해 많은 수의 개별 메시지 그룹 ID를 사용할 것을 권장합니다.

처리량이 높고 Amazon SQS FIFO 대기열이 하나 이상 구독되어 있는 상태에서 Amazon SNS FIFO 주제에 게시하는 경우 대기열에 높은 처리량을 활성화하는 것이 좋습니다. 자세한 Amazon Simple Queue Service 개발자 안내서의 FIFO 대기열의 높은 처리량을 참조하세요.

FIFO 주제에 대한 메시지 전송

Amazon SNS FIFO(선입 선출) 주제는 Amazon SQS 표준 대기열과 FIFO 대기열 모두에 전송을 지원하므로 거의 실시간으로 데이터 일관성이 필요한 분산 애플리케이션을 통합할 때 유연하게 제어할 수 있습니다.

엄격한 메시지 순서 지정 또는 중복 제거를 유지해야 하는 워크로드의 경우, Amazon SNS FIFO 주제와 전송 엔드포인트로 구독된 Amazon SQS FIFO 대기열을 함께 사용하면 작업 및 이벤트 순서가 중요하거나 중복이 허용되지 않을 때 애플리케이션 간 메시징이 개선됩니다.

최선의 순서 지정 및 최소 한 번 전달을 허용하는 워크로드의 경우 Amazon SQS 표준 대기열에서 Amazon SNS FIFO 주제를 구독하면 FIFO를 활용하지 않는 워크로드 전체에서 대기열을 공유할 뿐만 아니라 비용을 절감할 수 있습니다.

> **Note**
>
> Amazon SNS FIFO 주제에서 AWS Lambda 함수로 메시지를 팬아웃하려면 추가 단계가 필요합니다. 먼저 Amazon SQS FIFO 또는 표준 대기열에서 주제를 구독합니다. 그런 다음 함수를 트리거하도록 대기열을 구성합니다. 자세한 정보는 AWS 컴퓨팅 블로그의 이벤트 소스인 SQS FIFO 게시글을 참조하세요.

SNS FIFO 주제는 이메일 주소, 모바일 앱, 문자 메시지용 전화번호(SMS) 또는 HTTP(S) 엔드포인트와 같은 고객 관리형 엔드포인트에 메시지를 전송할 수 없습니다. 이러한 엔드포인트 유형은 엄격한 메시지 순서를 유지하지 못할 수 있습니다. 고객 관리형 엔드포인트에서 SNS FIFO 주제를 구독하려고 하면 오류가 발생합니다.

SNS FIFO 주제는 표준 주제와 동일한 메시지 필터링 기능을 지원합니다. 자세한 정보는 FIFO 주제에 대한 메시지 필터링 및 AWS 컴퓨팅 블로그의 Amazon SNS 메시지 필터링으로 Pub/Sub 메시징 단순화 게시물을 참조하세요.

FIFO 주제에 대한 메시지 필터링

Amazon SNS FIFO 주제는 메시지 필터링을 지원합니다. 메시지 필터링을 사용하면 게시자 시스템의 메시지 라우팅 논리와 구독자 시스템의 메시지 필터링 논리를 오프로드하여 아키텍처가 단순화됩니다.

Amazon SQS FIFO 또는 표준 대기열에서 SNS FIFO 주제를 구독할 때 메시지 필터링을 사용하여 구독자가 모든 메시지가 아닌 일부 메시지를 수신하도록 지정할 수 있습니다. 각 구독자는 자체 필터 정책을 구독 속성으로 설정할 수 있습니다. 필터 정책 범위에 따라 필터 정책을 수신 메시지 속성 또는 메시지 본문과 일치시킵니다. 필터 정책이 일치하는 경우 주제는 구독자에게 메시지 복사본을 전달합니다. 일치하는 항목이 없으면 주제는 메시지 복사본을 전송하지 않습니다.

자동차 부품 가격 관리 예 사용 사례에서 다음 Amazon SNS 필터 정책이 설정되고 필터 정책 범위는 `MessageBody`라고 가정합니다.

- 도매 대기열의 경우 필터 정책 `{"business":["wholesale"]}`은 이름이 `business`인 키와 값 집합의 `wholesale`이 포함된 모든 메시지를 일치시킵니다. 다음 다이어그램에서 메시지 m1의 키 중 하나는 값이 `wholesale`인 `business`입니다. 메시지 m3의 키 중 하나는 값이 `["wholesale,retail"]`인 `business`입니다. 따라서 m1 및 m3은 모두 필터 정책의 기준과 일치하며 두 메시지 모두 도매 대기열로 전달됩니다.
- 소매 대기열의 경우 필터 정책 `{"business":["retail"]}`는 이름이 `business`인 키와 값 집합의 `retail`이 포함된 모든 메시지를 일치시킵니다. 다이어그램에서 메시지 m2의 키 중 하나는 값이 `retail`인 `business`입니다. 메시지 m3의 키 중 하나는 값이 `["wholesale,retail"]`인 `business`입니다. 따라서 m2 및 m3은 모두 필터 정책의 기준과 일치하며 두 메시지 모두 소매 대기열로 전달됩니다.
- 분석 대기열의 경우 Amazon Athena가 모든 레코드를 수신하도록 할 것이므로 필터 정책이 적용되지 않습니다.

SNS FIFO 주제는 속성 문자열 값, 속성 숫자 값 및 속성 키를 비롯한 다양한 일치 연산자를 지원합니다. 자세한 내용은 Amazon SNS 메시지 필터링 섹션을 참조하세요.

SNS FIFO 주제는 구독된 엔드포인트에 중복 메시지를 전달하지 않습니다. 자세한 내용은 FIFO 주제에 대한 메시지 중복 제거 섹션을 참조하세요.

FIFO 주제에 대한 메시지 중복 제거

Amazon SNS FIFO 주제 및 Amazon SQS FIFO 대기열은 다음 조건이 충족되는 한 정확히 한 번만 메시지 전송 및 처리를 제공하는 메시지 중복 제거 기능을 지원합니다.

- 구독한 Amazon SQS FIFO 대기열이 존재하며 이 대기열에는 Amazon SNS 서비스 보안 주체가 대기열에 메시지를 전달할 수 있는 권한이 있습니다.
- Amazon SQS FIFO 대기열 소비자는 가시성 시간 제한이 만료되기 전에 메시지를 처리하고 대기열에서 해당 메시지를 삭제합니다.
- Amazon SNS 구독 주제에는 메시지 필터링 기능이 없습니다. 메시지 필터링을 구성하면 구독 필터 정책에 따라 메시지를 필터링할 수 있으므로 Amazon SNS FIFO 주제가 at-most-once 전송을 지원합니다.
- 메시지 전송 확인을 방해하는 네트워크 중단은 없습니다.

> **Note**
>
> 메시지 중복 제거는 개별 메시지 그룹이 아닌 전체 Amazon SNS FIFO 주제에 적용됩니다.

메시지를 Amazon SNS FIFO 주제에 게시하면 해당 메시지에는 중복 제거 ID가 포함되어야 합니다. 이 ID는 Amazon SNS FIFO 주제가 구독된 Amazon SQS FIFO 대기열에 전송하는 메시지에 포함됩니다.

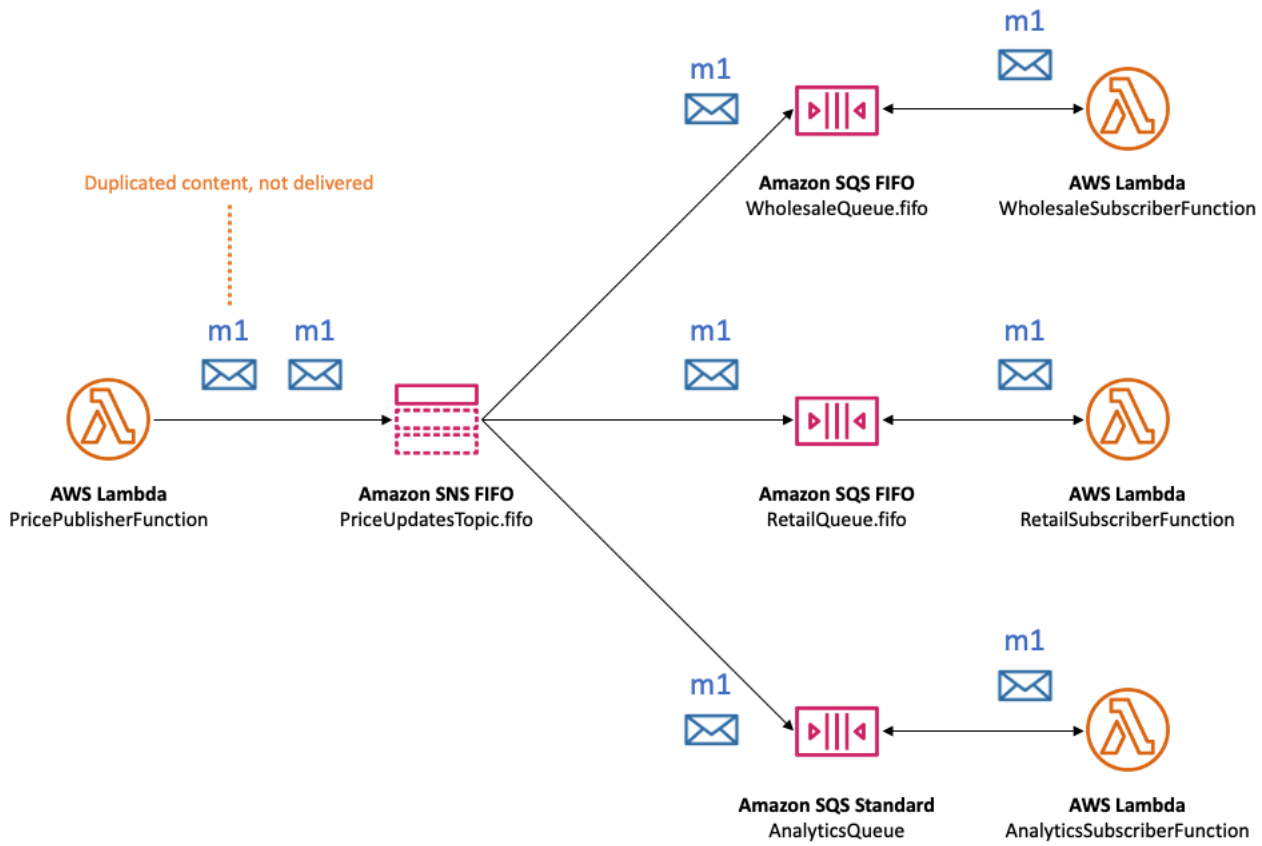
특정 중복 제거 ID가 있는 메시지가 Amazon SNS FIFO 주제에 성공적으로 게시된 경우 5분 중복 제거 간격 내에 동일한 중복 제거 ID로 게시된 모든 메시지는 수락되지만 전송되지는 않습니다. Amazon SNS FIFO 주제는 메시지가 구독된 엔드포인트에 전송된 후에도 메시지 중복 제거 ID를 계속 추적합니다.

메시지 본문이 게시된 각 메시지에 대해 고유하다고 보장되는 경우 Amazon SNS FIFO 주제 및 구독된 Amazon SQS FIFO 대기열에 대해 콘텐츠 기반 중복 제거를 사용할 수 있습니다. Amazon SNS는 메시지 본문을 사용하여 각 메시지의 중복 제거 ID로 사용할 고유한 해시 값을 생성하므로 각 메시지를 보낼 때 중복 제거 ID를 설정할 필요가 없습니다.

> **Note**
>
> 메시지 속성은 해시 계산에 포함되지 않습니다.

Amazon SNS FIFO 주제에 대해 콘텐츠 기반 중복 제거가 활성화되고 메시지가 중복 제거 ID와 함께 게시되면 게시된 중복 제거 ID가 생성된 콘텐츠 기반 중복 제거 ID보다 우선합니다.

자동차 부품 가격 관리 예 사용 사례에서 회사는 각 가격 업데이트에 대해 범용 고유 중복 제거 ID를 설정해야 합니다. 도매 및 소매의 메시지 속성이 다른 경우에도 메시지 본문이 동일할 수 있기 때문입니다. 그러나 회사가 제품 ID 및 제품 가격과 함께 메시지 본문에 비즈니스 유형(도매 또는 소매)을 추가했다면 Amazon SNS FIFO 주제 및 구독된 Amazon SQS FIFO 대기열에서 콘텐츠 기반 복제를 사용할 수 있습니다.



Amazon SNS FIFO 주제는 메시지 순서 지정 및 중복 제거 외에도 AWS KMS 키를 사용한 메시지 서버 측 암호화 (SSE) 와 VPC 엔드포인트를 통한 메시지 개인 정보 보호를 지원합니다. AWS PrivateLink자세한 내용은 FIFO 주제에 대한 메시지 보안을(를) 참조하세요.

FIFO 주제에 대한 메시지 보안

Amazon SNS 및 Amazon SQS가 AWS Key Management Service(AWS KMS) 고객 마스터 키(CMK)를 사용하여 FIFO 주제 및 대기열로 전송된 메시지를 암호화하도록 선택할 수 있습니다. 암호화된 FIFO 주제 및 대기열을 생성하거나 기존 FIFO 주제 및 대기열을 암호화하도록 선택할 수 있습니다. Amazon SNS 및 Amazon SQS는 메시지 본문만 암호화합니다. 메시지 속성, 리소스 메타데이터 또는 리소스 지표를 암호화하지 않습니다.

> **Note**
>
> 기존 FIFO 주제 또는 대기열에 암호화를 추가해도 백로그된 메시지는 암호화되지 않으며, 주제 또는 대기열에서 암호화를 제거하면 백로그된 메시지가 암호화된 상태로 남습니다.

SNS FIFO 주제는 구독된 엔드포인트로 메시지를 전달하기 직전에 메시지를 복호화합니다. SQS FIFO 대기열은 메시지를 소비자 애플리케이션에 반환하기 직전에 암호를 복호화합니다. 자세한 정보는 데이터 암호화 및 AWS 컴퓨팅 블로그의 AWS KMS로 Amazon SNS에 게시된 메시지 암호화 게시물을 참조하세요.

또한 SNS FIFO 주제 및 SQS FIFO 대기열은 AWS PrivateLink에서 제공하는 인터페이스 VPC 엔드포인트로 메시지 개인 정보 보호를 지원합니다. 인터페이스 엔드포인트를 사용하면 공용 인터넷을 통과하지 않고도 Amazon Virtual Private Cloud(Amazon VPC) 서브넷에서 FIFO 주제 및 대기열로 메시지를 보낼 수 있습니다. 이 모델은 AWS 인프라 및 네트워크 내에서 메시징을 유지하여 애플리케이션의 전반적인 보안을 향상시킵니다. AWS PrivateLink를 사용하는 경우 인터넷 게이트웨이, 네트워크 주소 변환(NAT) 또는 가상 사설 네트워크(VPN)를 설정할 필요가 없습니다. 자세한 정보는 인터네트워크 트래픽 개인 정보 보호 및 AWS 보안 블로그의 AWS PrivateLink로 Amazon SNS에 게시된 메시지 보안 게시물을 참조하세요.

SNS FIFO 주제는 가용 영역 전체에서 배달 못한 편지 대기열 및 메시지 스토리지도 지원합니다. 자세한 정보는 FIFO 주제에 대한 메시지 내구성에서 확인하세요.

FIFO 주제에 대한 메시지 내구성

Amazon SNS FIFO 주제 및 Amazon SQS 대기열은 내구성이 뛰어납니다. 두 리소스 유형 모두 여러 가용 영역 전반에서 중복 메시지를 저장하고 예외적인 경우를 처리할 수 있는 배달 못한 편지 대기열을 제공합니다.

Amazon SNS에서 클라이언트 측 또는 서버 측 오류로 인해 Amazon SNS 주제가 구독된 Amazon SQS 대기열에 액세스할 수 없으면 메시지 전송이 실패합니다.

- 클라이언트 측 오류는 Amazon SNS FIFO 주제에 오래된 구독 메타데이터가 있는 경우에 발생합니다. 클라이언트 측 오류가 발생하는 일반적인 원인 두가지는 Amazon SQS 대기열 소유자가 다음 중 하나를 수행하는 경우입니다.
  - 대기열을 삭제합니다.
  - Amazon SNS 서비스 주체가 메시지를 전달하지 못하도록 대기열 정책을 변경합니다.

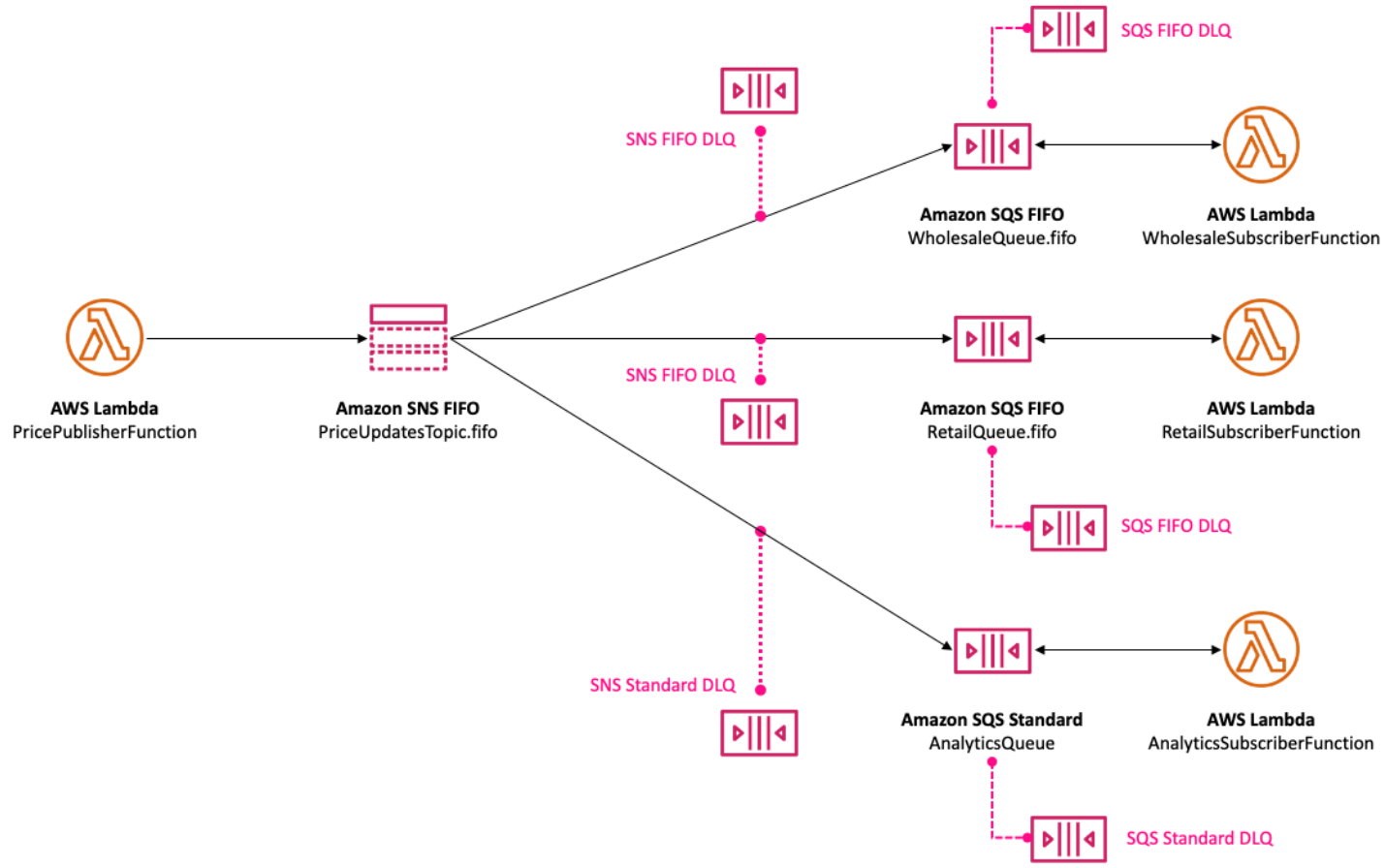
  Amazon SNS 클라이언트 측 오류로 인해 실패한 메시지 전송을 다시 시도하지 않습니다.
- 서버 측 오류는 다음과 같은 상황에서 발생할 수 있습니다.
  - Amazon SQS 서비스를 사용할 수 없습니다.
  - Amazon SQS가 Amazon SNS 서비스의 유효한 요청을 처리하지 못합니다.

서버 측 오류가 발생하면 Amazon SNS FIFO 주제는 23일 동안 최대 100,015회까지 실패한 전송을 다시 시도합니다. 자세한 내용은 Amazon SNS 메시지 전송 재시도 섹션을 참조하세요.

모든 유형의 오류에 대해 Amazon SNS는 데이터가 손실되지 않도록 Amazon SQS 배달 못한 편지 대기열에 대한 메시지를 제외할 수 있습니다.

Amazon SQS에서 소비자 애플리케이션이 메시지 수신, 처리 및 대기열에서 삭제에 실패하면 메시지 처리가 실패합니다. 최대 수신 요청 수가 실패하면 Amazon SQS는 메시지를 배달 못한 편지 대기열로 제외하여 데이터가 손실되지 않도록 할 수 있습니다.

자동차 부품 가격 관리 예 사용 사례에서 회사는 각 Amazon SNS FIFO 주제 구독과 구독된 각 Amazon SQS 대기열에 Amazon SQS DLQ(Dead Letter Queue)를 할당할 수 있습니다. 이러한 작업은 가격 업데이트 손실로부터 회사를 보호합니다.



Amazon SNS 구독에 연결된 DLQ(Dead Letter Queue)는 구독 대기열과 동일한 유형의 Amazon SQS 대기열이어야 합니다. 예를 들어 Amazon SQS FIFO 대기열의 Amazon SNS FIFO 구독에는 DLQ(Dead Letter Queue)로 Amazon SQS FIFO 대기열이 있어야 합니다. 마찬가지로, Amazon SQS

표준 대기열의 Amazon SNS FIFO 구독에는 DLQ(Dead Letter Queue)로 Amazon SQS 표준 대기열이 있어야 합니다. 자세한 정보는 AWS 컴퓨팅 블로그에 게시된 Amazon SNS 배달 못한 편지 대기열(DLQ) 및 Amazon SNS, Amazon SQS 및 AWS Lambda용 DLQ를 사용하여 내구성 있는 서버리스 앱 설계를 참조하세요.

다운스트림 오류 복구를 지원하는 확장된 내구성을 위해 주제 소유자는 FIFO 주제를 사용하여 메시지를 최대 365일까지 아카이브할 수도 있습니다. 주제 구독자는 이러한 메시지를 구독된 엔드포인트에 재생하여 다운스트림 애플리케이션의 오류로 인해 손실된 메시지를 복구하거나 기존 애플리케이션의 상태를 복제할 수 있습니다. 자세한 내용은 FIFO 주제의 메시지 아카이브 및 재생 섹션을 참조하세요.

FIFO 주제의 메시지 아카이브 및 재생

주제

메시지 아카이브 및 재생이란?

Amazon SNS 메시지 아카이브 및 재생은 코드 없는 인플레이스 메시지 아카이브로서, 주제 소유자가 주제 내에 메시지를 저장(또는 아카이브)할 수 있도록 합니다. 주제 구독자는 다음과 같은 용도를 위해 구독한 엔드포인트로 아카이브된 메시지를 다시 가져올(또는 재생할) 수 있습니다.

- 다운스트림 애플리케이션의 오류로 인해 손실되었을 수 있는 메시지를 복구합니다.
- 새 엔드포인트를 구독하고 복제할 원하는 타임스탬프를 선택하여 기존 애플리케이션의 상태를 새 애플리케이션에 복제합니다.

AWS API, SDK, AWS CloudFormation 및 AWS Management Console과 함께 메시지 아카이브 및 재생을 사용할 수 있습니다.

> **Note**
>
> Amazon SNS 메시지 아카이브 및 재생은 애플리케이션 간(A2A) FIFO 주제에만 사용할 수 있습니다.

메시지 아카이브 및 재생은 두 가지 주요 구성 요소로 구성됩니다.

1. 메시지 아카이브 - 주제 소유자가 주제에 대한 아카이브 및 재생 기능을 활성화하고 메시지 보존 기간(최대 365일)을 설정합니다. 주제 소유자는 Amazon CloudWatch 지표를 사용하여 아카이브된 메시지를 모니터링할 수도 있습니다. 자세한 내용은 FIFO 주제 소유자의 메시지 아카이브 섹션을 참조하세요.
2. 메시지 재생 - 주제 구독자가 주제에서 구독한 엔드포인트에 일련의 메시지 재생을 시작합니다. 자세한 내용은 FIFO 주제 구독자를 위한 메시지 재생 섹션을 참조하세요.

FIFO 주제 소유자의 메시지 아카이브

메시지 아카이브 기능을 통해 주제에 게시된 모든 메시지의 단일 사본을 아카이브할 수 있습니다. 주제에 대한 메시지 아카이브 정책을 활성화하여 게시된 메시지를 주제 내에 저장할 수 있습니다. 그러면 해당 주제에 연결된 모든 구독의 메시지를 아카이브할 수 있습니다. 메시지는 최소 1일에서 최대 365일 동안 아카이브될 수 있습니다.

아카이브 정책을 설정하면 추가 요금이 부과됩니다. 요금 정보는 Amazon SNS 요금을 참조하세요.

주제

- AWS Management Console을 사용하여 메시지 아카이브 정책 생성
- API를 사용하여 메시지 아카이브 정책 생성
- SDK를 사용하여 메시지 아카이브 정책 생성
- AWS CloudFormation을 사용하여 메시지 아카이브 정책 생성
- 암호화된 아카이브에 대한 액세스 권한 부여
- Amazon CloudWatch를 사용하여 메시지 아카이브 지표 모니터링

AWS Management Console을 사용하여 메시지 아카이브 정책 생성

AWS Management Console을 사용하여 새 메시지 아카이브 정책을 생성하려면 이 옵션을 사용합니다.

1. Amazon SNS 콘솔에 로그인합니다.
2. 주제를 선택하거나 새로운 주제를 생성합니다. 주제 생성 및 구성에 대한 자세한 내용은 Amazon SNS 주제 생성을 참조하세요.

> **Note**
>
> Amazon SNS 메시지 아카이브 및 재생은 애플리케이션 간(A2A) FIFO 주제에만 사용할 수 있습니다.

3. 주제 편집 페이지에서 아카이브 정책 섹션을 확장합니다.
4. 아카이브 정책 기능을 활성화하고 주제에 메시지를 아카이브하려는 일수를 입력합니다.
5. 변경 사항 저장을 선택합니다.

메시지 아카이브 주제 정책을 확인, 편집 및 비활성화하려면

- 주제 세부 정보 페이지의 보존 정책에는 설정된 일수를 포함한 아카이브 정책의 상태가 표시됩니다. 아카이브 정책 탭을 선택하면 다음 메시지 아카이브 세부 정보를 볼 수 있습니다.
  - 상태 - 아카이브 정책이 적용되면 아카이브 및 재생 상태가 활성으로 표시됩니다. 아카이브 정책이 빈 JSON 객체로 설정되면 아카이브 및 재생 상태가 비활성으로 표시됩니다.
  - 메시지 보존 기간 - 지정된 메시지 보존 일수입니다.
  - 아카이브 시작 날짜 - 구독자가 메시지를 재생할 수 있는 날짜입니다.
  - JSON 미리 보기 - 아카이브 정책의 JSON 미리 보기입니다.
- (선택 사항) 아카이브 정책을 편집하려면 주제 요약 페이지로 이동하여 편집을 선택합니다.
- (선택 사항) 아카이브 정책을 비활성화하려면 주제 요약 페이지로 이동하여 편집을 선택합니다. 아카이브 정책을 비활성화하고 변경 사항 저장을 선택합니다.
- (선택 사항) 아카이브 정책을 사용하여 주제를 삭제하려면 먼저 앞서 설명한 대로 아카이브 정책을 비활성화해야 합니다.

> **Important**
>
> 실수로 메시지가 삭제되는 것을 방지하려는 경우, 활성 메시지 아카이브 정책이 적용된 주제는 삭제할 수 없습니다. 주제를 삭제하려면 먼저 주제의 메시지 아카이브 정책을 비활성화해야 합니다. 메시지 아카이브 정책을 비활성화하면 Amazon SNS에서 아카이브된 모든 메시지가 삭제됩니다. 주제를 삭제하면 구독이 제거되고 전송 중인 메시지가 전송되지 않을 수 있습니다.

API를 사용하여 메시지 아카이브 정책 생성

API를 사용하여 메시지 아카이브 정책을 생성하려면 주제에 ArchivePolicy 속성을 추가해야 합니다. API 작업 CreateTopic 및 SetTopicAttributes를 사용하여 ArchivePolicy를 설정할 수 있습니다. ArchivePolicy에는 Amazon SNS가 메시지를 보관하는 일수를 나타내는 단일 값인 MessageRetentionPeriod가 있습니다. 주제에 대한 메시지 아카이브를 활성화하려면 MessageRetentionPeriod를 0보다 큰 정수 값으로 설정합니다. 예를 들어 아카이브에서 메시지를 30일 동안 보존하려면 ArchivePolicy를 다음과 같이 설정합니다.

```
{
    "ArchivePolicy": {
        "MessageRetentionPeriod": "30"
    }
}
```

주제에 대한 메시지 아카이브를 비활성화하고 아카이브를 지우려면 다음과 같이 ArchivePolicy 설정을 해제합니다.

```
{}
```

SDK를 사용하여 메시지 아카이브 정책 생성

AWS SDK를 사용하려면 자격 증명을 사용하여 구성해야 합니다. 자세한 정보는 AWS SDK 및 도구 참조 가이드의 공유 config 및 credentials 파일을 참조하세요.

다음 코드 예시에서는 해당 주제에 게시된 모든 메시지를 30일 동안 보존하도록 Amazon SNS 주제에 ArchivePolicy를 설정하는 방법을 보여줍니다.

```
// Specify the ARN of the Amazon SNS topic to set the ArchivePolicy for.
String topicArn =
    "arn:aws:sns:us-east-2:123456789012:MyArchiveTopic.fifo";

// Set the MessageRetentionPeriod to 30 days for the ArchivePolicy.
String archivePolicy =
    "{\"MessageRetentionPeriod\":\"30\"}";

// Set the ArchivePolicy for the Amazon SNS topic
SetTopicAttributesRequest request = new SetTopicAttributesRequest()
    .withTopicArn(topicArn)
    .withAttributeName("ArchivePolicy")
    .withAttributeValue(archivePolicy);
```

```
sns.setTopicAttributes(request);
```

AWS CloudFormation을 사용하여 메시지 아카이브 정책 생성

AWS CloudFormation을 사용하여 아카이브 정책을 생성하려면 AWS CloudFormation 사용 설명서의 `AWS::SNS::Topic` 섹션을 참조하세요.

암호화된 아카이브에 대한 액세스 권한 부여

구독자가 암호화된 주제에서 메시지 재생을 시작할 수 있도록 하려면 먼저 다음 단계를 완료해야 합니다. 이전 메시지가 재생되므로 아카이브의 메시지를 암호화하는 데 사용된 KMS 키에 대한 `Decrypt` 액세스 권한을 Amazon SNS에 프로비저닝해야 합니다.

1. KMS 키로 메시지를 암호화하고 주제 내에 저장하는 경우 키 정책을 통해 이러한 메시지를 해독할 수 있는 권한을 Amazon SNS에 부여해야 합니다. 자세한 내용은 Amazon SNS에 암호 해독 권한 부여 섹션을 참조하세요.
2. Amazon SNS용 AWS KMS를 활성화합니다. 자세한 내용은 AWS KMS 권한 구성 섹션을 참조하세요.

> **Important**
>
> KMS 키 정책에 새 섹션을 추가할 때 정책의 기존 섹션을 변경하지 마세요. 주제에 암호화가 활성화되고 KMS 키가 비활성화 또는 삭제되거나 KMS 키 정책이 Amazon SNS에서 올바르게 구성되지 않은 경우 Amazon SNS는 구독자에게 메시지를 재생할 수 없습니다.

Amazon SNS에 암호 해독 권한 부여

Amazon SNS가 주제 아카이브 내에서 암호화된 메시지에 액세스하고 이를 구독한 엔드포인트에 재생하려면 Amazon SNS 서비스 보안 주체를 활성화하여 이러한 메시지를 해독해야 합니다.

다음은 주제 내에서 이전 메시지를 재생하는 동안 Amazon SNS 서비스 보안 주체가 저장된 메시지를 해독하도록 허용하는 데 필요한 정책의 예입니다.

```
{
    "Sid": "Allow SNS to decrypt archived messages",
    "Effect": "Allow",
    "Principal": {
        "Service": "sns.amazonaws.com"
```

```
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*"
}
```

Amazon CloudWatch를 사용하여 메시지 아카이브 지표 모니터링

Amazon CloudWatch에서 다음 지표를 사용하여 아카이브된 메시지를 모니터링할 수 있습니다. 워크로드의 이상 현상에 대한 알림을 받고 관련 영향을 방지하기 위해 이러한 지표에 Amazon CloudWatch 경보를 구성할 수 있습니다. 자세한 내용은 Amazon SNS의 로깅 및 모니터링을(를) 참조하십시오.

지표	Description
ApproximateNumberOfMessagesArchived	주제 소유자에게 주제 아카이브에 아카이브된 총 메시지 수를 60분 단위로 제공합니다.
ApproximateNumberOfBytesArchived	주제 소유자에게 주제 아카이브의 모든 메시지에 걸쳐 아카이브된 총 바이트 수를 60분 단위로 제공합니다.
NumberOfMessagesArchiveProcessing	주제 소유자에게 해당 간격 동안 주제 아카이브에 저장된 메시지 수를 1분 단위로 제공합니다.
NumberOfBytesArchiveProcessing	주제 소유자에게 해당 간격 동안 주제 아카이브에 저장된 총 바이트 수를 1분 단위로 제공합니다.

`GetTopicAttributes` API에는 구독자가 재생을 시작할 수 있는 가장 오래된 타임스탬프를 나타내는 `BeginningArchiveTime` 속성이 있습니다. 다음은 이 API 작업에 대한 샘플 응답입니다.

```
{
 "ArchivePolicy": {
    "MessageRetentionPeriod": "<integer>"
  },
  "BeginningArchiveTime": "<timestamp>",
  ...
```

```
}
```

FIFO 주제 구독자를 위한 메시지 재생

Amazon SNS 재생 기능을 사용하면 주제 구독자가 주제 데이터 스토어에서 아카이브된 메시지를 검색하고 구독한 엔드포인트에 다시 전송(또는 재생)할 수 있습니다. 구독이 생성되는 즉시 메시지를 재생할 수 있습니다. 재생된 메시지에는 원본과 동일한 콘텐츠, `MessageId` 및 `Timestamp`가 있으며 재생된 메시지임을 식별하는 데 도움이 되는 `Replayed` 속성도 포함되어 있습니다. 일부 메시지만 재생하려는 경우 구독에 필터 정책을 추가할 수 있습니다. 메시지 필터링에 대한 자세한 내용은 재생된 메시지 필터링 섹션을 참조하세요.

주제

- AWS Management Console을 사용하여 메시지 재생 정책 생성
- API를 사용하여 구독에 재생 정책 추가
- SDK를 사용하여 구독에 재생 정책 추가
- 재생된 메시지 필터링
- Amazon CloudWatch를 사용하여 메시지 재생 지표 모니터링

AWS Management Console을 사용하여 메시지 재생 정책 생성

AWS Management Console을 사용하여 새 재생 정책을 생성하려면 이 옵션을 사용합니다.

1. Amazon SNS 콘솔에 로그인합니다.
2. 주제 구독을 선택하거나 새 주제 구독을 생성합니다. 구독 생성에 대한 자세한 내용은 Amazon SNS 주제 구독 섹션을 참조하세요.
3. 메시지 재생을 시작하려면 재생 드롭다운으로 이동하여 재생 시작을 선택합니다.
4. 재생 기간 모달에서 다음과 같이 선택합니다.
   a. 재생 시작 날짜 및 시간 선택 - 아카이브된 메시지의 재생을 시작할 날짜(YYYY/MM/DD 형식)와 시간(24시간 hh:mm:ss 형식)을 선택합니다. 시작 시간은 대략적인 아카이브 시간의 시작 시간보다 이후여야 합니다.
   b. (선택 사항) 재생 종료 날짜 및 시간 선택 - 아카이브된 메시지의 재생을 중지할 날짜(YYYY/MM/DD 형식)와 시간(24시간 hh:mm:ss 형식)을 선택합니다.
   c. 재생 시작을 선택합니다.
5. (선택 사항) 메시지 재생을 중지하려면 구독 세부 정보 페이지로 이동한 다음 재생 드롭다운에서 재생 중지를 선택합니다.

6. (선택 사항) CloudWatch를 사용하여 이 워크플로 내에서 메시지 재생 지표를 모니터링하려면 Amazon CloudWatch를 사용하여 메시지 재생 지표 모니터링 섹션을 참조하세요.

메시지 재생 정책을 확인하고 편집하려면

구독 세부 정보 페이지에서 다음 작업을 수행할 수 있습니다.

- 메시지 재생 상태를 확인하려는 경우, 재생 상태 필드에 다음 값이 표시됩니다.
  - 완료 - 재생 기능이 모든 메시지를 성공적으로 다시 전송했으며 이제 새로 게시된 메시지를 전송하고 있습니다.
  - 진행 중 - 현재 선택한 메시지가 재생 중입니다.
  - 실패 - 재생을 완료할 수 없습니다.
  - 보류 중 - 재생이 시작되는 동안의 기본 상태입니다.
- (선택 사항) 메시지 재생 정책을 수정하려면 구독 세부 정보 페이지로 이동한 다음 재생 드롭다운에서 재생 시작을 선택합니다. 재생을 시작하면 기존 재생이 대체됩니다.

API를 사용하여 구독에 재생 정책 추가

아카이브된 메시지를 재생하려면 `ReplayPolicy` 속성을 사용합니다. `ReplayPolicy`는 `Subscribe` 및 `SetSubscriptionAttributes` API 작업과 함께 사용할 수 있습니다. 이 정책에는 다음 값이 포함되어 있습니다.

- `StartingPoint`(필수) - 메시지 재생을 시작할 위치를 나타냅니다.
- `EndingPoint`(선택 사항) - 메시지 재생을 중지할 시기를 나타냅니다. `EndingPoint`를 생략하면 현재 시간에 도달할 때까지 재생이 계속됩니다.
- `PointType`(필수) - 시작점과 종료점의 유형을 설정합니다. 현재 `PointType`에 지원되는 값은 `Timestamp`입니다.

예를 들어 다운스트림 오류를 복구하고 2023년 10월 1일에 2시간 동안 모든 메시지를 다시 보내려면 `SetSubscriptionAttributes` API 작업을 사용하여 다음과 같이 `ReplayPolicy`를 설정합니다.

```
{
  "PointType":"Timestamp",
  "StartingPoint":"2023-10-01T10:00:00.000Z",
  "EndingPoint":"2023-10-01T12:00:00.000Z"
}
```

2023년 10월 1일 현재 주제에 전송된 모든 메시지를 재생하고 주제에 새로 게시된 모든 메시지를 계속 수신하려면 SetSubscriptionAttributes API 작업을 사용하여 다음과 같이 구독에 ReplayPolicy를 설정합니다.

```
{
  "PointType":"Timestamp",
  "StartingPoint":"2023-10-01T00:00:00.000Z"
}
```

메시지가 재생되었는지 확인하기 위해 재생된 각 메시지에 부울 속성 Replayed가 추가됩니다.

SDK를 사용하여 구독에 재생 정책 추가

AWS SDK를 사용하려면 자격 증명을 사용하여 구성해야 합니다. 자세한 정보는 AWS SDK 및 도구 참조 가이드의 공유 config 및 credentials 파일을 참조하세요.

다음 코드 예제는 2023년 10월 1일 2시간 동안 Amazon SNS FIFO 주제 아카이브에서 메시지를 재전송하도록 구독에 ReplayPolicy를 설정하는 방법을 보여줍니다.

```
// Specify the ARN of the Amazon SNS subscription to initiate the ReplayPolicy on.
String subscriptionArn =
    "arn:aws:sns:us-
east-2:123456789012:MyArchiveTopic.fifo:1d2a3e9d-7f2f-447c-88ae-03f1c68294da";

// Set the ReplayPolicy to replay messages from the topic's archive
// for a 2 hour time period on October 1st 2023 between 10am and 12pm UTC.
String replayPolicy =
    "{\"PointType\":\"Timestamp\",\"StartingPoint\":\"2023-10-01T10:00:00.000Z\",
\"EndingPoint\":\"2023-10-01T12:00:00.000Z\"}";

// Set the ArchivePolicy for the Amazon SNS topic
SetSubscriptionAttributesRequest request = new SetSubscriptionAttributesRequest()
    .withSubscriptionArn(subscriptionArn)
    .withAttributeName("ReplayPolicy")
    .withAttributeValue(replayPolicy);
sns.setSubscriptionAttributes(request);
```

재생된 메시지 필터링

Amazon SNS 메시지 필터링을 사용하면 Amazon SNS가 구독자 엔드포인트에 재생하는 재생된 메시지를 제어할 수 있습니다. 메시지 필터링과 메시지 보관이 모두 활성화된 경우 Amazon SNS는 먼저 주제의 데이터 스토어에서 메시지를 검색한 다음 구독의 FilterPolicy에 대해 메시지를 적용합니다.

일치하는 항목이 있으면 메시지가 구독된 엔드포인트로 전송되고 그렇지 않으면 메시지가 필터링됩니다. 자세한 내용은 Amazon SNS 구독 필터 정책 섹션을 참조하세요.

Amazon CloudWatch를 사용하여 메시지 재생 지표 모니터링

Amazon CloudWatch에서 다음 지표를 사용하여 재생 메시지를 모니터링할 수 있습니다. 워크로드의 이상 현상에 대한 알림을 받고 관련 영향을 방지하기 위해 이러한 지표에 Amazon CloudWatch 경보를 구성할 수 있습니다. 자세한 내용은 Amazon SNS의 로깅 및 모니터링을(를) 참조하십시오.

지표	Description
NumberOfReplayedNotificationsDelivered	구독자에게 주제 아카이브에서 재생된 총 메시지 수를 1분 단위로 제공합니다.
NumberOfReplayedNotificationsFailed	구독자에게 주제 아카이브에서 전송에 실패한 재생된 총 메시지 수를 1분 단위로 제공합니다.

FIFO 주제에 대한 코드 예제

다음 코드 예제를 통해 Amazon SNS FIFO 주제를 사용하는 자동차 부품 가격 관리 예 사용 사례를 Amazon SQS FIFO 대기열 또는 표준 대기열과 통합할 수 있습니다.

주제

- AWS SDK 사용
- AWS CloudFormation 사용

AWS SDK 사용

AWS SDK를 사용해 `FifoTopic` 속성을 **`true`**로 설정하여 Amazon SNS FIFO 주제를 생성합니다. `FifoQueue` 속성을 **`true`**로 설정하여 Amazon SQS FIFO 대기열을 생성합니다. 또한 각 FIFO 리소스의 이름에 **`.fifo`** 접미사를 추가해야 합니다. FIFO 주제 또는 대기열을 생성한 후에는 표준 주제 또는 대기열로 변환할 수 없습니다.

다음 코드 예제에서는 이러한 FIFO 및 표준 대기열 리소스를 생성합니다.

- 가격 업데이트를 배포하는 Amazon SNS FIFO 주제
- 도매 및 소매 애플리케이션에 이러한 업데이트를 제공하는 Amazon SQS FIFO 대기열

- 비즈니스 인텔리전스(BI) 목적으로 쿼리할 수 있는 레코드를 저장하는 분석 애플리케이션에 대한 Amazon SQS 표준 대기열
- 세 대기열을 주제에 연결하는 Amazon SNS FIFO 구독

이 예에서는 구독에 대한 필터 정책을 설정합니다. 주제에 메시지를 게시하여 예제를 테스트하는 경우 `business` 속성을 사용하여 메시지를 게시해야 합니다. 속성 값에 대해 `retail` 또는 `wholesale` 중 하나를 지정합니다. 그렇지 않으면 메시지가 필터링되어 구독된 대기열에 배달되지 않습니다. 자세한 내용은 FIFO 주제에 대한 메시지 필터링 섹션을 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예에서는

- Amazon SNS FIFO 주제 1개, Amazon SQS FIFO 대기열 2개, 표준 대기열 1개를 생성합니다.
- 대기열에서 주제를 구독하고 해당 주제에 메시지를 게시합니다.

테스트에서는 각 대기열에 대한 메시지 수신 여부를 확인합니다. 또한 전체 예제에서는 액세스 정책을 추가하는 것을 보여주고 마지막에 리소스를 삭제합니다.

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {

        final String usage = "\n" +
                "Usage: " +
                "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
 <analyticsQueueName>\n\n" +
                "Where:\n" +
                "   fifoTopicName - The name of the FIFO topic that you want to
 create. \n\n" +
```

```
                "   wholesaleQueueARN - The name of a SQS FIFO queue that will be
created for the wholesale consumer. \n\n"
                +
                "   retailQueueARN - The name of a SQS FIFO queue that will
created for the retail consumer. \n\n" +
                "   analyticsQueueARN - The name of a SQS standard queue that
will be created for the analytics consumer. \n\n";
        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        final String fifoTopicName = args[0];
        final String wholeSaleQueueName = args[1];
        final String retailQueueName = args[2];
        final String analyticsQueueName = args[3];

        // For convenience, the QueueData class holds metadata about a queue:
ARN, URL,
        // name and type.
        List<QueueData> queues = List.of(
                new QueueData(wholeSaleQueueName, QueueType.FIFO),
                new QueueData(retailQueueName, QueueType.FIFO),
                new QueueData(analyticsQueueName, QueueType.Standard));

        // Create queues.
        createQueues(queues);

        // Create a topic.
        String topicARN = createFIFOTopic(fifoTopicName);

        // Subscribe each queue to the topic.
        subscribeQueues(queues, topicARN);

        // Allow the newly created topic to send messages to the queues.
        addAccessPolicyToQueuesFINAL(queues, topicARN);

        // Publish a sample price update message with payload.
        publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

        // Clean up resources.
        deleteSubscriptions(queues);
        deleteQueues(queues);
```

```
        deleteTopic(topicARN);
    }

    public static String createFIFOTopic(String topicName) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = Map.of(
                    "FifoTopic", "true",
                    "ContentBasedDeduplication", "false");

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                    .name(topicName)
                    .attributes(topicAttributes)
                    .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            String topicArn = response.topicArn();
            System.out.println("The topic ARN is" + topicArn);

            return topicArn;

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void subscribeQueues(List<QueueData> queues, String topicARN) {
        queues.forEach(queue -> {
            SubscribeRequest subscribeRequest = SubscribeRequest.builder()
                    .topicArn(topicARN)
                    .endpoint(queue.queueARN)
                    .protocol("sqs")
                    .build();

            // Subscribe to the endpoint by using the SNS service client.
            // Only Amazon SQS queues can receive notifications from an Amazon
SNS FIFO
            // topic.
            SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
            System.out.println("The queue [" + queue.queueARN + "] subscribed to
the topic [" + topicARN + "]");
```

```
            queue.subscriptionARN = subscribeResponse.subscriptionArn();
        });
    }

    public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

        try {
            // Create and publish a message that updates the wholesale price.
            String subject = "Price Update";
            String dedupId = UUID.randomUUID().toString();
            String attributeName = "business";
            String attributeValue = "wholesale";

            MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
                    .dataType("String")
                    .stringValue(attributeValue)
                    .build();

            Map<String, MessageAttributeValue> attributes = new HashMap<>();
            attributes.put(attributeName, msgAttValue);
            PublishRequest pubRequest = PublishRequest.builder()
                    .topicArn(topicArn)
                    .subject(subject)
                    .message(payload)
                    .messageGroupId(groupId)
                    .messageDeduplicationId(dedupId)
                    .messageAttributes(attributes)
                    .build();

            final PublishResponse response = snsClient.publish(pubRequest);
            System.out.println(response.messageId());
            System.out.println(response.sequenceNumber());
            System.out.println("Message was published to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 주제를 참조하십시오.

- CreateTopic
- 게시
- Subscribe

Python

SDK for Python (Boto3)

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon FIFO 주제를 생성하고, Amazon SQS FIFO 및 표준 대기열에서 주제를 구독하고, 해당 주제에 메시지를 게시합니다.

```
def usage_demo():
    """Shows how to subscribe queues to a FIFO topic."""
    print("-" * 88)
    print("Welcome to the `Subscribe queues to a FIFO topic` demo!")
    print("-" * 88)

    sns = boto3.resource("sns")
    sqs = boto3.resource("sqs")
    fifo_topic_wrapper = FifoTopicWrapper(sns)
    sns_wrapper = SnsWrapper(sns)

    prefix = "sqs-subscribe-demo-"
    queues = set()
    subscriptions = set()

    wholesale_queue = sqs.create_queue(
        QueueName=prefix + "wholesale.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
```

```
    )
    queues.add(wholesale_queue)
    print(f"Created FIFO queue with URL: {wholesale_queue.url}.")

    retail_queue = sqs.create_queue(
        QueueName=prefix + "retail.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(retail_queue)
    print(f"Created FIFO queue with URL: {retail_queue.url}.")

    analytics_queue = sqs.create_queue(QueueName=prefix + "analytics",
Attributes={})
    queues.add(analytics_queue)
    print(f"Created standard queue with URL: {analytics_queue.url}.")

    topic = fifo_topic_wrapper.create_fifo_topic("price-updates-topic.fifo")
    print(f"Created FIFO topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        fifo_topic_wrapper.add_access_policy(q, topic.attributes["TopicArn"])

    print(f"Added access policies for topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        sub = fifo_topic_wrapper.subscribe_queue_to_topic(
            topic, q.attributes["QueueArn"]
        )
        subscriptions.add(sub)

    print(f"Subscribed queues to topic: {topic.attributes['TopicArn']}.")

    input("Press Enter to publish a message to the topic.")

    message_id = fifo_topic_wrapper.publish_price_update(
        topic, '{"product": 214, "price": 79.99}', "Consumables"
    )
```

```
    print(f"Published price update with message ID: {message_id}.")

    # Clean up the subscriptions, queues, and topic.
    input("Press Enter to clean up resources.")
    for s in subscriptions:
        sns_wrapper.delete_subscription(s)

    sns_wrapper.delete_topic(topic)

    for q in queues:
        fifo_topic_wrapper.delete_queue(q)

    print(f"Deleted subscriptions, queues, and topic.")

    print("Thanks for watching!")
    print("-" * 88)



class FifoTopicWrapper:
    """Encapsulates Amazon SNS FIFO topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource

    def create_fifo_topic(self, topic_name):
        """
        Create a FIFO topic.
        Topic names must be made up of only uppercase and lowercase ASCII
 letters,
        numbers, underscores, and hyphens, and must be between 1 and 256
 characters long.
        For a FIFO topic, the name must end with the .fifo suffix.

        :param topic_name: The name for the topic.
        :return: The new topic.
        """
        try:
            topic = self.sns_resource.create_topic(
                Name=topic_name,
                Attributes={
```

```
                    "FifoTopic": str(True),
                    "ContentBasedDeduplication": str(False),
                },
            )
            logger.info("Created FIFO topic with name=%s.", topic_name)
            return topic
        except ClientError as error:
            logger.exception("Couldn't create topic with name=%s!", topic_name)
            raise error


    @staticmethod
    def add_access_policy(queue, topic_arn):
        """
        Add the necessary access policy to a queue, so
        it can receive messages from a topic.

        :param queue: The queue resource.
        :param topic_arn: The ARN of the topic.
        :return: None.
        """
        try:
            queue.set_attributes(
                Attributes={
                    "Policy": json.dumps(
                        {
                            "Version": "2012-10-17",
                            "Statement": [
                                {
                                    "Sid": "test-sid",
                                    "Effect": "Allow",
                                    "Principal": {"AWS": "*"},
                                    "Action": "SQS:SendMessage",
                                    "Resource": queue.attributes["QueueArn"],
                                    "Condition": {
                                        "ArnLike": {"aws:SourceArn": topic_arn}
                                    },
                                }
                            ],
                        }
                    )
                }
            )
            logger.info("Added trust policy to the queue.")
```

```
        except ClientError as error:
            logger.exception("Couldn't add trust policy to the queue!")
            raise error


    @staticmethod
    def subscribe_queue_to_topic(topic, queue_arn):
        """
        Subscribe a queue to a topic.

        :param topic: The topic resource.
        :param queue_arn: The ARN of the queue.
        :return: The subscription resource.
        """
        try:
            subscription = topic.subscribe(
                Protocol="sqs",
                Endpoint=queue_arn,
            )
            logger.info("The queue is subscribed to the topic.")
            return subscription
        except ClientError as error:
            logger.exception("Couldn't subscribe queue to topic!")
            raise error


    @staticmethod
    def publish_price_update(topic, payload, group_id):
        """
        Compose and publish a message that updates the wholesale price.

        :param topic: The topic to publish to.
        :param payload: The message to publish.
        :param group_id: The group ID for the message.
        :return: The ID of the message.
        """
        try:
            att_dict = {"business": {"DataType": "String", "StringValue":
"wholesale"}}
            dedup_id = uuid.uuid4()
            response = topic.publish(
                Subject="Price Update",
                Message=payload,
                MessageAttributes=att_dict,
```

```
                MessageGroupId=group_id,
                MessageDeduplicationId=str(dedup_id),
            )
            message_id = response["MessageId"]
            logger.info("Published message to topic %s.", topic.arn)
        except ClientError as error:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise error
        return message_id


    @staticmethod
    def delete_queue(queue):
        """
        Removes an SQS queue. When run against an AWS account, it can take up to
        60 seconds before the queue is actually deleted.

        :param queue: The queue to delete.
        :return: None
        """
        try:
            queue.delete()
            logger.info("Deleted queue with URL=%s.", queue.url)
        except ClientError as error:
            logger.exception("Couldn't delete queue with URL=%s!", queue.url)
            raise error
```

- API 세부 정보는 AWSSDK for Python (Boto3) API 참조의 다음 주제를 참조하십시오.
  - CreateTopic
  - 게시
  - Subscribe

SAP ABAP

SDK for SAP ABAP API

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

FIFO 주제를 생성하고, 주제에 대한 Amazon SQS FIFO 대기열을 구독하고, Amazon SNS 주제에 메시지를 게시합니다.

```
    " Creates a FIFO topic. "
    DATA lt_tpc_attributes TYPE /aws1/
cl_snstopicattrsmap_w=>tt_topicattributesmap.
    DATA ls_tpc_attributes TYPE /aws1/
cl_snstopicattrsmap_w=>ts_topicattributesmap_maprow.
    ls_tpc_attributes-key = 'FifoTopic'.
    ls_tpc_attributes-value = NEW /aws1/cl_snstopicattrsmap_w( iv_value =
 'true' ).
    INSERT ls_tpc_attributes INTO TABLE lt_tpc_attributes.

    TRY.
        DATA(lo_create_result) = lo_sns->createtopic(
               iv_name = iv_topic_name
               it_attributes = lt_tpc_attributes
        ).
        DATA(lv_topic_arn) = lo_create_result->get_topicarn( ).
        ov_topic_arn = lv_topic_arn.                                        "
 ov_topic_arn is returned for testing purposes. "
        MESSAGE 'FIFO topic created' TYPE 'I'.
      CATCH /aws1/cx_snstopiclimitexcdex.
        MESSAGE 'Unable to create more topics. You have reached the maximum
 number of topics allowed.' TYPE 'E'.
    ENDTRY.

    " Subscribes an endpoint to an Amazon Simple Notification Service (Amazon
 SNS) topic. "
    " Only Amazon Simple Queue Service (Amazon SQS) FIFO queues can be subscribed
 to an SNS FIFO topic. "
```

```
    TRY.
        DATA(lo_subscribe_result) = lo_sns->subscribe(
               iv_topicarn = lv_topic_arn
               iv_protocol = 'sqs'
               iv_endpoint = iv_queue_arn
           ).
        DATA(lv_subscription_arn) = lo_subscribe_result->get_subscriptionarn( ).
        ov_subscription_arn = lv_subscription_arn.                        "
 ov_subscription_arn is returned for testing purposes. "
        MESSAGE 'SQS queue was subscribed to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snssubscriptionlmte00.
        MESSAGE 'Unable to create subscriptions. You have reached the maximum
 number of subscriptions allowed.' TYPE 'E'.
    ENDTRY.

    " Publish message to SNS topic. "
    TRY.
        DATA lt_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>tt_messageattributemap.
        DATA ls_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>ts_messageattributemap_maprow.
        ls_msg_attributes-key = 'Importance'.
        ls_msg_attributes-value = NEW /aws1/cl_snsmessageattrvalue( iv_datatype =
 'String' iv_stringvalue = 'High' ).
        INSERT ls_msg_attributes INTO TABLE lt_msg_attributes.

        DATA(lo_result) = lo_sns->publish(
             iv_topicarn = lv_topic_arn
             iv_message = 'The price of your mobile plan has been increased from
 $19 to $23'
             iv_subject = 'Changes to mobile plan'
             iv_messagegroupid = 'Update-2'
             iv_messagededuplicationid = 'Update-2.1'
             it_messageattributes = lt_msg_attributes
      ).
        ov_message_id = lo_result->get_messageid( ).                      "
 ov_message_id is returned for testing purposes. "
        MESSAGE 'Message was published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 다음 주제를 참조하십시오.
  - CreateTopic
  - 게시
  - Subscribe

FIFO 구독의 메시지 받기

이제 구독한 애플리케이션 3개에서 가격 업데이트를 받을 수 있습니다. the section called “FIFO 주제 사용 사례”에 표시된 대로 각 소비자 애플리케이션의 진입점은 해당 AWS Lambda 함수가 자동으로 폴링할 수 있는 Amazon SQS 대기열입니다. Amazon SQS 대기열이 Lambda 함수의 이벤트 소스인 경우 Lambda는 메시지를 효율적으로 소비하기 위해 필요에 따라 폴러의 폴링을 확장합니다.

자세한 정보는 AWS Lambda 개발자 안내서의 Amazon SQS에서 AWS Lambda 사용을 참조하세요. 사용자 고유의 대기열 폴러 작성에 대한 자세한 내용은 Amazon Simple Queue Service 개발자 안내서의 Amazon SQS 표준 및 FIFO 대기열에 대한 권장 사항 및 Amazon Simple Queue Service API 참조의 ReceiveMessage를 참조하세요.

AWS CloudFormation 사용

AWS CloudFormation을 통해 템플릿 파일을 사용하여 AWS 리소스 모음을 단일 유닛으로 생성 및 구성할 수 있습니다. 이 섹션은 다음을 생성하는 예제 템플릿을 보여줍니다.

- 가격 업데이트를 배포하는 Amazon SNS FIFO 주제
- 도매 및 소매 애플리케이션에 이러한 업데이트를 제공하는 Amazon SQS FIFO 대기열
- 비즈니스 인텔리전스(BI) 목적으로 쿼리할 수 있는 레코드를 저장하는 분석 애플리케이션에 대한 Amazon SQS 표준 대기열
- 세 대기열을 주제에 연결하는 Amazon SNS FIFO 구독
- 구독자 애플리케이션이 필요한 가격 업데이트만 수신하도록 지정하는 필터 정책

> **Note**
>
> 주제에 메시지를 게시하여 이 코드 샘플을 테스트하는 경우 `business` 속성을 사용하여 메시지를 게시해야 합니다. 속성 값에 대해 `retail` 또는 `wholesale` 중 하나를 지정합니다. 그렇지 않으면 메시지가 필터링되어 구독된 대기열에 배달되지 않습니다.

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Resources": {
    "PriceUpdatesTopic": {
      "Type": "AWS::SNS::Topic",
      "Properties": {
        "TopicName": "PriceUpdatesTopic.fifo",
        "FifoTopic": true,
        "ContentBasedDeduplication": false,
        "ArchivePolicy": {
        "MessageRetentionPeriod": "30"
        }
      }
    },
    "WholesaleQueue": {
      "Type": "AWS::SQS::Queue",
      "Properties": {
        "QueueName": "WholesaleQueue.fifo",
        "FifoQueue": true,
        "ContentBasedDeduplication": false
      }
    },
    "RetailQueue": {
      "Type": "AWS::SQS::Queue",
      "Properties": {
        "QueueName": "RetailQueue.fifo",
        "FifoQueue": true,
        "ContentBasedDeduplication": false
      }
    },
    "AnalyticsQueue": {
      "Type": "AWS::SQS::Queue",
      "Properties": {
        "QueueName": "AnalyticsQueue"
      }
    },
    "WholesaleSubscription": {
      "Type": "AWS::SNS::Subscription",
      "Properties": {
        "TopicArn": {
          "Ref": "PriceUpdatesTopic"
        },
        "Endpoint": {
```

```
          "Fn::GetAtt": [
            "WholesaleQueue",
            "Arn"
          ]
        },
        "Protocol": "sqs",
        "RawMessageDelivery": "false",
        "FilterPolicyScope": "MessageBody",
        "FilterPolicy": {
          "business": [
            "wholesale"
          ]
        }
      }
    },
    "RetailSubscription": {
      "Type": "AWS::SNS::Subscription",
      "Properties": {
        "TopicArn": {
          "Ref": "PriceUpdatesTopic"
        },
        "Endpoint": {
          "Fn::GetAtt": [
            "RetailQueue",
            "Arn"
          ]
        },
        "Protocol": "sqs",
        "RawMessageDelivery": "false",
        "FilterPolicyScope": "MessageBody",
        "FilterPolicy": {
          "business": [
            "retail"
          ]
        }
      }
    },
    "AnalyticsSubscription": {
      "Type": "AWS::SNS::Subscription",
      "Properties": {
        "TopicArn": {
          "Ref": "PriceUpdatesTopic"
        },
        "Endpoint": {
```

```
          "Fn::GetAtt": [
            "AnalyticsQueue",
            "Arn"
          ]
        },
        "Protocol": "sqs",
        "RawMessageDelivery": "false"
      }
    },
    "SalesQueuesPolicy": {
      "Type": "AWS::SQS::QueuePolicy",
      "Properties": {
        "PolicyDocument": {
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Service": "sns.amazonaws.com"
              },
              "Action": [
                "sqs:SendMessage"
              ],
              "Resource": "*",
              "Condition": {
                "ArnEquals": {
                  "aws:SourceArn": {
                    "Ref": "PriceUpdatesTopic"
                  }
                }
              }
            }
          ]
        },
        "Queues": [
          {
            "Ref": "WholesaleQueue"
          },
          {
            "Ref": "RetailQueue"
          },
          {
            "Ref": "AnalyticsQueue"
          }
        ]
```

```
        }
      }
    }
}
```

AWS CloudFormation 템플릿을 사용하여 AWS 리소스를 배포하는 방법에 대한 자세한 정보는 AWS CloudFormation 사용 설명서의 시작하기를 참조하세요.

Amazon SNS 메시지 게시

Amazon SNS 주제를 생성하고 엔드포인트에서 주제를 구독 설정한 후 주제에 메시지를 게시할 수 있습니다. 메시지가 게시되면 Amazon SNS는 구독 설정된 엔드포인트로 메시지를 전달하려고 시도합니다.

주제

- AWS Management Console을 사용하여 Amazon SNS 주제에 메시지를 게시하려면
- AWS SDK를 사용하여 주제에 메시지를 게시하려면
- Amazon SNS 및 Amazon S3 S3를 사용하여 대용량 메시지 게시
- Amazon SNS 메시지 속성
- Amazon SNS 메시지 배치 처리

AWS Management Console을 사용하여 Amazon SNS 주제에 메시지를 게시하려면

1. Amazon SNS 콘솔에 로그인합니다.
2. 왼쪽 탐색 창에서 주제를 선택합니다.
3. 주제 페이지에서 주제를 선택하고 메시지 게시를 선택합니다.

   콘솔에서 주제에 메시지 게시 페이지가 열립니다.
4. 메시지 세부 정보 섹션에서 다음을 수행합니다.

   a. (선택 사항) 메시지 제목을 입력합니다.

   b. FIFO 주제에 메시지 그룹 ID를 입력합니다. 동일한 메시지 그룹의 메시지는 게시된 순서대로 전송됩니다.

   c. FIFO 주제에 메시지 중복 제거 ID를 입력합니다. 주제에 대해 콘텐츠 기반 메시지 중복 제거 설정을 사용하는 경우 이 ID는 선택 사항입니다.

   d. (선택 사항) 모바일 푸시 알림에 초 단위로 유지 시간(TTL)을 입력합니다. 이 시간은 Apple Push Notification Service(APNS) 또는 Firebase Cloud Messaging(FCM)과 같은 푸시 알림 서비스가 메시지를 엔드포인트에 전달해야 하는 시간입니다.
5. 메시지 본문 섹션에서 다음 중 하나를 수행합니다.

   a. 모든 전송 프로토콜에 대해 동일한 페이로드를 선택한 다음 메시지를 입력합니다.

b. 각 전송 프로토콜에 대해 사용자 지정 페이로드를 선택한 다음 JSON 객체를 입력하여 각 전송 프로토콜에 보낼 메시지를 정의합니다.

   자세한 정보는 플랫폼별 페이로드를 사용한 게시을 참조하세요.

6. 메시지 속성 섹션에서, Amazon SNS를 구독 속성인 `FilterPolicy`와 일치시켜 구독된 엔드포인트가 게시된 메시지와 관련이 있는지 확인하려는 속성을 추가합니다.

   a. Type에서 속성 유형(예:String.Array)을 선택합니다.

   > **Note**
   >
   > 속성 유형 String.Array에 대해 배열을 대괄호로 묶습니다(`[]`). 배열 안에서는 문자열 값을 큰따옴표로 묶습니다. 숫자 또는 키워드 `true`, `false` 및 `null`에는 따옴표가 필요 없습니다.

   b. 속성 Name(예: `customer_interests`)을 입력합니다.

   c. 속성 Value(예: `["soccer", "rugby", "hockey"]`)를 입력합니다.

   속성 유형이 String, String.Array 또는 Number인 경우, 필터 정책 범위가 명시적으로 `MessageBody`로 설정되지 않았다면 Amazon SNS는 구독에 메시지를 전송하기 전에 구독의 필터 정책(있는 경우)에 따라 메시지 속성을 평가합니다.

   자세한 정보는 Amazon SNS 메시지 속성을 참조하세요.

7. 메시지 게시를 선택합니다.

   메시지가 주제에 게시되고 콘솔에서 주제의 세부 정보 페이지가 열립니다.

AWS SDK를 사용하여 주제에 메시지를 게시하려면

AWS SDK를 사용하려면 사용자 인증 정보로 구성해야 합니다. 자세한 정보는 AWS SDK 및 도구 참조 가이드의 공유 구성 및 자격 증명 파일을 참조하세요.

다음 코드 예제는 사용 `Publish` 방법을 보여줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 내용이 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

주제에 메시지를 게시합니다.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example publishes a message to an Amazon Simple Notification
    /// Service (Amazon SNS) topic.
    /// </summary>
    public class PublishToSNSTopic
    {
        public static async Task Main()
        {
            string topicArn = "arn:aws:sns:us-
east-2:000000000000:ExampleSNSTopic";
            string messageText = "This is an example message to publish to the
 ExampleSNSTopic.";

            IAmazonSimpleNotificationService client = new
 AmazonSimpleNotificationServiceClient();

            await PublishToTopicAsync(client, topicArn, messageText);
        }

        /// <summary>
        /// Publishes a message to an Amazon SNS topic.
        /// </summary>
        /// <param name="client">The initialized client object used to publish
        /// to the Amazon SNS topic.</param>
        /// <param name="topicArn">The ARN of the topic.</param>
        /// <param name="messageText">The text of the message.</param>
```

```
        public static async Task PublishToTopicAsync(
            IAmazonSimpleNotificationService client,
            string topicArn,
            string messageText)
        {
            var request = new PublishRequest
            {
                TopicArn = topicArn,
                Message = messageText,
            };

            var response = await client.PublishAsync(request);

            Console.WriteLine($"Successfully published message ID:
{response.MessageId}");
        }
    }
```

그룹, 복제, 속성 옵션을 사용하여 주제에 메시지를 게시하세요.

```
    /// <summary>
    /// Publish messages using user settings.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task PublishMessages()
    {
        Console.WriteLine("Now we can publish messages.");

        var keepSendingMessages = true;
        string? deduplicationId = null;
        string? toneAttribute = null;
        while (keepSendingMessages)
        {
            Console.WriteLine();
            var message = GetUserResponse("Enter a message to publish.", "This is
a sample message");

            if (_useFifoTopic)
            {
                Console.WriteLine("Because you are using a FIFO topic, you must
set a message group ID." +
```

```
                                    "\r\nAll messages within the same group will be
received in the order " +
                                    "they were published.");

                Console.WriteLine();
                var messageGroupId = GetUserResponse("Enter a message group ID
for this message:", "1");

                if (!_useContentBasedDeduplication)
                {
                    Console.WriteLine("Because you are not using content-based
deduplication, " +
                                      "you must enter a deduplication ID.");

                    Console.WriteLine("Enter a deduplication ID for this
message.");
                    deduplicationId = GetUserResponse("Enter a deduplication ID
for this message.", "1");
                }

                if (GetYesNoResponse("Add an attribute to this message?"))
                {
                    Console.WriteLine("Enter a number for an attribute.");
                    for (int i = 0; i < _tones.Length; i++)
                    {
                        Console.WriteLine($"\t{i + 1}. {_tones[i]}");
                    }

                    var selection = GetUserResponse("", "1");
                    int.TryParse(selection, out var selectionNumber);

                    if (selectionNumber > 0 && selectionNumber < _tones.Length)
                    {
                        toneAttribute = _tones[selectionNumber - 1];
                    }
                }

                var messageID = await SnsWrapper.PublishToTopicWithAttribute(
                    _topicArn, message, "tone", toneAttribute, deduplicationId,
messageGroupId);

                Console.WriteLine($"Message published with id {messageID}.");
            }
```

```
            keepSendingMessages = GetYesNoResponse("Send another message?",
 false);
        }
    }
```

사용자의 선택을 게시 작업에 적용합니다.

```
    /// <summary>
    /// Publish a message to a topic with an attribute and optional deduplication
 and group IDs.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="message">The message to publish.</param>
    /// <param name="attributeName">The optional attribute for the message.</
param>
    /// <param name="attributeValue">The optional attribute value for the
 message.</param>
    /// <param name="deduplicationId">The optional deduplication ID for the
 message.</param>
    /// <param name="groupId">The optional group ID for the message.</param>
    /// <returns>The ID of the message published.</returns>
    public async Task<string> PublishToTopicWithAttribute(
        string topicArn,
        string message,
        string? attributeName = null,
        string? attributeValue = null,
        string? deduplicationId = null,
        string? groupId = null)
    {
        var publishRequest = new PublishRequest()
        {
            TopicArn = topicArn,
            Message = message,
            MessageDeduplicationId = deduplicationId,
            MessageGroupId = groupId
        };

        if (attributeValue != null)
        {
            // Add the string attribute if it exists.
            publishRequest.MessageAttributes =
                new Dictionary<string, MessageAttributeValue>
```

```
                {
                    { attributeName!, new MessageAttributeValue() { StringValue =
 attributeValue, DataType = "String"} }
                };
        }

        var publishResponse = await
 _amazonSNSClient.PublishAsync(publishRequest);
        return publishResponse.MessageId;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 Publish를 참조하십시오.

C++

SDK for C++

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
//! Send a message to an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param message: The message to publish.
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::publishToTopic(const Aws::String &message,
                                 const Aws::String &topicARN,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);
```

```
    if (outcome.IsSuccess()) {
        std::cout << "Message published successfully with id '"
                  << outcome.GetResult().GetMessageId() << "'." << std::endl;
    }
    else {
        std::cerr << "Error while publishing message "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

속성이 있는 메시지를 게시하세요.

```
        static const Aws::String TONE_ATTRIBUTE("tone");
        static const Aws::Vector<Aws::String> TONES = {"cheerful", "funny",
 "serious",
                                                       "sincere"};

        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

        Aws::SNS::Model::PublishRequest request;
        request.SetTopicArn(topicARN);
        Aws::String message = askQuestion("Enter a message text to publish.  ");
        request.SetMessage(message);

        if (filteringMessages && askYesNoQuestion(
                "Add an attribute to this message? (y/n) ")) {
            for (size_t i = 0; i < TONES.size(); ++i) {
                std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
            }
            int selection = askQuestionForIntRange(
                    "Enter a number for an attribute. ",
                    1, static_cast<int>(TONES.size()));
            Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
            messageAttributeValue.SetDataType("String");
```

```
            messageAttributeValue.SetStringValue(TONES[selection - 1]);
            request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
        }

        Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

        if (outcome.IsSuccess()) {
            std::cout << "Your message was successfully published." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::Publish. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 Publish를 참조하십시오.

CLI

AWS CLI

예제 1: 주제에 메시지를 게시하려면

다음 `publish` 예제에서는 지정된 Amazon SNS 주제에 지정된 메시지를 게시합니다. 메시지는 줄 바꿈을 포함할 수 있는 텍스트 파일에서 제공됩니다.

```
aws sns publish \
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic" \
    --message file://message.txt
```

`message.txt`의 콘텐츠:

```
Hello World
```

```
Second Line
```

출력:

```
{
    "MessageId": "123a45b6-7890-12c3-45d6-111122223333"
}
```

예제 2: 전화번호에 SMS 메시지를 게시하려면

다음 publish 예제에서는 Hello world! 메시지를 전화번호 +1-555-555-0100에 게시합니다.

```
aws sns publish \
    --message "Hello world!" \
    --phone-number +1-555-555-0100
```

출력:

```
{
    "MessageId": "123a45b6-7890-12c3-45d6-333322221111"
}
```

- API 세부 정보는 AWS CLI 명령 참조의 Publish를 참조하세요.

Go

SDK for Go V2

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// 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(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(context.TODO(), &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 Publish를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PublishTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <topicArn>

                Where:
                   message - The message text to send.
                   topicArn - The ARN of the topic to publish.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
```

```
        String topicArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTopic(snsClient, message, topicArn);
        snsClient.close();
    }

    public static void pubTopic(SnsClient snsClient, String message, String
 topicArn) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .topicArn(topicArn)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status is " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Publish를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

별도의 모듈에서 클라이언트를 생성하고 내보냅니다.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
import { PublishCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string | Record<string, any>} message - The message to send. Can be a
 plain string or an object
 *                                                  if you are using the `json`
 `MessageStructure`.
 * @param {string} topicArn - The ARN of the topic to which you would like to
 publish.
 */
export const publish = async (
  message = "Hello from SNS!",
  topicArn = "TOPIC_ARN",
) => {
  const response = await snsClient.send(
    new PublishCommand({
      Message: message,
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'e7f77526-e295-5325-9ee4-281a43ad1f05',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   MessageId: 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
```

```
  return response;
};
```

그룹, 복제, 속성 옵션을 사용하여 주제에 메시지를 게시하세요.

```
  async publishMessages() {
    const message = await this.prompter.input({
      message: MESSAGES.publishMessagePrompt,
    });

    let groupId, deduplicationId, choices;

    if (this.isFifo) {
      await this.logger.log(MESSAGES.groupIdNotice);
      groupId = await this.prompter.input({
        message: MESSAGES.groupIdPrompt,
      });

      if (this.autoDedup === false) {
        await this.logger.log(MESSAGES.deduplicationIdNotice);
        deduplicationId = await this.prompter.input({
          message: MESSAGES.deduplicationIdPrompt,
        });
      }

      choices = await this.prompter.checkbox({
        message: MESSAGES.messageAttributesPrompt,
        choices: toneChoices,
      });
    }

    await this.snsClient.send(
      new PublishCommand({
        TopicArn: this.topicArn,
        Message: message,
        ...(groupId
          ? {
              MessageGroupId: groupId,
            }
          : {}),
        ...(deduplicationId
          ? {
```

```
              MessageDeduplicationId: deduplicationId,
            }
          : {}),
        ...(choices
          ? {
              MessageAttributes: {
                tone: {
                  DataType: "String.Array",
                  StringValue: JSON.stringify(choices),
                },
              },
            }
          : {}),
      }),
    );

    const publishAnother = await this.prompter.confirm({
      message: MESSAGES.publishAnother,
    });

    if (publishAnother) {
      await this.publishMessages();
    }
  }
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 Publish를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun pubTopic(topicArnVal: String, messageVal: String) {
```

```
    val request = PublishRequest {
        message = messageVal
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 Publish를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Sends a message to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
```

```
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 자세한 정보는 AWS SDK for PHP 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for PHP API 참조의 Publish를 참조하십시오.

PowerShell

를 위한 도구 PowerShell

예 1: 이 예제에서는 MessageAttribute 선언된 단일 인라인으로 메시지를 게시하는 방법을 보여 줍니다.

```
Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -
Message "Hello" -MessageAttribute
 @{'City'=[Amazon.SimpleNotificationService.Model.MessageAttributeValue]@{DataType='Strin
 StringValue ='AnyCity'}}
```

예 2: 이 예에서는 여러 개를 미리 MessageAttributes 선언하여 메시지를 게시하는 방법을 보여 줍니다.

```
$cityAttributeValue = New-Object
 Amazon.SimpleNotificationService.Model.MessageAttributeValue
$cityAttributeValue.DataType = "String"
$cityAttributeValue.StringValue = "AnyCity"
```

```
$populationAttributeValue = New-Object
 Amazon.SimpleNotificationService.Model.MessageAttributeValue
$populationAttributeValue.DataType = "Number"
$populationAttributeValue.StringValue = "1250800"

$messageAttributes = New-Object System.Collections.Hashtable
$messageAttributes.Add("City", $cityAttributeValue)
$messageAttributes.Add("Population", $populationAttributeValue)

Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -
Message "Hello" -MessageAttribute $messageAttributes
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet에 게시 참조를 참조하십시오.

Python

SDK for Python(Boto3)

> **Note**
>
> 자세한 내용은 다음과 같습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

구독이 속성을 기준으로 필터링할 수 있도록 속성이 포함된 메시지를 게시합니다.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def publish_message(topic, message, attributes):
        """
        Publishes a message, with attributes, to a topic. Subscriptions can be
 filtered
        based on message attributes so that a subscription receives messages only
```

```
        when specified attributes are present.

        :param topic: The topic to publish to.
        :param message: The message to publish.
        :param attributes: The key-value attributes to attach to the message.
 Values
                            must be either `str` or `bytes`.
        :return: The ID of the message.
        """
        try:
            att_dict = {}
            for key, value in attributes.items():
                if isinstance(value, str):
                    att_dict[key] = {"DataType": "String", "StringValue": value}
                elif isinstance(value, bytes):
                    att_dict[key] = {"DataType": "Binary", "BinaryValue": value}
            response = topic.publish(Message=message, MessageAttributes=att_dict)
            message_id = response["MessageId"]
            logger.info(
                "Published message with attributes %s to topic %s.",
                attributes,
                topic.arn,
            )
        except ClientError:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise
        else:
            return message_id
```

구독자의 프로토콜에 따라 다른 양식을 사용하는 메시지를 게시합니다.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource
```

```
    @staticmethod
    def publish_multi_message(
        topic, subject, default_message, sms_message, email_message
    ):
        """
        Publishes a multi-format message to a topic. A multi-format message takes
        different forms based on the protocol of the subscriber. For example,
        an SMS subscriber might receive a short version of the message
        while an email subscriber could receive a longer version.

        :param topic: The topic to publish to.
        :param subject: The subject of the message.
        :param default_message: The default version of the message. This version
is
                                sent to subscribers that have protocols that are
not
                                otherwise specified in the structured message.
        :param sms_message: The version of the message sent to SMS subscribers.
        :param email_message: The version of the message sent to email
subscribers.
        :return: The ID of the message.
        """
        try:
            message = {
                "default": default_message,
                "sms": sms_message,
                "email": email_message,
            }
            response = topic.publish(
                Message=json.dumps(message), Subject=subject,
MessageStructure="json"
            )
            message_id = response["MessageId"]
            logger.info("Published multi-format message to topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise
        else:
            return message_id
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 Publish를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# Service class for sending messages using Amazon Simple Notification Service
 (SNS)
class SnsMessageSender
  # Initializes the SnsMessageSender with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Sends a message to a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param message [String] The message to send
  # @return [Boolean] true if message was successfully sent, false otherwise
  def send_message(topic_arn, message)
    @sns_client.publish(topic_arn: topic_arn, message: message)
    @logger.info("Message sent successfully to #{topic_arn}.")
    true
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error while sending the message: #{e.message}")
    false
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = "SNS_TOPIC_ARN" # Should be replaced with a real topic ARN
  message = "MESSAGE"         # Should be replaced with the actual message
 content
```

```
  sns_client = Aws::SNS::Client.new
  message_sender = SnsMessageSender.new(sns_client)

  @logger.info("Sending message.")
  unless message_sender.send_message(topic_arn, message)
    @logger.error("Message sending failed. Stopping program.")
    exit 1
  end
end
```

- 자세한 정보는 AWS SDK for Ruby 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for Ruby API 참조의 Publish를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
async fn subscribe_and_publish(
    client: &Client,
    topic_arn: &str,
    email_address: &str,
) -> Result<(), Error> {
    println!("Receiving on topic with ARN: `{}`", topic_arn);

    let rsp = client
        .subscribe()
        .topic_arn(topic_arn)
        .protocol("email")
        .endpoint(email_address)
        .send()
        .await?;

    println!("Added a subscription: {:?}", rsp);
```

```
    let rsp = client
        .publish()
        .topic_arn(topic_arn)
        .message("hello sns!")
        .send()
        .await?;

    println!("Published message: {:?}", rsp);

    Ok(())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 Publish을 참조하십시오.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        oo_result = lo_sns->publish(              " oo_result is returned for
testing purposes. "
          iv_topicarn = iv_topic_arn
          iv_message = iv_message
        ).
        MESSAGE 'Message published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- API에 대한 세부 정보는 SAP ABAP용AWS SDK API 참조의 Publish를 참조하세요.

Amazon SNS 및 Amazon S3 S3를 사용하여 대용량 메시지 게시

대용량 Amazon SNS 메시지를 게시하려면 Java용 Amazon SNS 확장 클라이언트 라이브러리 또는 Python용 Amazon SNS 확장 클라이언트 라이브러리를 사용할 수 있습니다. 이 라이브러리는 현재 최대값인 256KB(최대 2GB)보다 큰 메시지에 유용합니다. 모든 라이브러리는 실제 페이로드를 Amazon S3 버킷에 저장하고 저장된 Amazon S3 객체의 참조를 Amazon SNS 주제에 게시합니다. 구독한 Amazon SQS 대기열은 Java용 Amazon SQS 확장 클라이언트 라이브러리를 사용하여 Amazon S3에서 페이로드를 역참조하고 검색할 수 있습니다. Lambda와 같은 다른 엔드포인트는 AWS용 페이로드 오프로딩 Java 공통 라이브러리를 사용하여 페이로드를 역참조 및 검색할 수 있습니다.

> **Note**
>
> Amazon SNS 확장 클라이언트 라이브러리는 표준 주제 및 FIFO 주제 모두와 호환됩니다.

주제

- Java용 확장 클라이언트 라이브러리
- Python용 확장 클라이언트 라이브러리

Java용 확장 클라이언트 라이브러리

주제

- 필수 조건
- 메시지 스토리지 구성
- 예: Amazon S3에 저장된 페이로드로 Amazon SNS에 메시지 게시
- 기타 엔드포인트 프로토콜

필수 조건

다음은 Java용 Amazon SNS 확장 클라이언트 라이브러리 사용을 위한 사전 조건입니다.

- AWS SDK.

  이 페이지의 예시에서는 AWS Java SDK를 사용합니다. SDK를 설치 및 설정하려면 AWS SDK for Java 개발자 안내서의 Java용 AWS SDK 설정을 참조하세요.

- 그리고 AWS 계정 적절한 자격 증명이 필요합니다.

  계정을 AWS 계정생성하려면 AWS 홈 페이지로 이동한 다음 AWS 계정 생성을 선택합니다. 지침을 따릅니다.

  자격 증명에 대한 자세한 내용은 AWS SDK for Java 개발자 안내서의 AWS 자격 증명 및 개발 지역 설정을 참조하십시오.
- Java 8 이상.
- Java용 Amazon SNS 확장 클라이언트 라이브러리(Maven에서도 사용 가능).

메시지 스토리지 구성

Amazon SNS 확장 클라이언트 라이브러리는 메시지 저장 및 검색을 AWS 위해 페이로드 오프로딩 Java 공용 라이브러리를 사용합니다. 다음 Amazon S3 메시지 스토리지 옵션을 구성할 수 있습니다.

- 사용자 지정 메시지 크기 임계값 – 페이로드 및 속성이 이 크기를 초과하는 메시지는 Amazon S3에 자동으로 저장됩니다.
- `alwaysThroughS3` 플래그 — 이 값을 `true`로 설정하여 모든 메시지 페이로드가 Amazon S3에 강제로 저장되도록 합니다. 예:

```
SNSExtendedClientConfiguration snsExtendedClientConfiguration = new
SNSExtendedClientConfiguration() .withPayloadSupportEnabled(s3Client,
 BUCKET_NAME).withAlwaysThroughS3(true);
```

- 사용자 지정 KMS 키 – Amazon S3 버킷에서 서버 측 암호화에 사용할 키입니다.
- 버킷 이름 — 메시지 페이로드를 저장하기 위한 Amazon S3 버킷의 이름입니다.

예: Amazon S3에 저장된 페이로드로 Amazon SNS에 메시지 게시

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 샘플 주제 및 대기열을 만듭니다.
- 대기열에서 구독하여 주제의 메시지를 수신합니다.
- 테스트 메시지를 게시합니다.

메시지 페이로드는 Amazon S3에 저장되고 이에 대한 참조가 게시됩니다. Amazon SQS 확장 클라이언트는 메시지를 수신하는 데 사용됩니다.

Java 1.x용 SDK

> **Note**
>
> 자세한 내용은 다음과 같습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

대용량 메시지를 게시하려면 Java용 Amazon SNS 확장 클라이언트 라이브러리를 사용하세요. 보내는 메시지는 실제 메시지 내용이 포함된 Amazon S3 객체를 참조합니다.

```
import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.sns.AmazonSNS;
import com.amazonaws.services.sns.AmazonSNSClientBuilder;
import com.amazonaws.services.sns.model.CreateTopicRequest;
import com.amazonaws.services.sns.model.PublishRequest;
import com.amazonaws.services.sns.model.SetSubscriptionAttributesRequest;
import com.amazonaws.services.sns.util.Topics;
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.CreateQueueRequest;
import com.amazonaws.services.sqs.model.ReceiveMessageResult;
import software.amazon.sns.AmazonSNSExtendedClient;
import software.amazon.sns.SNSExtendedClientConfiguration;

public class Example {

        public static void main(String[] args) {
                final String BUCKET_NAME = "extended-client-bucket";
                final String TOPIC_NAME = "extended-client-topic";
                final String QUEUE_NAME = "extended-client-queue";
                final Regions region = Regions.DEFAULT_REGION;

                // Message threshold controls the maximum message size that will be
 allowed to
                // be published
```

```
                // through SNS using the extended client. Payload of messages
exceeding this
                // value will be stored in
                // S3. The default value of this parameter is 256 KB which is the
maximum
                // message size in SNS (and SQS).
                final int EXTENDED_STORAGE_MESSAGE_SIZE_THRESHOLD = 32;

                // Initialize SNS, SQS and S3 clients
                final AmazonSNS snsClient =
AmazonSNSClientBuilder.standard().withRegion(region).build();
                final AmazonSQS sqsClient =
AmazonSQSClientBuilder.standard().withRegion(region).build();
                final AmazonS3 s3Client =
AmazonS3ClientBuilder.standard().withRegion(region).build();

                // Create bucket, topic, queue and subscription
                s3Client.createBucket(BUCKET_NAME);
                final String topicArn = snsClient.createTopic(
                                new
CreateTopicRequest().withName(TOPIC_NAME)).getTopicArn();
                final String queueUrl = sqsClient.createQueue(
                                new
CreateQueueRequest().withQueueName(QUEUE_NAME)).getQueueUrl();
                final String subscriptionArn = Topics.subscribeQueue(
                                snsClient, sqsClient, topicArn, queueUrl);

                // To read message content stored in S3 transparently through SQS
extended
                // client,
                // set the RawMessageDelivery subscription attribute to TRUE
                final SetSubscriptionAttributesRequest subscriptionAttributesRequest
= new SetSubscriptionAttributesRequest();
                subscriptionAttributesRequest.setSubscriptionArn(subscriptionArn);

subscriptionAttributesRequest.setAttributeName("RawMessageDelivery");
                subscriptionAttributesRequest.setAttributeValue("TRUE");
                snsClient.setSubscriptionAttributes(subscriptionAttributesRequest);

                // Initialize SNS extended client
                // PayloadSizeThreshold triggers message content storage in S3 when
the
                // threshold is exceeded
                // To store all messages content in S3, use AlwaysThroughS3 flag
```

```
                final SNSExtendedClientConfiguration snsExtendedClientConfiguration
 = new SNSExtendedClientConfiguration()
                                .withPayloadSupportEnabled(s3Client, BUCKET_NAME)

 .withPayloadSizeThreshold(EXTENDED_STORAGE_MESSAGE_SIZE_THRESHOLD);
                final AmazonSNSExtendedClient snsExtendedClient = new
 AmazonSNSExtendedClient(snsClient,
                                snsExtendedClientConfiguration);

                // Publish message via SNS with storage in S3
                final String message = "This message is stored in S3 as it exceeds
 the threshold of 32 bytes set above.";
                snsExtendedClient.publish(topicArn, message);

                // Initialize SQS extended client
                final ExtendedClientConfiguration sqsExtendedClientConfiguration =
 new ExtendedClientConfiguration()
                                .withPayloadSupportEnabled(s3Client, BUCKET_NAME);
                final AmazonSQSExtendedClient sqsExtendedClient = new
 AmazonSQSExtendedClient(sqsClient,
                                sqsExtendedClientConfiguration);

                // Read the message from the queue
                final ReceiveMessageResult result =
 sqsExtendedClient.receiveMessage(queueUrl);
                System.out.println("Received message is " +
 result.getMessages().get(0).getBody());
        }
}
```

기타 엔드포인트 프로토콜

Amazon SNS 및 Amazon SQS 라이브러리는 모두 AWS용 페이로드 오프로딩 Java 공통 라이브러리를 사용하여 Amazon S3로 메시지 페이로드를 저장하고 검색합니다. 모든 Java 지원 엔드포인트(예: Java로 구현된 HTTPS 엔드포인트)는 동일한 라이브러리를 사용하여 메시지 콘텐츠를 역참조할 수 있습니다.

페이로드 오프로딩 Java 공용 라이브러리를 사용할 수 없는 엔드포인트는 Amazon S3에 저장된 페이로드와 함께 메시지를 AWS 게시할 수 있습니다. 다음은 위의 코드 예제에서 게시한 Amazon S3 참조의 예입니다.

```
[
  "software.amazon.payloadoffloading.PayloadS3Pointer",
  {
    "s3BucketName": "extended-client-bucket",
    "s3Key": "xxxx-xxxxx-xxxxx-xxxxxx"
  }
]
```

Python용 확장 클라이언트 라이브러리

주제

필수 조건

다음은 Python용 Amazon SNS 확장 클라이언트 라이브러리 사용을 위한 사전 조건입니다.

- SDK. AWS

  이 페이지의 예제는 AWS Python SDK Boto3를 사용합니다. SDK를 설치하고 설정하려면 Python용 AWS SDK 설명서를 참조하세요.
- 그리고 AWS 계정 적절한 자격 증명이 필요합니다.

  계정을 AWS 계정생성하려면 AWS 홈 페이지로 이동한 다음 AWS 계정 생성을 선택합니다. 지침을 따릅니다.

  자격 증명에 대한 자세한 내용은 Python용AWS SDK 개발자 안내서에서 자격 증명을 참조하세요.
- Python 3.x (또는 이후 버전) 및 pip.
- Python용 Amazon SNS 확장 클라이언트 라이브러리(PyPI에서도 사용 가능).

메시지 스토리지 구성

Boto3 Amazon SNS 클라이언트, 주제 및 PlatformEndpoint객체에서 다음 속성을 사용하여 Amazon S3 메시지 스토리지 옵션을 구성할 수 있습니다.

- `large_payload_support` — 대용량 메시지를 저장하는 Amazon S3 버킷 이름입니다.

- `message_size_threshold` — 대용량 메시지 버킷에 메시지를 저장하기 위한 임계값입니다. 값은 0 이하 또는 262144 이상이면 안 됩니다. 기본값은 262144입니다.
- `always_through_s3` — `True`인 경우 모든 메시지가 Amazon S3에 저장됩니다. 기본값은 `False`입니다.
- `s3` — Amazon S3에 `resource` 객체를 저장하는 데 사용되는 Boto3 Amazon S3 객체입니다. Amazon S3 리소스(예: 사용자 지정 Amazon S3 구성 또는 자격 증명)를 제어하려면 이 옵션을 사용하세요. 처음 사용할 때 이전에 설정하지 않은 경우 기본값은 `boto3.resource("s3")`입니다.

예: Amazon S3에 저장된 페이로드로 Amazon SNS에 메시지 게시

다음 코드 예제에서는 다음과 같은 작업을 수행하는 방법을 보여줍니다.

- 샘플 Amazon SNS 주제 및 Amazon SQS 대기열을 생성합니다.
- 대기열에서 구독하여 주제의 메시지를 수신합니다.
- 테스트 메시지를 게시합니다.
- 메시지 페이로드는 Amazon S3에 저장되고 이에 대한 참조가 게시됩니다.
- 대기열에 게시된 메시지를 Amazon S3에서 검색된 원본 메시지와 함께 인쇄합니다.

대용량 메시지를 게시하려면 Python용 Amazon SNS 확장 클라이언트 라이브러리를 사용하세요. 보내는 메시지는 실제 메시지 내용이 포함된 Amazon S3 객체를 참조합니다.

```
import boto3
import sns_extended_client
from json import loads

s3_extended_payload_bucket = "extended-client-bucket-store"
TOPIC_NAME = "---TOPIC-NAME---"
QUEUE_NAME = "---QUEUE-NAME---"

# Create an helper to fetch message from S3
def get_msg_from_s3(body):
    json_msg = loads(body)
    s3_client = boto3.client("s3")
    s3_object = s3_client.get_object(
        Bucket=json_msg[1].get("s3BucketName"), Key=json_msg[1].get("s3Key")
    )
    msg = s3_object.get("Body").read().decode()
    return msg
```

```
# Create an helper to fetch and print message SQS queue and S3
def fetch_and_print_from_sqs(sqs, queue_url):
    """Handy Helper to fetch and print message from SQS queue and S3"""
    message = sqs.receive_message(
        QueueUrl=queue_url, MessageAttributeNames=["All"], MaxNumberOfMessages=1
    ).get("Messages")[0]
    message_body = message.get("Body")
    print("Published Message: {}".format(message_body))
    print("Message Stored in S3 Bucket is: {}\n".format(get_msg_from_s3(message_body)))

# Initialize the SNS client and create SNS Topic
sns_extended_client = boto3.client("sns", region_name="us-east-1")
create_topic_response = sns_extended_client.create_topic(Name=TOPIC_NAME)
demo_topic_arn = create_topic_response.get("TopicArn")

# Create and subscribe an SQS queue to the SNS client
sqs = boto3.client("sqs")
demo_queue_url = sqs.create_queue(QueueName=QUEUE_NAME).get("QueueUrl")
demo_queue_arn = sqs.get_queue_attributes(QueueUrl=demo_queue_url,
AttributeNames=["QueueArn"])["Attributes"].get("QueueArn")
# Set the RawMessageDelivery subscription attribute to TRUE
sns_extended_client.subscribe(TopicArn=demo_topic_arn, Protocol="sqs",
Endpoint=demo_queue_arn, Attributes={"RawMessageDelivery":"true"})

sns_extended_client.large_payload_support = s3_extended_payload_bucket

# To store all messages content in S3, set always_through_s3 to True
# In the example, we set message size threshold as 32 bytes, adjust this threshold as
 per your usecase
# Message will only be uploaded to S3 when its payload size exceeded threshold
sns_extended_client.message_size_threshold = 32
sns_extended_client.publish(
    TopicArn=demo_topic_arn,
    Message="This message should be published to S3 as it exceeds the
 message_size_threshold limit",
)
# Print message stored in s3
fetch_and_print_from_sqs(sqs, demo_queue_url)
```

출력

```
Published Message:
[
    "software.amazon.payloadoffloading.PayloadS3Pointer",
    {
        "s3BucketName": "extended-client-bucket-store",
        "s3Key": "xxxx-xxxxx-xxxxx-xxxxxx"
    }
]
Message Stored in S3 Bucket is: This message should be published to S3 as it exceeds
 the message_size_threshold limit
```

Amazon SNS 메시지 속성

Amazon SNS는 메시지에 대한 구조적 메타데이터 항목(예: 타임스탬프, 지형 정보 데이터, 서명 및 식별자)을 제공하도록 해주는 메시지 속성의 전송을 지원합니다. SQS 구독의 경우 원시 메시지 전송이 사용된다면 최대 10개의 메시지 속성을 보낼 수 있습니다. 10개가 넘는 메시지 속성을 보내려면 원시 메시지 전송을 사용 중지해야 합니다. 원시 메시지 전송이 활성화된 Amazon SQS 구독으로 전달되는 메시지 속성이 10개를 초과하는 메시지는 클라이언트 측 오류로 삭제됩니다.

메시지 속성은 선택 사항이며 메시지 본문과 분리되지만 함께 전송됩니다. 수신자는 이 정보를 사용하여 메시지 본문을 먼저 처리할 필요 없이 메시지를 처리하는 방법을 결정할 수 있습니다.

AWS Management Console 또는 AWS SDK for Java를 사용하여 속성이 있는 메시지를 전송하는 방법에 대한 자세한 내용은 AWS Management Console을 사용하여 Amazon SNS 주제에 메시지를 게시하려면 자습서를 참조하세요.

> **Note**
>
> 메시지 구조가 문자열이고 JSON이 아닌 경우에만 메시지 속성이 전송됩니다.

메시지 속성을 사용하여 모바일 엔드포인트에 대한 푸시 알림 메시지를 구성할 수도 있습니다. 이 시나리오에서 메시지 속성은 푸시 알림 메시지를 구성하는 데만 사용됩니다. 이 속성은 메시지 속성이 있는 메시지를 Amazon SQS 엔드포인트에 전송할 때와 같이 엔드포인트에 제공되지 않습니다.

메시지 속성을 사용하여 구독 필터 정책으로 메시지를 필터링할 수 있도록 만들 수 있습니다. 주제 구독에 필터 정책을 적용할 수 있습니다. `MessageAttributes`(기본)로 설정된 필터 정책 범위로 필터 정책이 적용되면 구독은 정책이 수락하는 속성을 가진 메시지만 수신합니다. 자세한 내용은 Amazon SNS 메시지 필터링 섹션을 참조하세요.

> **Note**
>
> 메시지 속성을 필터링에 사용하는 경우 값은 유효한 JSON 문자열이어야 합니다. 이렇게 하면 메시지가 메시지 속성 필터링이 활성화된 구독으로 전달됩니다.

메시지 속성 항목 및 유효성 검사

각 메시지 속성은 다음 항목으로 구성됩니다.

- Name – 메시지 속성 이름에는 A-Z, a-z, 0-9, 밑줄(_), 하이픈(-) 및 마침표(.)와 같은 문자가 포함될 수 있습니다. 이름은 마침표로 시작하거나 끝나서는 안 되며 연속 마침표가 있으면 안 됩니다. 이름은 대소문자를 구별하며 메시지에 대한 모든 속성 이름 중에서 고유해야 합니다. 이름은 최대 256자일 수 있습니다. 이름은 `AWS.` 또는 `Amazon.`(또는 모든 대소문자 변형)으로 시작할 수 없습니다. 이러한 접두사는 Amazon Web Services에서 사용하도록 예약되어 있기 때문입니다.
- Type – 지원되는 메시지 속성 데이터 형식은 `String`, `String.Array`, `Number` 및 `Binary`입니다. 데이터 형식은 내용에 대해 메시지 본문과 동일한 제한 사항이 있습니다. 데이터 형식은 대소문자를 구별하며 최대 256바이트일 수 있습니다. 자세한 정보는 메시지 속성 데이터 형식 및 유효성 검사 섹션을 참조하세요.
- Value – 사용자가 지정한 메시지 속성 값입니다. 문자열 데이터 형식의 경우 값 속성은 내용에 대해 메시지 본문과 동일한 제한 사항이 있습니다. 자세한 정보는 Amazon Simple Notification Service API 참조의 게시 작업을 참조하세요.

이름, 형식 및 값이 비어 있거나 널이면 안 됩니다. 또한 메시지 본문도 비어 있거나 널이면 안 됩니다. 이름, 형식 및 값을 포함하여 메시지 속성의 모든 부분은 256KB인 메시지 크기 제한에 포함됩니다.

메시지 속성 데이터 형식 및 유효성 검사

메시지 속성 데이터 형식은 메시지 속성 값이 Amazon SNS에서 처리되는 방식을 식별합니다. 예를 들어, 형식이 숫자이면 Amazon SNS는 값이 숫자인지 확인합니다.

Amazon SNS는 명시된 경우를 제외하고 모든 엔드포인트에 대해 다음 논리적 데이터 유형을 지원합니다.

- String – 문자열은 UTF-8 이진 인코딩을 사용하는 유니코드입니다. 코드 값 목록은 http://en.wikipedia.org/wiki/ASCII#ASCII_printable_characters를 참조하세요.

> **Note**
>
> 대리 값은 메시지 속성에서 지원되지 않습니다. 예를 들어, 이모티콘을 나타내는 대리 값을 사용하면 "`Invalid attribute value was passed in for message attribute`"와 같은 오류가 발생합니다.

- String.Array – 여러 값이 포함될 수 있는 문자열 형식의 배열입니다. 값은 문자열, 숫자 또는 키워드 `true`, `false` 및 `null`일 수 있습니다. 숫자 또는 부울 형식의 String.Array에는 따옴표가 필요하지 않습니다. String.Array 값이 여러 개인 경우 쉼표로 구분합니다.

  이 데이터 형식은 AWS Lambda 구독에 대해 지원되지 않습니다. Lambda 엔드포인트에 대해 이 데이터 유형을 지정하면 Amazon SNS가 Lambda에 전달하는 JSON 페이로드의 `String` 데이터 유형으로 전달됩니다.
- Number – 숫자는 양수 또는 음수 정수이거나 부동 소수점 숫자입니다. 숫자는 정수, 부동 소수점 수 및 실수가 일반적으로 지원하는 가능한 대부분의 값을 포함할 수 있는 충분한 범위와 정밀도를 갖추고 있습니다. 숫자 값은 -10^9에서 10^9까지 가능하며, 정확도는 소수점 이하 5자리입니다. 앞과 끝의 0은 잘립니다.

  이 데이터 형식은 AWS Lambda 구독에 대해 지원되지 않습니다. Lambda 엔드포인트에 대해 이 데이터 유형을 지정하면 Amazon SNS가 Lambda에 전달하는 JSON 페이로드의 `String` 데이터 유형으로 전달됩니다.
- Binary – 이전 형식 속성은 모든 이진 데이터를 저장할 수 있습니다(예: 압축된 데이터, 암호화된 데이터 또는 이미지).

모바일 푸시 알림에 예약된 메시지 속성

다음 표에서는 푸시 알림 메시지를 구성하는 데 사용할 수 있는 모바일 푸시 알림 서비스에 예약된 메시지 속성을 나열합니다.

푸시 알림 서비스	예약된 메시지 속성
ADM	`AWS.SNS.MOBILE.ADM.TTL`
APN[1]	`AWS.SNS.MOBILE.APNS_MDM.TTL`
	`AWS.SNS.MOBILE.APNS_MDM_SANDBOX.TTL`

푸시 알림 서비스	예약된 메시지 속성
	AWS.SNS.MOBILE.APNS_PASSBOOK.TTL
	AWS.SNS.MOBILE.APNS_PASSBOOK_SANDBOX.TTL
	AWS.SNS.MOBILE.APNS_SANDBOX.TTL
	AWS.SNS.MOBILE.APNS_VOIP.TTL
	AWS.SNS.MOBILE.APNS_VOIP_SANDBOX.TTL
	AWS.SNS.MOBILE.APNS.COLLAPSE_ID
	AWS.SNS.MOBILE.APNS.PRIORITY
	AWS.SNS.MOBILE.APNS.PUSH_TYPE
	AWS.SNS.MOBILE.APNS.TOPIC
	AWS.SNS.MOBILE.APNS.TTL
Baidu	AWS.SNS.MOBILE.BAIDU.DeployStatus
	AWS.SNS.MOBILE.BAIDU.MessageKey
	AWS.SNS.MOBILE.BAIDU.MessageType
	AWS.SNS.MOBILE.BAIDU.TTL
FCM	AWS.SNS.MOBILE.FCM.TTL
	AWS.SNS.MOBILE.GCM.TTL
macOS	AWS.SNS.MOBILE.MACOS_SANDBOX.TTL
	AWS.SNS.MOBILE.MACOS.TTL
MPNS	AWS.SNS.MOBILE.MPNS.NotificationClass
	AWS.SNS.MOBILE.MPNS.TTL

푸시 알림 서비스	예약된 메시지 속성
	AWS.SNS.MOBILE.MPNS.Type
WNS	AWS.SNS.MOBILE.WNS.CachePolicy
	AWS.SNS.MOBILE.WNS.Group
	AWS.SNS.MOBILE.WNS.Match
	AWS.SNS.MOBILE.WNS.SuppressPopup
	AWS.SNS.MOBILE.WNS.Tag
	AWS.SNS.MOBILE.WNS.TTL
	AWS.SNS.MOBILE.WNS.Type

[1] 메시지 속성이 요구 사항을 충족하지 않는 경우 Apple은 Amazon SNS 알림을 거부합니다. 자세한 내용은 Apple 개발자 웹 사이트에서 APN에 알림 요청 보내기를 참조하세요.

Amazon SNS 메시지 배치 처리

메시지 일괄 처리란 무엇입니까?

개별 Publish API 요청에서 표준 또는 FIFO 주제에 메시지를 게시하는 대신 Amazon SNS PublishBatch API를 사용하여 단일 API 요청에서 최대 10개의 메시지를 게시할 수 있습니다. 메시지를 배치로 전송하면 Amazon SNS를 통해 분산 애플리케이션 연결(A2A 메시징) 또는 사람에게 알림 전송(A2P 메시징)과 관련된 비용을 최대 10배까지 줄일 수 있습니다. Amazon SNS에는 사용자가 운영 중인 리전에 따라 주제에 초당 게시할 수 있는 메시지 수에 대한 할당량이 있습니다. API 할당량에 대한 자세한 내용은 AWS 일반 참조 가이드의 Amazon SNS 엔드포인트 및 할당량 페이지를 참조하세요.

> **Note**
>
> 단일 PublishBatch API 요청에서 전송하는 전체 메시지의 총 크기 합계는 262,144바이트(256KB)를 초과할 수 없습니다.
> 이 PublishBatch API는 IAM 정책에 대해 동일한 Publish API 작업을 사용합니다.

메시지 배치 처리는 어떻게 작동합니까?

PublishBatch API를 사용하여 메시지를 게시하는 것은 Publish API를 사용하여 메시지를 게시하는 것과 유사합니다. 주된 차이점은 PublishBatch API 요청 내의 각 메시지에 고유한 배치 ID(최대 80자)가 할당되어야 한다는 것입니다. 이러한 방식으로 Amazon SNS는 배치 내의 모든 메시지에 대해 개별 API 응답을 반환하여 각 메시지가 게시되었는지 또는 오류가 발생했는지 확인할 수 있습니다. FIFO 주제에 게시되는 메시지의 경우 고유한 배치 ID 할당을 포함하는 것 외에도 각 개별 메시지에 대해 MessageDeduplicationID 및 MessageGroupId를 포함해야 합니다.

예시

표준 주제에 10개의 메시지 배치 게시

```
// Imports
import com.amazonaws.services.sns.AmazonSNS;
import com.amazonaws.services.sns.model.PublishBatchRequest;
import com.amazonaws.services.sns.model.PublishBatchRequestEntry;
import com.amazonaws.services.sns.model.PublishBatchResult;
import com.amazonaws.services.sns.model.AmazonSNSException;
import java.util.List;
import java.util.stream.Collectors;

// Code
private static final int MAX_BATCH_SIZE = 10;

public static void publishBatchToTopic(AmazonSNS snsClient, String topicArn) {
    try {
        // Create the batch entries to send
        List<PublishBatchRequestEntry> entries = IntStream.range(0, MAX_BATCH_SIZE)
                .mapToObj(i -> new PublishBatchRequestEntry()
                        .withId("id" + i)
                        .withMessage("message" + i))
                .collect(Collectors.toList());

        // Create the batch request
        PublishBatchRequest request = new PublishBatchRequest()
                .withTopicArn(topicArn)
                .withPublishBatchRequestEntries(entries);

        // Publish the batch request
        PublishBatchResult publishBatchResult = snsClient.publishBatch(request);
```

```
        // Handle the successfully sent messages
        publishBatchResult.getSuccessful().forEach(publishBatchResultEntry -> {
            System.out.println("Batch Id for successful message: " +
 publishBatchResultEntry.getId());
            System.out.println("Message Id for successful message: " +
 publishBatchResultEntry.getMessageId());
        });

        // Handle the failed messages
        publishBatchResult.getFailed().forEach(batchResultErrorEntry -> {
            System.out.println("Batch Id for failed message: " +
 batchResultErrorEntry.getId());
            System.out.println("Error Code for failed message: " +
 batchResultErrorEntry.getCode());
            System.out.println("Sender Fault for failed message: " +
 batchResultErrorEntry.getSenderFault());
            System.out.println("Failure Message for failed message: " +
 batchResultErrorEntry.getMessage());
        });
    } catch (AmazonSNSException e) {
        // Handle any exceptions from the request
        System.err.println(e.getMessage());
        System.exit(1);
    }
}
```

FIFO 주제에 10개의 메시지 배치 게시

```
// Imports
import com.amazonaws.services.sns.AmazonSNS;
import com.amazonaws.services.sns.model.PublishBatchRequest;
import com.amazonaws.services.sns.model.PublishBatchRequestEntry;
import com.amazonaws.services.sns.model.PublishBatchResult;
import com.amazonaws.services.sns.model.AmazonSNSException;
import java.util.List;
import java.util.stream.Collectors;

// Code
private static final int MAX_BATCH_SIZE = 10;
```

```
public static void publishBatchToFifoTopic(AmazonSNS snsClient, String topicArn) {
    try {
        // Create the batch entries to send
        List<PublishBatchRequestEntry> entries = IntStream.range(0, MAX_BATCH_SIZE)
                .mapToObj(i -> new PublishBatchRequestEntry()
                        .withId("id" + i)
                        .withMessage("message" + i)
                        .withMessageGroupId("groupId")
                        .withMessageDeduplicationId("deduplicationId" + i))
                .collect(Collectors.toList());

        // Create the batch request
        PublishBatchRequest request = new PublishBatchRequest()
                .withTopicArn(topicArn)
                .withPublishBatchRequestEntries(entries);

        // Publish the batch request
        PublishBatchResult publishBatchResult = snsClient.publishBatch(request);

        // Handle the successfully sent messages
        publishBatchResult.getSuccessful().forEach(publishBatchResultEntry -> {
            System.out.println("Batch Id for successful message: " +
 publishBatchResultEntry.getId());
            System.out.println("Message Id for successful message: " +
 publishBatchResultEntry.getMessageId());
            System.out.println("SequenceNumber for successful message: " +
 publishBatchResultEntry.getSequenceNumber());
        });

        // Handle the failed messages
        publishBatchResult.getFailed().forEach(batchResultErrorEntry -> {
            System.out.println("Batch Id for failed message: " +
 batchResultErrorEntry.getId());
            System.out.println("Error Code for failed message: " +
 batchResultErrorEntry.getCode());
            System.out.println("Sender Fault for failed message: " +
 batchResultErrorEntry.getSenderFault());
            System.out.println("Failure Message for failed message: " +
 batchResultErrorEntry.getMessage());
        });

    } catch (AmazonSNSException e) {
        // Handle any exceptions from the request
        System.err.println(e.getMessage());
```

```
            System.exit(1);
        }
    }
```

Amazon SNS 메시지 필터링

기본적으로 Amazon SNS 주제 구독자는 주제에 게시된 모든 메시지를 수신합니다. 메시지의 하위 세트만 수신하려면 구독자는 주제 구독에 필터 정책을 할당해야 합니다.

필터링 정책은 구독자가 어떤 메시지를 수신할지 정의하는 속성을 포함하는 JSON 객체입니다. Amazon SNS는 구독에 설정한 필터 정책 범위에 따라 메시지 속성 또는 메시지 본문에 적용되는 정책을 지원합니다. 메시지 본문의 필터 정책에서는 메시지 페이로드가 올바른 형식의 JSON 객체라고 가정합니다.

구독에 필터 정책이 없는 경우 구독자는 해당 주제에 게시된 모든 메시지를 받습니다. 메시지를 주제에 게시하면 Amazon SNS는 메시지 속성 또는 본문을 각 주제의 구독에 대한 필터 정책에 있는 속성과 비교합니다. 일치하는 속성 또는 메시지 본문 속성이 있을 경우 Amazon SNS는 구독자에게 메시지를 전송합니다. 그렇지 않으면 Amazon SNS는 해당 구독자에게 메시지를 보내지 않습니다.

자세한 정보는 주제에 게시된 메시지 필터링을 참조하세요.

주제

Amazon SNS 구독 필터 정책 범위

`FilterPolicyScope` 구독 속성을 사용하면 다음 값 중 하나를 설정하여 필터링 범위를 선택할 수 있습니다.

- `MessageAttributes` - 필터 정책이 메시지 속성에 적용됩니다. 이 값이 기본값입니다.
- `MessageBody` - 필터 정책이 메시지 본문에 적용됩니다.

> **Note**
>
> 기존 필터 정책에 대해 정의된 필터 정책 범위가 없는 경우 범위는 `MessageAttributes`로 기본 설정됩니다.

Amazon SNS 구독 필터 정책

구독 필터 정책을 사용하면 속성 이름을 지정하고 각 속성 이름에 값 목록을 지정할 수 있습니다. 자세한 정보는 Amazon SNS 메시지 필터링을 참조하세요.

Amazon SNS는 구독 필터 정책을 기준으로 메시지 속성 또는 메시지 본문 속성을 평가할 때 정책에 지정되지 않은 메시지 속성이나 메시지 본문 속성을 무시합니다.

> **Important**
>
> AWS IAM 및 Amazon SNS와 같은 서비스는 최종 일관성이라는 분산 컴퓨팅 모델을 사용합니다. 구독 필터 정책의 추가 또는 변경 사항이 완전히 적용되려면 최대 15분이 소요됩니다.

구독은 다음 조건에서 메시지를 수락합니다.

- 필터 정책 범위를 `MessageAttributes`로 설정하면 필터 정책의 각 속성 이름을 메시지 속성 이름과 일치시킵니다. 필터 정책에서 일치하는 각 속성 이름에 대해 하나 이상의 속성 값을 메시지 속성 값과 일치시킵니다.
- 필터 정책 범위를 `MessageBody`로 설정하면 필터 정책의 각 속성 이름을 메시지 본문 속성 이름과 일치시킵니다. 필터 정책에서 일치하는 각 속성 이름에 대해 하나 이상의 속성 값을 메시지 본문 속성 값과 일치시킵니다.

Amazon SNS에서는 현재 다음과 같은 필터 연산자를 지원합니다.

- AND 로직
- OR 로직
- OR 연산자
- 키 일치
- 숫자 값 정확한 일치
- 숫자 값 anything-but 일치
- 숫자 값 범위 일치
- 문자열 값 정확한 일치
- 문자열 값 anything-but 일치
- 접두사와 anything-but 연산자를 사용한 문자열 일치
- 문자열 값 equals-ignore case

- 문자열 값 IP 주소 일치
- 문자열 값 접두사 일치
- 문자열 값 접미사 일치

예제 필터 정책

다음 예제에서는 고객 트랜잭션을 처리하는 Amazon SNS 주제에서 전송된 메시지 페이로드를 보여 줍니다.

첫 번째 예에서 `MessageAttributes` 필드에는 트랜잭션을 설명하는 속성이 포함됩니다.

- 고객의 관심사
- 스토어 이름
- 이벤트 상태
- 구매 가격(USD)

이 메시지에는 `MessageAttributes` 필드가 포함되어 있으므로, 구독의 `FilterPolicyScope`가 `MessageAttributes`로 설정되어 있는 한 `FilterPolicy`가 설정된 주제 구독은 메시지를 선택적으로 수락 또는 거부할 수 있습니다. 메시지에 속성을 적용하는 방법에 대한 자세한 내용은 Amazon SNS 메시지 속성에서 확인하세요.

```
{
   "Type": "Notification",
   "MessageId": "a1b2c34d-567e-8f90-g1h2-i345j67klmn8",
   "TopicArn": "arn:aws:sns:us-east-2:123456789012:MyTopic",
   "Message": "message-body-with-transaction-details",
   "Timestamp": "2019-11-03T23:28:01.631Z",
   "SignatureVersion": "4",
   "Signature": "signature",
   "UnsubscribeURL": "unsubscribe-url",
   "MessageAttributes": {
      "customer_interests": {
         "Type": "String.Array",
         "Value": "[\"soccer\", \"rugby\", \"hockey\"]"
      },
      "store": {
         "Type": "String",
         "Value":"example_corp"
```

```
      },
      "event": {
         "Type": "String",
         "Value": "order_placed"
      },
      "price_usd": {
         "Type": "Number",
         "Value": "210.75"
      }
   }
}
```

다음 예에서는 Message 필드(메시지 페이로드 또는 메시지 본문이라고도 함)에 포함된 동일한 속성을 보여줍니다. FilterPolicy를 포함하는 모든 주제 구독은 구독의 FilterPolicyScope가 MessageBody로 설정되어 있는 한 메시지를 선택적으로 수락하거나 거부할 수 있습니다.

```
{
"Type": "Notification",
   "MessageId": "a1b2c34d-567e-8f90-g1h2-i345j67klmn8",
   "TopicArn": "arn:aws:sns:us-east-2:123456789012:MyTopic",
   "Message": "{
      \"customer_interests\": [\"soccer\", \"rugby\", \"hockey\"],
      \"store\": \"example_corp\",
      \"event\":\"order_placed\",
      \"price_usd\":210.75
   }",
   "Timestamp": "2019-11-03T23:28:01.631Z",
   "SignatureVersion": "4",
   "Signature": "signature",
   "UnsubscribeURL": "unsubscribe-url"
}
```

다음 필터 정책은 속성 이름과 값을 기준으로 메시지를 수락하거나 거부합니다.

예제 메시지를 수락하는 정책

다음 구독 필터 정책의 속성이 예제 메시지에 지정된 속성과 일치합니다. 참고로 MessageAttributes 또는 MessageBody로 설정되었는지와 관계없이 FilterPolicyScope에 대해 동일한 필터 정책이 적용됩니다. 각 구독자는 주제에서 받는 메시지의 구성에 따라 필터링 범위를 선택합니다.

이 정책의 속성 중 하나가 메시지에 지정된 속성과 일치하지 않으면 정책은 해당 메시지를 거부합니다.

```
{
   "store": ["example_corp"],
   "event": [{"anything-but": "order_cancelled"}],
   "customer_interests": [
      "rugby",
      "football",
      "baseball"
   ],
   "price_usd": [{"numeric": [">=", 100]}]
}
```

예제 메시지를 거부하는 정책

다음 구독 필터 정책의 속성과 예제 메시지에 지정된 속성이 서로 여러 가지가 불일치합니다. 예를 들어 `encrypted` 속성 이름이 메시지 속성에 없으므로 이 정책 속성은 할당된 값과 관계없이 메시지를 거부합니다.

불일치가 하나라도 있을 경우 정책은 해당 메시지를 거부합니다.

```
{
   "store": ["example_corp"],
   "event": ["order_cancelled"],
   "encrypted": [false],
   "customer_interests": [
      "basketball",
      "baseball"
   ]
}
```

필터 정책 제약

Amazon SNS 구독에 대한 필터 정책을 생성할 때는 정책의 키가 어떻게 계산되는지 이해하는 것이 중요합니다. 염두에 두어야 할 주요 측면은 다음과 같습니다.

1. 부모 키 — 부모 키는 필터 정책의 최상위 키입니다. 이러한 키는 값이나 제약 조건을 지정하는 데 사용됩니다.
2. 속성 이름 - 상위 키는 필터 정책에서 속성 이름으로 간주됩니다. 이러한 키에 지정하는 값이나 제약 조건은 메시지 페이로드의 해당 속성에 적용됩니다.
3. 유효한 값 — 부모 키에 지정된 값은 문자열, 문자열 배열 또는 숫자여야 합니다. 값이 객체 (예: JSON 객체) 인 경우 필터 정책에서 유효한 키로 간주되지 않습니다.

다음 필터 정책 예시를 살펴보겠습니다.

```
{
  "state": ["SUCCESS"],
  "severity": [{ "exists": true }],
  "message": [{ "exists": true }],
  "finding": {
    "standard_control": [{ "exists": true }],
    "region": [{ "exists": true }],
    "account": [{ "exists": true }]
  }
}
```

이 예시에서는 다음 키가 필터 정책의 일부로 계산됩니다.

- `state`
- `severity`
- `message`
- `standard_control`
- `region`
- `account`

키 검색은 문자열, 문자열 배열 또는 숫자가 아닌 JSON 객체를 값으로 포함하므로 집계되지 않습니다.

또 다른 예시:

```
{
  "key_a": {
    "key_b": {
      "key_c": {
        "key_d": ["value_one", "value_two", "value_three", "value_four"]
      }
    },
    "key_e": {
      "key_f": ["value_one", "value_two", "value_three"]
    }
  }
}
```

이 경우 key_d 키에만 문자열 또는 문자열 배열의 값이 할당되므로 필터 정책의 일부로 계산됩니다. key_f 부모 키 key_akey_b, 및 key_c 는 값으로 중첩된 JSON 객체를 포함하므로 계산되지 않습니다.

주제

- 필터 정책 제약 조건
- 속성 기반 필터링에 대한 정책 제약 조건
- 페이로드 기반 필터링에 대한 정책 제약 조건

필터 정책 제약 조건

- 문자열 매칭 — 필터 정책의 문자열 매칭의 경우 비교는 대소문자를 구분합니다.
- 숫자 일치 — 숫자 일치의 경우 값 범위는 $-10^{9} \sim 10^{9}$ (-10억에서 10억) 사이이며 소수점 이하 5자리 정확도를 가집니다.
- 필터 정책 복잡성 - 필터 정책이 복잡하기 때문에 총 값 조합은 150을 초과할 수 없습니다. 총 조합을 계산하려면 필터 정책에 있는 각 배열의 값 수를 곱하십시오.

다음 예제 정책을 고려해 보십시오.

```
{
   "key_a": ["value_one", "value_two", "value_three"],
   "key_b": ["value_one"],
   "key_c": ["value_one", "value_two"]
}
```

이 정책에서:

- 첫 번째 배열에는 3개의 값이 있습니다.
- 두 번째 배열의 값은 1개입니다.
- 세 번째 배열에는 두 개의 값이 있습니다.

따라서 전체 조합은 다음과 같습니다.

- 3 x 1 x 2 = 6

필터 정책 구문

필터 정책의 JSON에는 다음이 포함될 수 있습니다.

- 인용 부호로 묶인 문자열
- 숫자
- 인용 부호 없는 키워드 `true`, `false` 및 `null`

Amazon SNS API를 사용할 때는 필터 정책의 JSON을 유효한 UTF-8 문자열로 전달해야 합니다.

필터 정책 제한

- 필터 정책의 최대 크기는 256KB입니다.
- 기본적으로 주제당 최대 200개의 필터 정책과 AWS 계정당 10,000개의 필터 정책을 사용할 수 있습니다.
- 이 정책 한도는 API로 Amazon SQS 대기열 구독을 생성하는 것을 중단하지 않습니다. `Subscribe` 하지만 `Subscribe` API 호출(또는 `SetSubscriptionAttributes` API 호출)에 필터 정책을 연결하면 실패합니다.
- 할당량 증가를 요청하려면 AWS Service Quotas를 사용합니다.

속성 기반 필터링에 대한 정책 제약 조건

- 속성 기반 필터링은 기본 옵션으로, 구독에서 `FilterPolicyScope`는 `MessageAttributes`로 설정되어 있습니다.
- Amazon SNS는 속성 기반 필터링을 위한 중첩된 필터 정책을 허용하지 않습니다.
- Amazon SNS는 다음 데이터 형식을 갖는 메시지 속성만 정책 속성과 비교합니다.
  - `String`
  - `String.Array`

    > **Important**
    >
    > 배열의 객체를 전달하면 속성 기반 필터링에서 지원되지 않는 중첩으로 인해 예상치 못한 결과가 발생할 수 있으므로 배열의 객체를 전달하는 것은 권장되지 않습니다. 중첩된 정책에 페이로드 기반 필터링을 사용합니다.

  - `Number`
- Amazon SNS는 데이터 형식이 `Binary`인 메시지 속성을 무시합니다.

- 각 필터 정책에는 최대 5개의 속성 이름이 있을 수 있습니다.

페이로드 기반 필터링에 대한 정책 제약 조건

Amazon SNS는 페이로드 기반 필터링에 대한 중첩된 필터 정책을 수락합니다. 필터 정책의 총 값 조합을 계산하려면 각 중첩 배열의 값 수를 곱하십시오.

다음 예제 정책을 고려해 보십시오.

```
{
"key_a": {
    "key_b": {
        "key_c": ["value_one", "value_two", "value_three", "value_four"]
        }
    },
"key_d": {
    "key_e": ["value_one", "value_two", "value_three"]
    }
}
```

이 정책에서:

- 첫 번째 배열에는 3단계 중첩 키에 네 개의 값이 있습니다.
- 두 번째 키에는 2단계 중첩 키에 세 개의 값이 있습니다.

따라서 전체 조합은 다음과 같습니다.

- 4 x 3 x 3 x 2 = 72

정책 한도

필터 정책에는 최대 5개의 부모 키 (최상위 키) 가 있을 수 있습니다. 중첩된 정책의 경우 상위 키만 키 제한 5개에 포함됩니다.

숫자 범위

필터 정책에서의 숫자 일치의 경우 값의 범위는 -10 9 ~ 10^{9} (-10억~10억) 이며 소수점 이하 5자리 정확도를 가집니다.

페이로드 기반 필터링으로 전환

속성 기반(기본값)에서 페이로드 기반 필터링으로 전환하려면 구독에서 `FilterPolicyScope`를 `MessageBody`로 설정해야 합니다.

AND/OR 로직

AND/OR 로직을 포함하는 연산을 사용하여 메시지 속성 또는 메시지 본문 속성을 일치시킬 수 있습니다.

주제

AND 로직

여러 속성 이름을 사용하여 AND 로직을 적용할 수 있습니다.

다음 정책을 살펴보겠습니다.

```
{
  "customer_interests": ["rugby"],
  "price_usd": [{"numeric": [">", 100]}]
}
```

이 속성은 `rugby`로 설정된 `customer_interests` 값 그리고 100을 초과하는 값으로 설정된 `price_usd` 값을 갖는 모든 메시지 속성 또는 메시지 본문 속성과 일치합니다.

> **Note**
>
> 동일한 속성의 값에는 AND 로직을 적용할 수 없습니다.

OR 로직

속성 이름에 여러 값을 할당하여 OR 로직을 적용할 수 있습니다.

다음 정책을 살펴보겠습니다.

```
{
```

```
   "customer_interests": ["rugby", "football", "baseball"]
}
```

이 속성은 customer_interests 값이 rugby, football 또는 baseball로 설정된 모든 메시지 속성 또는 메시지 본문 속성과 일치합니다.

OR 연산자

"$or" 연산자를 사용해 필터 정책을 명시적으로 정의하여 정책에 있는 여러 속성 간의 OR 관계를 표현할 수 있습니다.

Amazon SNS는 정책이 다음 조건을 모두 충족하는 경우에만 "$or" 관계를 인식합니다. 이러한 조건이 모두 충족되지 않으면 "$or"은 정책의 다른 문자열과 마찬가지로 일반 속성 이름으로 처리됩니다.

- 규칙에 "$or" 필드 속성이 있고 그 뒤에 배열이 있습니다(예: “$or” : []).
- "$or" 배열에 최소 두 개의 객체가 있습니다. "$or": [{}, {}]
- "$or" 배열의 모든 객체에 예약된 키워드인 필드 이름이 없습니다.

그렇지 않으면 "$or"는 정책의 다른 문자열과 마찬가지로 일반 속성 이름으로 처리됩니다.

숫자와 접두사는 예약된 키워드이므로 다음 정책은 OR 관계로 구문 분석되지 않습니다.

```
{
   "$or": [ {"numeric" : 123}, {"prefix": "abc"} ]
}
```

OR 연산자 예제

표준 OR:

```
{
  "source": [ "aws.cloudwatch" ],
  "$or": [
    { "metricName": [ "CPUUtilization" ] },
    { "namespace": [ "AWS/EC2" ] }
  ]
}
```

이 정책의 필터 로직은 다음과 같습니다.

```
"source" && ("metricName" || "namespace")
```

이는 다음 메시지 속성 중 하나와 일치합니다.

```
"source": {"Type": "String", "Value": "aws.cloudwatch"},
"metricName": {"Type": "String", "Value": "CPUUtilization"}
```

또는

```
"source": {"Type": "String", "Value": "aws.cloudwatch"},
"namespace": {"Type": "String", "Value": "AWS/EC2"}
```

이 속성은 다음 메시지 본문 중 하나와도 일치합니다.

```
{
    "source": "aws.cloudwatch",
    "metricName": "CPUUtilization"
}
```

또는

```
{
    "source": "aws.cloudwatch",
    "namespace": "AWS/EC2"
}
```

OR 관계를 포함하는 정책 제약 조건

다음 정책을 살펴보겠습니다.

```
{
    "source": [ "aws.cloudwatch" ],
    "$or": [
      { "metricName": [ "CPUUtilization", "ReadLatency" ] },
      {
        "metricType": [ "MetricType" ] ,
        "$or" : [
          { "metricId": [ 1234, 4321 ] },
          { "spaceId": [ 1000, 2000, 3000 ] }
        ]
```

```
      }
    ]
  }
```

이 정책의 로직을 다음과 같이 단순화할 수도 있습니다.

```
("source" AND "metricName")
OR
("source" AND "metricType" AND "metricId")
OR
("source" AND "metricType" AND "spaceId")
```

OR 관계가 있는 정책의 복잡성 계산은 각 OR 문에 대한 조합 복잡성의 합으로 단순화할 수 있습니다.

따라서 전체 조합은 다음과 같습니다.

```
(source * metricName) + (source * metricType * metricId) + (source * metricType *
 spaceId)
= (1 * 2) + (1 * 1 * 2) + (1 * 1 * 3)
= 7
```

source에는 값이 하나, metricName에는 값이 두 개, metricType에는 값이 하나, metricId에는 값이 두 개, spaceId에는 값이 세 개 있습니다.

다음과 같은 중첩 필터 정책을 고려해 봅니다.

```
{
    "$or": [
      { "metricName": [ "CPUUtilization", "ReadLatency" ] },
      { "namespace": [ "AWS/EC2", "AWS/ES" ] }
    ],
    "detail" : {
      "scope" : [ "Service" ],
      "$or": [
        { "source": [ "aws.cloudwatch" ] },
        { "type": [ "CloudWatch Alarm State Change"] }
      ]
    }
  }
```

이 정책의 로직을 다음과 같이 단순화할 수 있습니다.

```
("metricName" AND ("detail"."scope" AND "detail"."source")
OR
("metricName" AND ("detail"."scope" AND "detail"."type")
OR
("namespace" AND ("detail"."scope" AND "detail"."source")
OR
("namespace" AND ("detail"."scope" AND "detail"."type")
```

총 조합 계산은 키의 중첩 수준을 고려해야 한다는 점을 제외하고 중첩되지 않은 정책과 동일합니다.

따라서 전체 조합은 다음과 같습니다.

```
(2 * 2 * 2) + (2 * 2 * 2) + (2 * 2 * 2) + (2 * 2 * 2) = 32
```

metricName에는 두 개의 값이 있고, namespace에는 두 개의 값이 있고, scope는 하나의 값이 있는 2수준 중첩 키이고, source는 하나의 값이 있는 2수준 중첩 키이며, type은 하나의 값이 있는 2수준 중첩 키입니다.

키 일치

exists 연산자를 사용하여 필터 정책에 지정된 속성이 있거나 없는 수신 메시지를 일치시킬 수 있습니다. exists 일치는 리프 노드에서만 작동합니다. 중간 노드에서는 작동하지 않습니다.

- "exists": true를 사용하여 지정된 속성을 포함하는 수신 메시지를 일치시킬 수 있습니다. 키는 null이 아니고 비어 있지 않은 값을 가져야 합니다.

  예를 들어, 다음 정책 속성은 true 값을 지닌 exists 연산자를 사용합니다.

  ```
  "store": [{"exists": true}]
  ```

  이 속성은 다음과 같이 store 속성 키가 있는 모든 메시지 속성 목록과 일치합니다.

  ```
  "store": {"Type": "String", "Value": "fans"}
  "customer_interests": {"Type": "String.Array", "Value": "[\"baseball\", \"basketball
  \"]"}
  ```

  이 속성은 다음 메시지 본문 중 하나와 일치합니다.

  ```
  {
      "store": "fans"
  ```

```
    "customer_interests": ["baseball", "basketball"]
}
```

그러나 다음과 같이 `store` 속성 키가 없는 메시지 속성 목록과는 일치하지 않습니다.

```
"customer_interests": {"Type": "String.Array", "Value": "[\"baseball\", \"basketball
\"]"}
```

다음 메시지 본문과도 일치하지 않습니다.

```
{
    "customer_interests": ["baseball", "basketball"]
}
```

- `"exists": false`를 사용하여 지정된 속성을 포함하지 않는 수신 메시지를 일치시킬 수 있습니다.

> **Note**
>
> `"exists": false`는 하나 이상의 속성이 있는 경우에만 일치합니다. 속성 세트가 비어 있으면 필터가 일치하지 않습니다.

예를 들어, 다음 정책 속성은 `false` 값을 지닌 `exists` 연산자를 사용합니다.

```
"store": [{"exists": false}]
```

이 속성은 다음과 같이 `store` 속성 키가 있는 모든 메시지 속성 목록과 일치하지 않습니다.

```
"store": {"Type": "String", "Value": "fans"}
"customer_interests": {"Type": "String.Array", "Value": "[\"baseball\", \"basketball
\"]"}
```

또한 다음 메시지 본문과도 일치하지 않습니다.

```
{
    "store": "fans"
    "customer_interests": ["baseball", "basketball"]
}
```

그러나 다음과 같이 store 속성 키가 없는 모든 메시지 속성 목록과 일치합니다.

```
"customer_interests": {"Type": "String.Array", "Value": "[\"baseball\", \"basketball
\"]"}
```

이 속성은 다음 메시지 본문과도 일치합니다.

```
{
    "customer_interests": ["baseball", "basketball"]
}
```

숫자 값 일치

숫자 값을 메시지 속성 값이나 메시지 본문 속성 값과 일치시켜 메시지를 필터링할 수 있습니다. 숫자 값은 JSON 정책에서 큰따옴표로 묶지 않습니다. 필터링에 다음과 같은 숫자 작업을 사용할 수 있습니다.

> **Note**
>
> 접두사는 문자열 일치에 대해서만 지원됩니다.

주제

- 정확한 일치
- Anything-but 일치
- 값 범위 일치

정확한 일치

정책 속성 값이 키워드 numeric과 연산자 =를 포함하는 경우, 동일한 이름과 동일한 숫자 값을 가지고 있는 모든 메시지 속성 또는 메시지 본문 속성값과 일치합니다.

다음 정책 속성을 살펴보겠습니다.

```
"price_usd": [{"numeric": ["=",301.5]}]
```

이 속성은 다음 메시지 속성 중 하나와 일치합니다.

```
"price_usd": {"Type": "Number", "Value": 301.5}
```

```
"price_usd": {"Type": "Number", "Value": 3.015e2}
```

이 속성은 다음 메시지 본문 중 하나와도 일치합니다.

```
{
   "price_usd": 301.5
}
```

```
{
   "price_usd": 3.015e2
}
```

Anything-but 일치

정책 속성 값에 키워드 anything-but가 포함되어 있으면 정책 속성 값을 포함하지 않는 모든 메시지 속성 또는 메시지 본문 속성값과 일치합니다.

다음 정책 속성을 살펴보겠습니다.

```
"price": [{"anything-but": [100, 500]}]
```

이 속성은 다음 메시지 속성 중 하나와 일치합니다.

```
"price": {"Type": "Number", "Value": 101}
```

```
"price": {"Type": "Number", "Value": 100.1}
```

이 속성은 다음 메시지 본문 중 하나와도 일치합니다.

```
{
   "price": 101
}
```

```
{
   "price": 100.1
}
```

또한 다음 메시지 속성과 일치합니다(100 또는 500이 아닌 값이 포함되어 있기 때문).

```
"price": {"Type": "Number.Array", "Value": "[100, 50]"}
```

또한 다음 메시지 본문과도 일치합니다(100 또는 500이 아닌 값이 포함되어 있기 때문).

```
{
   "price": [100, 50]
}
```

그러나 다음 메시지 속성과는 일치하지 않습니다.

```
"price": {"Type": "Number", "Value": 100}
```

다음 메시지 본문과도 일치하지 않습니다.

```
{
   "price": 100
}
```

값 범위 일치

연산자 = 외에도, 숫자 정책 속성은 다음 연산자를 포함할 수 있습니다. <, <=, > 및 >=.

다음 정책 속성을 살펴보겠습니다.

```
"price_usd": [{"numeric": ["<", 0]}]
```

이 속성은 음수 값을 갖는 모든 메시지 속성 또는 메시지 본문 속성과 일치합니다.

다른 메시지 속성을 살펴보겠습니다.

```
"price_usd": [{"numeric": [">", 0, "<=", 150]}]
```

이 속성은 최대 150까지의 양수를 갖는 모든 메시지 속성 또는 메시지 본문 속성과 일치합니다.

문자열 값 일치

문자열 값을 메시지 속성 값이나 메시지 본문 속성 값과 일치시켜 메시지를 필터링할 수 있습니다. 문자열 값은 JSON 정책에서 큰따옴표로 묶습니다. 다음 문자열 작업을 사용하여 메시지 속성 또는 메시지 본문을 일치시킬 수 있습니다.

주제

정확한 일치

정책 속성 값이 하나 이상의 메시지 속성 값과 일치하는 경우 정확한 일치가 발생합니다.

다음 정책 속성을 살펴보겠습니다.

```
"customer_interests": ["rugby", "tennis"]
```

이 속성은 다음 메시지 속성과 일치합니다.

```
"customer_interests": {"Type": "String", "Value": "rugby"}
```

```
"customer_interests": {"Type": "String", "Value": "tennis"}
```

또한 다음 메시지 본문과도 일치합니다.

```
{
   "customer_interests": "rugby"
}
```

```
{
   "customer_interests": "tennis"
}
```

그러나 다음 메시지 속성과는 일치하지 않습니다.

```
"customer_interests": {"Type": "String", "Value": "baseball"}
```

다음 메시지 본문과도 일치하지 않습니다.

```
{
   "customer_interests": "baseball"
}
```

Anything-but 일치

정책 속성 값에 키워드 anything-but가 포함되어 있으면 정책 속성 값을 포함하지 않는 모든 메시지 속성 또는 메시지 본문 값과 일치합니다. anything-but는 "exists": false와 함께 사용할 수 있습니다.

다음 정책 속성을 살펴보겠습니다.

```
"customer_interests": [{"anything-but": ["rugby", "tennis"]}]
```

이 속성은 다음 메시지 속성 중 하나와 일치합니다.

```
"customer_interests": {"Type": "String", "Value": "baseball"}
```

```
"customer_interests": {"Type": "String", "Value": "football"}
```

이 속성은 다음 메시지 본문 중 하나와도 일치합니다.

```
{
   "customer_interests": "baseball"
}
```

```
{
   "customer_interests": "football"
}
```

또한 다음 메시지 속성과 일치합니다(rugby 또는 tennis이 아닌 값이 포함되어 있기 때문).

```
"customer_interests": {"Type": "String.Array", "Value": "[\"rugby\", \"baseball\"]"}
```

또한 다음 메시지 본문과도 일치합니다(rugby 또는 tennis가 아닌 값이 포함되어 있기 때문).

```
{
```

```
   "customer_interests": ["rugby", "baseball"]
}
```

그러나 다음 메시지 속성과는 일치하지 않습니다.

```
"customer_interests": {"Type": "String", "Value": "rugby"}
```

다음 메시지 본문과도 일치하지 않습니다.

```
{
   "customer_interests": ["rugby"]
}
```

anything-but 연산자가 있는 접두사 사용

문자열 일치의 경우 anything-but 연산자와 함께 접두사를 사용할 수도 있습니다. 예를 들어 다음 정책 속성은 order- 접두사를 거부합니다.

```
"event":[{"anything-but": {"prefix": "order-"}}]
```

이 속성은 다음 속성 중 하나와 일치합니다.

```
"event": {"Type": "String", "Value": "data-entry"}
```

```
"event": {"Type": "String", "Value": "order_number"}
```

이 속성은 다음 메시지 본문 중 하나와도 일치합니다.

```
{
   "event": "data-entry"
}
```

```
{
   "event": "order_number"
}
```

그러나 다음 메시지 속성과는 일치하지 않습니다.

```
"event": {"Type": "String", "Value": "order-cancelled"}
```

다음 메시지 본문과도 일치하지 않습니다.

```
{
   "event": "order-cancelled"
}
```

E 매칭 quals-ignore-case

정책 속성이 키워드 `equals-ignore-case`를 포함하고 있는 경우, 모든 메시지 속성 또는 본문 속성 값에 대해 대소문자를 구분하지 않는 일치 연산을 수행합니다.

다음 정책 속성을 살펴보겠습니다.

```
"customer_interests": [{"equals-ignore-case": "tennis"}]
```

이 속성은 다음 메시지 속성 중 하나와 일치합니다.

```
"customer_interests": {"Type": "String", "Value": "TENNIS"}
```

```
"customer_interests": {"Type": "String", "Value": "Tennis"}
```

이 속성은 다음 메시지 본문 중 하나와도 일치합니다.

```
{
    "customer_interests": "TENNIS"
}
```

```
{
    "customer_interests": "teNnis"
{
```

IP 주소 일치

`cidr` 연산자를 사용하여 수신 메시지가 특정 IP 주소 또는 서브넷에서 발생하는지 확인할 수 있습니다.

다음 정책 속성을 살펴보겠습니다.

```
"source_ip":[{"cidr": "10.0.0.0/24"}]
```

이 속성은 다음 메시지 속성 중 하나와 일치합니다.

```
"source_ip": {"Type": "String", "Value": "10.0.0.0"}
```

```
"source_ip": {"Type": "String", "Value": "10.0.0.255"}
```

이 속성은 다음 메시지 본문 중 하나와도 일치합니다.

```
{
   "source_ip": "10.0.0.0"
}
```

```
{
   "source_ip": "10.0.0.255"
}
```

그러나 다음 메시지 속성과는 일치하지 않습니다.

```
"source_ip": {"Type": "String", "Value": "10.1.1.0"}
```

다음 메시지 본문과도 일치하지 않습니다.

```
{
   "source_ip": "10.1.1.0"
}
```

접두사 일치

정책 속성이 키워드 `prefix`를 포함하고 있는 경우, 이 값은 메시지 속성 또는 지정된 문자로 시작하는 모든 메시지 속성 값과 일치합니다.

다음 정책 속성을 살펴보겠습니다.

```
"customer_interests": [{"prefix": "bas"}]
```

이 속성은 다음 메시지 속성 중 하나와 일치합니다.

```
"customer_interests": {"Type": "String", "Value": "baseball"}
```

```
"customer_interests": {"Type": "String", "Value": "basketball"}
```

이 속성은 다음 메시지 본문 중 하나와도 일치합니다.

```
{
   "customer_interests": "baseball"
}
```

```
{
   "customer_interests": "basketball"
}
```

그러나 다음 메시지 속성과는 일치하지 않습니다.

```
"customer_interests": {"Type": "String", "Value": "rugby"}
```

다음 메시지 본문과도 일치하지 않습니다.

```
{
   "customer_interests": "rugby"
}
```

접미사 일치

정책 속성이 키워드 `suffix`를 포함하고 있는 경우, 이 값은 메시지 속성 또는 지정된 문자로 끝나는 모든 메시지 속성 값과 일치합니다.

다음 정책 속성을 살펴보겠습니다.

```
"customer_interests": [{"suffix": "ball"}]
```

이 속성은 다음 메시지 속성 중 하나와 일치합니다.

```
"customer_interests": {"Type": "String", "Value": "baseball"}
```

```
"customer_interests": {"Type": "String", "Value": "basketball"}
```

이 속성은 다음 메시지 본문 중 하나와도 일치합니다.

```
{
    "customer_interests": "baseball"
}
```

```
{
    "customer_interests": "basketball"
}
```

그러나 다음 메시지 속성과는 일치하지 않습니다.

```
"customer_interests": {"Type": "String", "Value": "rugby"}
```

다음 메시지 본문과도 일치하지 않습니다.

```
{
    "customer_interests": "rugby"
}
```

구독 필터 정책 적용

Amazon SNS 콘솔을 사용하여 필터 정책을 Amazon SNS 구독에 적용할 수 있습니다. 또는 Amazon SNS API, AWS Command Line Interface (AWS CLI) 또는 Amazon SNS를 지원하는 모든 AWS SDK를 사용하여 프로그래밍 방식으로 정책을 적용할 수 있습니다. 사용할 수도 있습니다. AWS CloudFormation

> **Important**
>
> AWS IAM 및 Amazon SNS와 같은 서비스는 최종 일관성이라는 분산 컴퓨팅 모델을 사용합니다. 구독 필터 정책의 추가 또는 변경 사항이 완전히 적용되려면 최대 15분이 소요됩니다.

AWS Management Console

1. Amazon SNS 콘솔에 로그인합니다.
2. 탐색 창에서 구독을 선택합니다.
3. 구독을 선택한 후 편집을 선택합니다.

4. 편집(Edit) 페이지에서 구독 필터 정책(Subscription filter policy) 섹션을 확장합니다.
5. 속성 기반 필터링 또는 페이로드 기반 필터링 중에서 선택합니다.
6. JSON 편집기(JSON editor) 필드에서 필터 정책의 JSON 본문(JSON body)을 제공합니다.
7. 변경 사항 저장를 선택합니다.

Amazon SNS에서 구독에 필터 정책을 적용합니다.

AWS CLI

AWS Command Line Interface (AWS CLI) 와 함께 필터 정책을 적용하려면 다음 예와 같이 `set-subscription-attributes`명령을 사용합니다. `--attribute-name` 옵션의 경우 `FilterPolicy`를 지정합니다. `--attribute-value`의 경우 JSON 정책을 지정합니다.

```
$ aws sns set-subscription-attributes --subscription-arn arn:aws:sns: ... --
attribute-name FilterPolicy --attribute-value '{"store":["example_corp"],"event":
["order_placed"]}'
```

정책에 유효한 JSON을 제공하려면 속성 이름과 값을 큰 따옴표로 묶습니다. 또한 전체 정책 인수를 따옴표로 묶어야 합니다. 따옴표 이스케이프를 방지하려면, 위의 예제와 같이 작은따옴표를 사용하여 정책을 묶고 큰따옴표를 사용하여 JSON 이름과 값을 묶을 수 있습니다.

속성 기반 (기본값) 에서 페이로드 기반 메시지 필터링으로 전환하려는 경우에도 명령을 사용할 수 있습니다. set-subscription-attributes `--attribute-name` 옵션의 경우 `FilterPolicyScope`를 지정합니다. `--attribute-value`에서 `MessageBody`를 지정합니다.

```
$ aws sns set-subscription-attributes --subscription-arn arn:aws:sns: ... --attribute-
name FilterPolicyScope --attribute-value MessageBody
```

필터 정책이 적용되었는지 확인하려면 `get-subscription-attributes` 명령을 사용합니다. 터미널 출력의 속성에는 다음 예제와 같이 `FilterPolicy` 키에 대한 필터 정책이 표시되어야 합니다:

```
$ aws sns get-subscription-attributes --subscription-arn arn:aws:sns: ...
{
    "Attributes": {
        "Endpoint": "endpoint . . .",
        "Protocol": "https",
        "RawMessageDelivery": "false",
        "EffectiveDeliveryPolicy": "delivery policy . . .",
        "ConfirmationWasAuthenticated": "true",
```

```
        "FilterPolicy": "{\"store\": [\"example_corp\"], \"event\": [\"order_placed
\"]}",
        "FilterPolicyScope": "MessageAttributes",
        "Owner": "111122223333",
        "SubscriptionArn": "arn:aws:sns: . . .",
        "TopicArn": "arn:aws:sns: . . ."
    }
}
```

AWS SDK

다음 코드 예제는 사용 SetSubscriptionAttributes 방법을 보여줍니다.

> **Important**
>
> Java 2.x 예제용 SDK를 사용하는 경우 클래스 SNSMessageFilterPolicy는 즉시 사용할 수 없습니다. 이 클래스를 설치하는 방법에 대한 지침은 GitHub 웹 사이트의 예제를 참조하십시오.

CLI

AWS CLI

구독 속성을 설정하려면

다음 set-subscription-attributes 예제에서는 SQS 구독에 RawMessageDelivery 속성을 설정합니다.

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name RawMessageDelivery \
    --attribute-value true
```

이 명령은 출력을 생성하지 않습니다.

다음 set-subscription-attributes 예제에서는 SQS 구독에 FilterPolicy 속성을 설정합니다.

```
aws sns set-subscription-attributes \
```

```
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name FilterPolicy \
    --attribute-value "{ \"anyMandatoryKey\": [\"any\", \"of\", \"these\"] }"
```

이 명령은 출력을 생성하지 않습니다.

다음 `set-subscription-attributes` 예제에서는 SQS 구독에서 `FilterPolicy` 속성을 제거합니다.

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name FilterPolicy \
    --attribute-value "{}"
```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 SetSubscriptionAttributes참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class UseMessageFilterPolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionArn>

                Where:
                   subscriptionArn - The ARN of a subscription.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        usePolicy(snsClient, subscriptionArn);
        snsClient.close();
    }

    public static void usePolicy(SnsClient snsClient, String subscriptionArn) {
        try {
            SNSMessageFilterPolicy fp = new SNSMessageFilterPolicy();
            // Add a filter policy attribute with a single value
            fp.addAttribute("store", "example_corp");
            fp.addAttribute("event", "order_placed");

            // Add a prefix attribute
            fp.addAttributePrefix("customer_interests", "bas");

            // Add an anything-but attribute
            fp.addAttributeAnythingBut("customer_interests", "baseball");

            // Add a filter policy attribute with a list of values
            ArrayList<String> attributeValues = new ArrayList<>();
            attributeValues.add("rugby");
```

```
            attributeValues.add("soccer");
            attributeValues.add("hockey");
            fp.addAttribute("customer_interests", attributeValues);

            // Add a numeric attribute
            fp.addAttribute("price_usd", "=", 0);

            // Add a numeric attribute with a range
            fp.addAttributeRange("price_usd", ">", 0, "<=", 100);

            // Apply the filter policy attributes to an Amazon SNS subscription
            fp.apply(snsClient, subscriptionArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API SetSubscriptionAttributes참조를 참조하십시오.

Python

SDK for Python(Boto3)

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource
```

```
    @staticmethod
    def add_subscription_filter(subscription, attributes):
        """
        Adds a filter policy to a subscription. A filter policy is a key and a
        list of values that are allowed. When a message is published, it must
have an
        attribute that passes the filter or it will not be sent to the
subscription.

        :param subscription: The subscription the filter policy is attached to.
        :param attributes: A dictionary of key-value pairs that define the
filter.
        """
        try:
            att_policy = {key: [value] for key, value in attributes.items()}
            subscription.set_attributes(
                AttributeName="FilterPolicy",
AttributeValue=json.dumps(att_policy)
            )
            logger.info("Added filter to subscription %s.", subscription.arn)
        except ClientError:
            logger.exception(
                "Couldn't add filter to subscription %s.", subscription.arn
            )
            raise
```

- API에 대한 자세한 내용은 파이썬용AWS SDK (Boto3) API 레퍼런스를 참조하십시오 SetSubscriptionAttributes.

Amazon SNS API

Amazon SNS API를 사용하여 필터 정책을 적용하려면 `SetSubscriptionAttributes` 작업에 대한 요청을 수행합니다. `AttributeName` 파라미터를 `FilterPolicy`로 설정하고 `AttributeValue` 파라미터를 필터 정책 JSON으로 설정합니다.

속성 기반(기본값)에서 페이로드 기반 메시지 필터링으로 전환하려는 경우에도 `SetSubscriptionAttributes` 작업을 사용할 수 있습니다. `AttributeName` 매개 변수를 `FilterPolicyScope`로 설정하고 `AttributeValue` 매개 변수를 `MessageBody`로 설정합니다.

AWS CloudFormation

를 사용하여 AWS CloudFormation필터 정책을 적용하려면 JSON 또는 YAML 템플릿을 사용하여 스택을 생성하십시오. AWS CloudFormation 자세한 내용은 AWS CloudFormation 사용 설명서의 *AWS::SNS::Subscription* 리소스 FilterPolicy속성 및 예제 AWS CloudFormation 템플릿을 참조하십시오.

1. AWS CloudFormation 콘솔에 로그인합니다.
2. 스택 생성을 선택합니다.
3. 템플릿 선택 페이지에서 Amazon S3에 템플릿 업로드를 선택하고 파일을 선택한 후 다음을 선택합니다.
4. 세부 정보 지정 페이지에서 다음 작업을 수행합니다.
   a. 스택 이름에 MyFilterPolicyStack을 입력합니다.
   b. 의 myHttpEndpoint경우 주제를 구독할 HTTP 엔드포인트를 입력합니다.

   > **Tip**
   > HTTP 엔드포인트가 없는 경우 하나를 생성합니다.

5. 옵션 페이지에서 다음을 선택합니다.
6. 검토(Review) 페이지에서 생성(Create)을 선택합니다.

구독 필터 정책 제거

구독자에게 보내는 메시지의 필터링을 중지하려면 해당 구독의 필터 정책을 빈 JSON 본문으로 덮어써서 제거합니다. 정책을 제거한 뒤에도 해당 구독은 자신을 상대로 게시되는 모든 메시지를 받게 됩니다.

AWS Management Console

1. Amazon SNS 콘솔에 로그인합니다.
2. 탐색 창에서 구독을 선택합니다.
3. 구독을 선택한 후 편집을 선택합니다.
4. Edit ***EXAMPLE1-23bc-4567-d890-ef12g3hij456***(예제 1-23bc-4567-d890-ef12g3hij456 편집) 페이지에서 Subscription filter policy(구독 필터 정책) 섹션을 확장합니다.

5. JSON 편집기 필드에서 필터 정책의 빈 JSON 본문({})을 제공합니다.
6. Save changes(변경 사항 저장)를 선택합니다.

   Amazon SNS에서 구독에 필터 정책을 적용합니다.

AWS CLI

AWS CLI에서 필터 정책을 제거하려면 `set-subscription-attributes` 명령을 사용하고 `--attribute-value` 인수에 빈 JSON 본문을 입력합니다.

```
$ aws sns set-subscription-attributes --subscription-arn arn:aws:sns: ... --attribute-name FilterPolicy --attribute-value "{}"
```

Amazon SNS API

Amazon SNS API를 사용하여 필터 정책을 제거하려면 `SetSubscriptionAttributes` 작업을 요청합니다. `AttributeName` 파라미터를 `FilterPolicy`로 설정하고 `AttributeValue` 파라미터에 빈 JSON 문자열을 입력합니다.

메시지 데이터 보호

주제

메시지 데이터 보호란 무엇입니까?

메시지 데이터 보호는 데이터 보호 정책을 사용하여 애플리케이션 또는 AWS 서비스 간에 이동하는 중요한 정보를 감사하고 마스킹하고 수정하고 차단함으로써 Amazon SNS 주제에 게시된 데이터를 보호합니다.

메시지 데이터 보호는 데이터 식별자를 사용하여 이동 중인 데이터에서 개인 식별 정보(PII) 및 보호 대상 건강 정보(PHI)를 검색합니다. 사전 정의된(또는 Amazon SNS 관리형) 데이터 식별자(예: 이름, 주소, 신용 카드 번호 및 처방약 코드)를 사용하거나 비즈니스 사용 사례에 맞는 사용자 지정 데이터 식별자를 생성할 수 있습니다. 검색한 정보를 사용하여 메시지 데이터 보호는 자세한 감사 로그를 제공하고 해당 데이터를 보호하기 위한 조치를 취할 수 있도록 합니다.

메시지 데이터 보호는 중요한 고객 정보를 보호하는 데 도움이 되는 다음 작업을 지원합니다.

- 감사 – Amazon SNS 주제에 게시된 데이터의 최대 99%를 감사합니다. 그런 다음 Amazon CloudWatch, Amazon S3 또는 Amazon Data Firehose로 결과를 전송하도록 선택할 수 있습니다.
- 비식별화 - 메시지 게시를 중단하지 않고 민감한 데이터를 마스킹하거나 수정합니다.
- 거부 – 페이로드 내에 중요한 데이터가 있는 경우 애플리케이션과 AWS 리소스 간의 데이터 전송을 차단합니다.

> **Note**
>
> Amazon SNS는 Amazon SNS 표준 주제에 대해서만 메시지 데이터 보호를 지원합니다.

메시지 데이터 보호를 사용해야 하는 이유는 무엇입니까?

거버넌스, 위험 관리 및 규정 준수 프로그램에 메시지 데이터 보호를 도입하여 데이터 유출을 식별하고 방지하는 데 도움이 되는 데이터 보호 정책을 구현할 수 있습니다. 이는 HIPAA, GDPR, PCI 및 FedRAMP와 같은 개인 정보 보호 규정을 준수하여 금융, 법률 및 규제 위험을 줄이는 데 도움이 되는 도구를 팀에 제공합니다. 또한 개발자가 중요한 데이터를 보호하기 위한 자체 도구를 구축하고 관리하는 것과 관련된 운영 오버헤드를 해소할 수 있습니다.

예를 들어 메시지 데이터 보호를 사용하여 감사 정책을 만들어 시스템에서 중요한 데이터를 실수로 보내거나 받는지 여부를 확인할 수 있습니다. 감사 결과 시스템이 신용 카드 정보가 필요하지 않은 시스템에 신용 카드 정보를 보내는 것으로 나타나면 차단 정책을 사용하여 데이터 전달을 방지할 수 있습니다.

> **Note**
>
> Amazon SNS는 Amazon SNS 표준 주제에 대해서만 메시지 데이터 보호를 지원합니다.

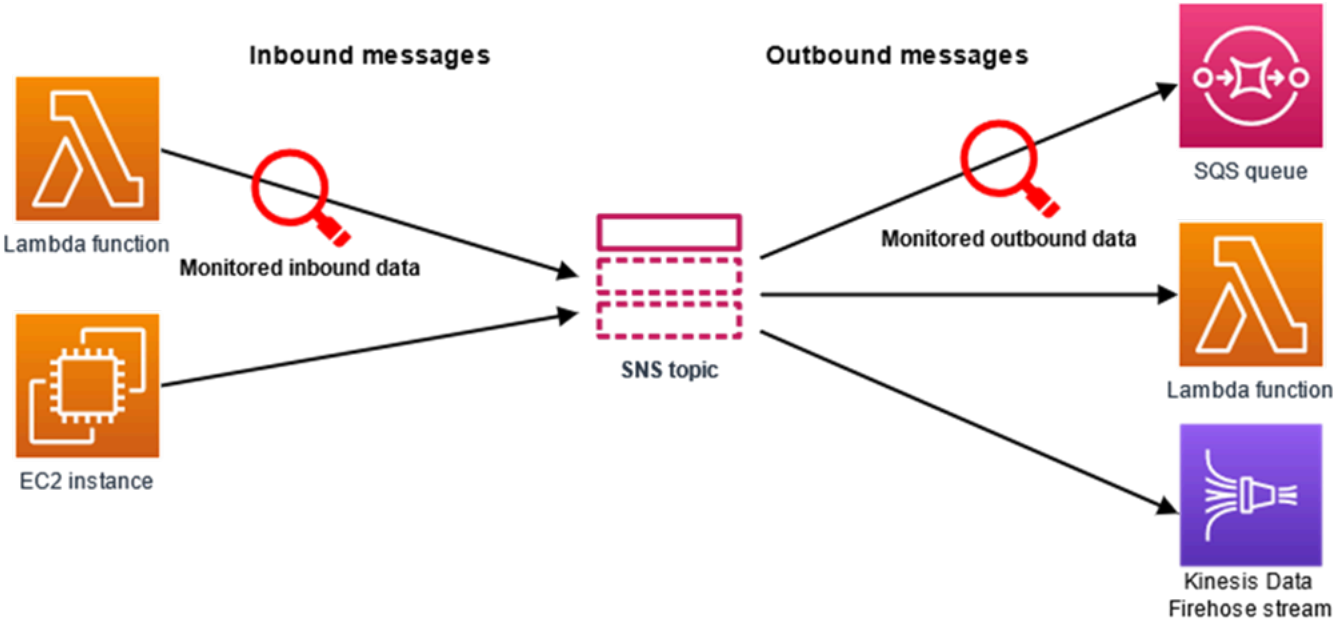
데이터 보호 정책 이해

주제

데이터 보호 정책이란 무엇입니까?

Amazon SNS는 데이터 보호 정책을 사용하여 검색하려는 중요한 데이터와 해당 데이터가 Amazon SNS 주제에 의해 교환되지 않도록 보호하기 위해 취하려는 작업을 선택합니다. 관심 있는 중요한 데이터를 선택하려면 데이터 식별자를 사용합니다. 그런 다음 Amazon SNS 메시지 데이터 보호는 기계 학습 및 패턴 일치를 사용하여 중요한 데이터를 탐지합니다. 발견된 데이터 식별자에 따라 조치를 취하기

위해 감사, 비식별 또는 거부 작업을 정의할 수 있습니다. 이러한 작업을 통해 발견된(또는 찾을 수 없는) 중요한 데이터를 기록하거나 마스킹, 수정 또는 메시지 배달을 거부할 수 있습니다.

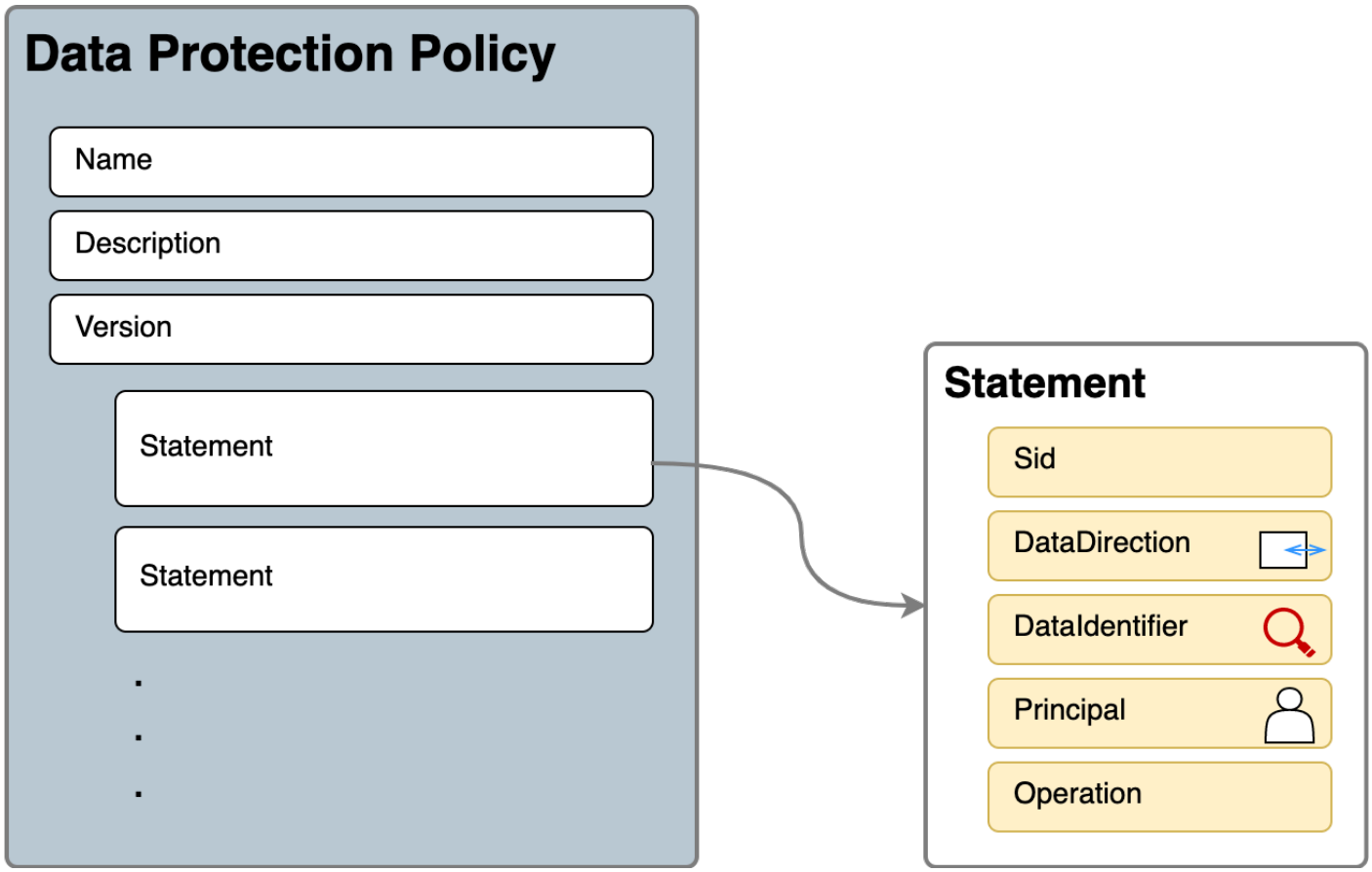


데이터 보호 정책은 어떻게 구성되어 있습니까?

다음 그림처럼 데이터 보호 정책 문서는 다음 요소를 포함합니다.

- 문서 상단에 위치하는 정책 전반의 선택적 정보
- 하나 이상의 개별 문

각 설명문에는 단일 권한에 대한 정보가 포함되어 있습니다.



Amazon SNS 주제당 하나의 데이터 보호 정책만 정의할 수 있습니다. 데이터 보호 정책은 하나 이상의 거부 또는 비식별 명령문과 하나의 감사 명령문을 가질 수 있습니다.

데이터 보호 정책의 JSON 속성

데이터 보호 정책에는 식별을 위해 다음과 같은 기본 정책 정보가 필요합니다.

- Name(이름) – 정책 이름입니다.
- Description(설명)(선택 사항) – 정책 설명입니다.
- Version(버전) - 정책 언어 버전입니다. 현재 버전은 2021-06-01입니다.
- Statement(명령문) - 데이터 보호 정책 조치를 지정하는 명령문 목록입니다.

```
{
  "Name": "basicPII-protection",
  "Description": "Protect basic types of sensitive data",
  "Version": "2021-06-01",
  "Statement": [
```

```
        ...
  ]
}
```

정책 명령문을 위한 JSON 속성

정책 명령문은 데이터 보호 작업에 대한 탐지 컨텍스트를 설정합니다.

- Sid(선택 사항) - 명령문 식별자입니다.
- DataDirection - Amazon SNS 주제에 대한 인바운드(Publish API 요청의 경우) 또는 아웃바운드(알림 전달의 경우)입니다.
- DataIdentifier – Amazon SNS 주제가 검색해야 하는 중요한 데이터입니다. 이러한 데이터로는 이름, 주소 또는 전화번호가 있습니다.
- Principal - 주제에 게시한 IAM 보안 주체 또는 해당 주제를 구독한 IAM 보안 주체입니다.
- Operation(작업) - Amazon SNS 주제가 중요한 데이터를 찾으면 실행하는 후속 작업인 감사, 비식별(마스킹 또는 수정) 또는 거부(차단)입니다.

```
{
    "Sid": "basicPII-inbound-protection",
    "DataDirection": "Inbound",
    "Principal": ["*"],
    "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/Name",
        "arn:aws:dataprotection::aws:data-identifier/PhoneNumber-US"
    ],
    "Operation": {
        ...
    }
}
```

정책 명령문 작업을 위한 JSON 속성

정책 명령문은 다음 데이터 보호 작업 중 하나를 설정합니다.

- Audit(감사) – 메시지 게시 또는 배달을 중단하지 않고 메트릭을 내보내고 로그를 찾습니다.
- De-identify(비식별) - 메시지 게시를 중단하지 않고 민감한 데이터를 마스킹하거나 수정합니다.
- Deny(거부) - Amazon SNS 게시 요청을 차단하거나 메시지 전송에 실패합니다.

내 데이터 보호 정책에 대한 IAM 보안 주체를 결정하려면 어떻게 해야 합니까?

메시지 데이터 보호는 Amazon SNS와 상호 작용하는 두 개의 IAM 보안 주체를 사용합니다.

1. Publish API 보안 주체(인바운드) – Amazon SNS `Publish` API를 호출하는 인증된 IAM 보안 주체입니다.
2. 구독 보안 주체(아웃바운드) – 구독 생성 중에 `Subscribe` API를 호출한 인증된 보안 주체입니다.

`SubscriptionPrincipal`은 `GetSubscriptionAttributes` API에서 검색할 수 있는 공개적으로 사용 가능한 Amazon SNS 구독 속성입니다.

```
{
  "Attributes": {
    "SubscriptionPrincipal": "arn:aws:iam::123456789012:user/NoNameAccess",
    "Owner": "123412341234",
    "RawMessageDelivery": "true",
    "TopicArn": "arn:aws:sns:us-east-1:123412341234:PII-data-topic",
    "Endpoint": "arn:aws:sqs:us-east-1:123456789012:NoNameAccess",
    "Protocol": "sqs",
    "PendingConfirmation": "false",
    "ConfirmationWasAuthenticated": "true",
    "SubscriptionArn": "arn:aws:sns:us-east-1:123412341234:PII-data-
topic:5d8634ef-67ef-49eb-a824-4042b28d6f55"
  }
}
```

데이터 보호 정책 작업

다음은 중요한 데이터를 감사 및 거부하는 데 사용할 수 있는 데이터 보호 정책의 예입니다. 예시 애플리케이션이 포함된 전체 자습서는 Amazon SNS에 대한 메시지 데이터 보호 소개 블로그 게시물을 참조하세요.

주제

- 감사 작업
- 비식별 작업
- 거부 작업

감사 작업

Audit(감사) 작업은 주제 인바운드 메시지를 샘플링하고 AWS 대상에 중요한 데이터 결과를 기록합니다. 샘플 레이트는 0~99 사이의 정수일 수 있습니다. 이 작업에는 다음 유형의 로깅 대상 중 하나가 필요합니다.

1. FindingsDestination— Amazon SNS 주제가 페이로드에서 민감한 데이터를 발견했을 때의 로깅 대상.
2. NoFindingsDestination— Amazon SNS 주제가 페이로드에서 민감한 데이터를 찾지 못한 경우의 로깅 대상.

다음 각 로그 대상 유형에서 다음 AWS 서비스 서비스를 사용할 수 있습니다.

- Amazon CloudWatch Logs (선택 사항) — 주제 지역에 `LogGroup` 있어야 하고 이름은 /aws/vendedlogs/로 시작해야 합니다.
- Amazon Data Firehose (선택 사항) — 주제 지역에 `DeliveryStream` 있어야 하며 전송 스트림의 소스로 Direct PUT이 있어야 합니다. 자세한 내용은 Amazon Data Firehose 개발자 안내서의 소스, 대상 및 이름을 참조하십시오.
- Amazon S3(선택 사항) - Amazon S3 버킷 이름입니다. SSE-KMS 암호화가 활성화된 Amazon S3 버킷을 사용하려면 추가 작업이 필요합니다.

```
{
  "Operation": {
    "Audit": {
      "SampleRate": "99",
      "FindingsDestination": {
            "CloudWatchLogs": {
                "LogGroup": "/aws/vendedlogs/log-group-name"
            },
            "Firehose": {
                "DeliveryStream": "delivery-stream-name"
            },
            "S3": {
                "Bucket": "bucket-name"
            }
      },
      "NoFindingsDestination": {
            "CloudWatchLogs": {
                "LogGroup": "/aws/vendedlogs/log-group-name"
```

```
            },
            "Firehose": {
                "DeliveryStream": "delivery-stream-name"
            },
            "S3": {
                "Bucket": "bucket-name"
            }
        }
    }
  }
}
```

로그 대상 지정 시 필요한 권한

데이터 보호 정책에서 로깅 대상을 지정할 때 Amazon SNS `PutDataProtectionPolicy` API 또는 `--data-protection-policy` 파라미터가 있는 `CreateTopic` API를 호출하는 IAM 보안 주체의 IAM 자격 증명 정책에 다음 권한을 추가해야 합니다.

감사 대상	IAM 권한
기본값	logs:CreateLogDelivery logs:GetLogDelivery logs:UpdateLogDelivery logs:DeleteLogDelivery logs:ListLogDeliveries
CloudWatchLogs	logs:PutResourcePolicy logs:DescribeResourcePolicies logs:DescribeLogGroups
Firehose	iam:CreateServiceLinkedRole firehose:TagDeliveryStream
S3	s3:PutBucketPolicy s3:GetBucketPolicy

감사 대상	IAM 권한
	SSE-KMS 암호화가 활성화된 Amazon S3 버킷을 사용하려면 추가 작업이 필요합니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogDelivery",
        "logs:GetLogDelivery",
        "logs:UpdateLogDelivery",
        "logs:DeleteLogDelivery",
        "logs:ListLogDeliveries"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "logs:PutResourcePolicy",
        "logs:DescribeResourcePolicies",
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:region:account-id:SampleLogGroupName:*:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole",
        "firehose:TagDeliveryStream"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
```

```
      "Action": [
        "s3:PutBucketPolicy",
        "s3:GetBucketPolicy"
      ],
      "Resource": [
        "arn:aws:s3:::bucket-name"
      ]
    }
  ]
}
```

SSE-KMS를 사용할 경우 필요한 키 정책

Amazon S3 버킷을 로그 대상으로 사용하면 Amazon S3-관리형 키(SSE-S3)를 사용한 서버 측 암호화 또는 AWS KMS keys(SSE-KMS)를 사용한 서버 측 암호화를 활성화하여 버킷의 데이터를 보호할 수 있습니다. 자세한 내용은 Amazon S3 사용 설명서의 서버 측 암호화를 사용하여 데이터 보호를 참조하세요.

SSE-S3를 선택하면 추가 구성이 필요하지 않습니다. Amazon S3는 암호화 키를 처리합니다.

SSE-KMS를 선택하면 고객 관리형 키를 사용해야 합니다. 로그 전달 계정이 S3 버킷에 쓸 수 있도록 고객 관리형 키에 대한 키 정책을 업데이트해야 합니다. SSE-KMS와 함께 사용하는 데 필요한 키 정책에 대한 자세한 내용은 Amazon Logs 사용 설명서의 Amazon S3 버킷 서버 측 암호화를 참조하십시오. CloudWatch

감사 대상 로그 예시

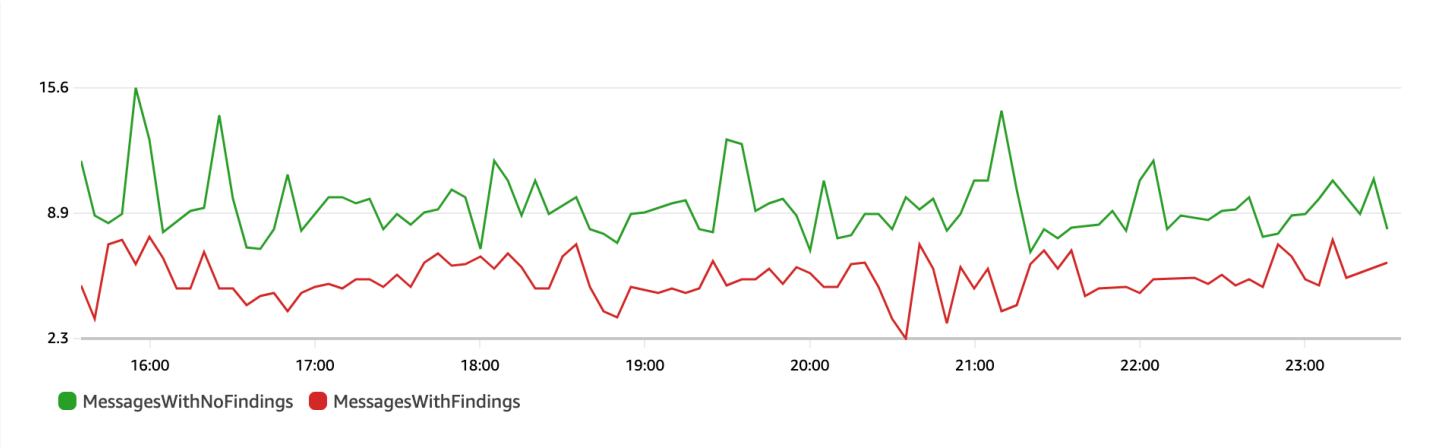
다음 예에서는 `callerPrincipal`을 사용하여 민감한 콘텐츠의 소스를 식별하고 `messageID`를 참조로 사용하여 Publish API 응답을 확인합니다.

```
{
  "messageId": "34d9b400-c6dd-5444-820d-fbeb0f1f54cf",
  "auditTimestamp": "2022-05-12T2:10:44Z",
  "callerPrincipal": "arn:aws:iam::123412341234:role/Publisher",
  "resourceArn": "arn:aws:sns:us-east-1:123412341234:PII-data-topic",
  "dataIdentifiers": [
    {
      "name": "Name",
      "count": 1,
      "detections": [
        {
          "start": 1,
          "end": 2
```

```
        }
      ]
    },
    {
      "name": "PhoneNumber",
      "count": 2,
      "detections": [
        {
          "start": 3,
          "end": 4
        },
        {
          "start": 5,
          "end": 6
        }
      ]
    }
  ]
}
```

감사 작업 지표

감사 작업에서 NoFindingsDestination 속성 FindingsDestination 또는 속성을 지정하면 주제 소유자에게도 지표가 제공됩니다. CloudWatch MessagesWithFindings MessagesWithNoFindings



비식별 작업

비식별화 작업은 게시되거나 배달된 메시지에서 민감한 데이터를 마스킹하거나 수정합니다. 이 작업은 인바운드 메시지와 아웃바운드 메시지에 모두 사용할 수 있으며 다음 유형의 구성 중 하나가 필요합니다.

- MaskConfig— 다음 표에서 지원되는 문자를 사용하여 마스킹하십시오. 예를 들어, ssn: 123-45-6789는 ssn: ###########이 됩니다.

```
{
"Operation": {
    "Deidentify": {
        "MaskConfig": {
            "MaskWithCharacter": "#"
          }
    }
}
```

지원되는 마스킹 문자	명칭
*	별표
A~Z, a~z, 0~9	영숫자
	공간
!	느낌표
$	달러 기호
%	퍼센트 기호
&	앰퍼샌드
()	괄호
+	더하기 기호
,	쉼표
-	하이픈
.	기간
/\	슬래시, 백 슬래시
#	숫자 기호

지원되는 마스킹 문자	명칭
:	콜론
;	세미콜론
=, <>	같음, 보다 작음, 보다 큼
@	at 기호
[]	대괄호
^	캐럿
_	밑줄
`	백틱
|	수직 막대
~	틸데

- RedactConfig— 데이터를 완전히 제거하여 수정합니다. 예를 들어, ssn: 123-45-6789는 ssn: 이 됩니다.

```
{
"Operation": {
    "Deidentify": {
        "RedactConfig": {}
    }
}
```

인바운드 메시지에서 중요한 데이터는 감사 작업 후 익명화되며, 전체 메시지가 중요한 경우 SNS:Publish API 호출자는 다음과 같은 잘못된 매개변수 오류를 수신합니다.

```
Error code: AuthorizationError ...
```

거부 작업

Deny(거부) 작업은 Publish API 요청 또는 메시지에 중요한 데이터가 포함된 경우 메시지 배달을 중단합니다. Deny(거부) 작업 개체는 추가 구성이 필요하지 않으므로 비어 있습니다.

```
"Operation": {
    "Deny": {}
}
```

인바운드 메시지에서 SNS:Publish API 호출자는 권한 부여 오류를 수신합니다.

```
Error code: AuthorizationError ...
```

아웃바운드 메시지에서 Amazon SNS 주제는 구독에 메시지를 전달하지 않습니다. 승인되지 않은 배달을 추적하려면 주제의 전송 상태 로깅을 사용하도록 설정하세요. 다음은 전송 상태 로그의 예입니다.

```
{
    "notification": {
        "messageMD5Sum": "29638742ffb68b32cf56f42a79bcf16b",
        "messageId": "34d9b400-c6dd-5444-820d-fbeb0f1f54cf",
        "topicArn": "arn:aws:sns:us-east-1:123412341234:PII-data-topic",
        "timestamp": "2022-05-12T2:12:44Z"
    },
    "delivery": {
        "deliveryId": "98236591c-56aa-51ee-a5ed-0c7d43493170",
        "destination": "arn:aws:sqs:us-east-1:123456789012:NoNameAccess",
        "providerResponse": "The topic's data protection policy prohibits this message
 from being delivered to <subscription-arn>",
        "dwellTimeMs":20,
        "attempts":1,
        "statusCode": 403
    },
    "status": "FAILURE"
}
```

데이터 보호 정책 예시

다음 예는 중요한 데이터를 감사 및 거부하는 데 사용할 수 있는 데이터 보호 정책입니다. 예시 애플리케이션이 포함된 전체 자습서는 Amazon SNS에 대한 메시지 데이터 보호 소개 블로그 게시물을 참조하세요.

주제

- 감사 정책 예시
- 인바운드 비식별 마스킹 문이 포함된 정책 예시
- 인바운드 비식별 수정 문이 포함된 정책 예시

- 아웃바운드 비식별 마스킹 문이 포함된 정책 예시
- 아웃바운드 비식별 수정 문이 포함된 정책 예시
- 정책 인바운드 거부 명령문 예시
- 정책 아웃바운드 거부 명령문 예시

감사 정책 예시

감사 정책을 사용하면 최대 99% 의 인바운드 메시지를 감사하고 조사 결과를 Amazon CloudWatch, Amazon Data Firehose 및 Amazon S3로 보낼 수 있습니다.

예를 들어 감사 정책을 만들어 시스템에서 중요한 데이터를 실수로 보내거나 받는지 여부를 평가할 수 있습니다. 감사 결과 시스템에서 신용 카드 정보가 필요하지 않은 시스템에 신용 카드 정보를 보내는 것으로 나타나면 데이터 보호 정책을 구현하여 데이터 전달을 차단할 수 있습니다.

다음 예제는 신용 카드 번호를 찾고 그 결과를 CloudWatch Logs, Firehose 및 Amazon S3로 전송하여 주제를 관통하는 메시지의 99% 를 감사합니다.

데이터 보호 정책:

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Inbound",
      "Principal": ["*"],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Audit": {
          "SampleRate": "99",
          "FindingsDestination": {
            "CloudWatchLogs": {
              "LogGroup": "<example log name>"
            },
            "Firehose": {
              "DeliveryStream": "<example stream name>"
            },
            "S3": {
```

```
              "Bucket": "<example bucket name>"
            }
          }
        }
      }
    }
  ]
}
```

감사 결과 형식 예시:

```
{
    "messageId": "...",
    "callerPrincipal": "arn:aws:sts::123456789012:assumed-role/ExampleRole",
    "resourceArn": "arn:aws:sns:us-east-1:123456789012:ExampleArn",
    "dataIdentifiers": [
        {
            "name": "CreditCardNumber",
            "count": 1,
            "detections": [
                { "start": 1, "end": 2 }
            ]
        }
    ],
    "timestamp": "2021-04-20T00:33:40.241Z"
}
```

인바운드 비식별 마스킹 문이 포함된 정책 예시

다음 예에서는 메시지 콘텐츠에서 민감한 데이터를 마스킹하여 사용자가 `CreditCardNumber`가 포함된 메시지를 주제에 게시하지 못하도록 차단합니다.

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Inbound",
      "Principal": [
        "arn:aws:iam::123456789012:user/ExampleUser"
      ],
```

```
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Deidentify": {
          "MaskConfig": {
            "MaskWithCharacter": "#"
          }
        }
      }
    }
  ]
}
```

인바운드 비식별 마스킹 결과 예시:

```
// original message
My credit card number is 4539894458086459

// delivered message
My credit card number is ################
```

인바운드 비식별 수정 문이 포함된 정책 예시

다음 예에서는 메시지 콘텐츠에서 민감한 데이터를 수정하여 사용자가 CreditCardNumber가 포함된 메시지를 주제에 게시하지 못하도록 차단합니다.

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Inbound",
      "Principal": [
        "arn:aws:iam::123456789012:user/ExampleUser"
      ],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Deidentify": {
          "RedactConfig": {}
```

```
        }
      }
    }
  ]
}
```

인바운드 비식별 수정 결과 예시:

```
// original message
My credit card number is 4539894458086459

// delivered message
My credit card number is
```

아웃바운드 비식별 마스킹 문이 포함된 정책 예시

다음 예에서는 메시지 콘텐츠에서 민감한 데이터를 마스킹하여 사용자가 `CreditCardNumber`가 포함된 메시지를 수신하지 않도록 차단합니다.

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Outbound",
      "Principal": [
        "arn:aws:iam::123456789012:user/ExampleUser"
      ],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Deidentify": {
          "MaskConfig": {
            "MaskWithCharacter": "-"
          }
        }
      }
    }
  ]
}
```

아웃바운드 비식별 마스킹 결과 예시:

```
// original message
My credit card number is 4539894458086459

// delivered message
My credit card number is ----------------
```

아웃바운드 비식별 수정 문이 포함된 정책 예시

다음 예에서는 메시지 콘텐츠에서 민감한 데이터를 수정하여 사용자가 `CreditCardNumber`가 포함된 메시지를 수신하지 않도록 차단합니다.

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Outbound",
      "Principal": [
        "arn:aws:iam::123456789012:user/ExampleUser"
      ],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Deidentify": {
          "RedactConfig": {}
        }
      }
    }
  ]
}
```

아웃바운드 비식별 수정 결과 예시:

```
// original message
My credit card number is 4539894458086459

// delivered message
My credit card number is
```

정책 인바운드 거부 명령문 예시

다음 예에서는 사용자가 메시지 콘텐츠에 CreditCardNumber가 있는 주제에 메시지를 게시하지 못하도록 차단합니다. API 응답에서 거부된 페이로드의 상태 코드는 "403 AuthorizationError"입니다.

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Inbound",
      "Principal": [
        "arn:aws:iam::123456789012:user/ExampleUser"
      ],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
        "Deny": {}
      }
    }
  ]
}
```

정책 아웃바운드 거부 명령문 예시

다음 예에서는 AWS 계정이 CreditCardNumber가 포함된 메시지를 수신하지 못하도록 차단합니다.

```
{
  "Name": "__example_data_protection_policy",
  "Description": "Example data protection policy",
  "Version": "2021-06-01",
  "Statement": [
    {
      "DataDirection": "Outbound",
      "Principal": [
        "arn:aws:iam::123456789012:user/ExampleUser"
      ],
      "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/CreditCardNumber"
      ],
      "Operation": {
```

```
        "Deny": {}
      }
    }
  ]
}
```

Amazon에서 로그인한 아웃바운드 거부 결과 예제: CloudWatch

```
{
  "notification": {
    "messageMD5Sum": "2e8f58ff2eeed723b56b15493fbfb5a5",
    "messageId": "8747a956-ebf1-59da-b291-f2c2e4b87c9c",
    "topicArn": "arn:aws:sns:us-east-2:664555388960:test1",
    "timestamp": "2022-09-08 15:40:57.144"
  },
  "delivery": {
    "deliveryId": "6a422437-78cc-5171-ad64-7fa3778507aa",
    "destination": "arn:aws:sqs:us-east-2:664555388960:test",
    "providerResponse": "The topic's data protection policy prohibits this message from
 being delivered to <subscription arn>",
    "dwellTimeMs": 22,
    "attempts": 1,
    "statusCode": 403
  },
  "status": "FAILURE"
}
```

데이터 보호 정책 생성

데이터 보호 정책은 애플리케이션 또는 AWS 서비스 간에 이동하는 중요한 정보를 감사, 비식별(마스킹 또는 수정), 거부(차단)하여 Amazon SNS 주제에 게시된 데이터를 보호하는 데 도움이 됩니다. AWS API, AWS CLI, AWS CloudFormation 또는 AWS Management Console을 사용하여 Amazon SNS에소 데이터 보호 정책을 생성할 수 있습니다. Amazon SNS 주제당 하나만 정의할 수 있습니다. 각 데이터 보호 정책은 하나 이상의 비식별 및 거부 명령문을 가질 수 있지만, 하나의 감사 명령문을 가질 수 있습니다.

주제

- 메시지 데이터 보안을 위한 데이터 보호 정책 생성(API)
- 메시지 데이터 보안을 위한 데이터 보호 정책 생성(CLI)
- 메시지 데이터를 보호하기 위한 데이터 보호 정책 생성(CloudFormation)

- 메시지 데이터 보안을 위한 데이터 보호 정책 생성(콘솔)
- 메시지 데이터 보안을 위한 데이터 보호 정책 생성(SDK)

메시지 데이터 보안을 위한 데이터 보호 정책 생성(API)

AWS 계정의 Amazon SNS 리소스 수와 크기는 제한되어 있습니다. 자세한 내용은 Amazon Simple Notification Service 엔드포인트 및 할당량을 참조하세요.

데이터 보호 정책 생성(AWS API)

AWS API를 사용하여 Amazon SNS 데이터 보호 정책을 생성할 수 있습니다.

Amazon SNS 주제와 함께 데이터 보호 정책 생성(AWS API)

표준 Amazon SNS 주제의 `DataProtectionPolicy` 속성을 사용합니다.

- `CreateTopic`

기존 Amazon SNS 주제에 대한 데이터 보호 정책을 생성 또는 검색(AWS API)

다음 작업 중 하나를 호출합니다.

- GetDataProtectionPolicy
- PutDataProtectionPolicy

메시지 데이터 보안을 위한 데이터 보호 정책 생성(CLI)

AWS 계정의 Amazon SNS 리소스 수와 크기는 제한되어 있습니다. 자세한 내용은 Amazon Simple Notification Service 엔드포인트 및 할당량을 참조하세요.

데이터 보호 정책 생성(AWS CLI)

AWS Command Line Interface를 사용하여 Amazon SNS 데이터 보호 정책을 생성할 수 있습니다.

Amazon SNS 주제와 함께 데이터 보호 정책 생성(AWS CLI)

이 옵션을 사용하여 표준 Amazon SNS 주제와 함께 새 데이터 보호 정책을 생성합니다.

- create-topic

기존 Amazon SNS 주제에 대한 데이터 보호 정책을 생성 또는 검색(AWS CLI)

다음 작업 중 하나를 호출합니다.

- get-data-protection-policy
- put-data-protection-policy

메시지 데이터를 보호하기 위한 데이터 보호 정책 생성(CloudFormation)

AWS 계정의 Amazon SNS 리소스 수와 크기는 제한되어 있습니다. 자세한 내용은 Amazon Simple Notification Service 엔드포인트 및 할당량을 참조하세요.

데이터 보호 정책 생성(CloudFormation)

AWS CloudFormation을 사용하여 Amazon SNS 데이터 보호 정책을 생성할 수 있습니다.

Amazon SNS 주제와 함께 데이터 보호 정책 생성(CloudFormation)

이 옵션을 사용하여 표준 Amazon SNS 주제와 함께 새 데이터 보호 정책을 생성합니다.

- AWS::SNS::Topic

메시지 데이터 보안을 위한 데이터 보호 정책 생성(콘솔)

AWS 계정의 Amazon SNS 리소스 수와 크기는 제한되어 있습니다. 자세한 내용은 Amazon Simple Notification Service 엔드포인트 및 할당량을 참조하세요.

Amazon SNS 주제와 함께 데이터 보호 정책 생성(콘솔)

표준 Amazon SNS 주제와 함께 새 데이터 보호 정책을 생성하려면 이 옵션을 사용합니다.

1. Amazon SNS 콘솔에 로그인합니다.
2. 주제를 선택하거나 새로운 주제를 생성합니다. 주제 생성에 대한 자세한 내용은 Amazon SNS 주제 생성을 참조하세요.
3. Create topic(주제 생성) 페이지의 Details(세부 정보) 섹션에서 Standard(표준)를 선택합니다.
   a. 주제의 이름을 입력합니다.
   b. (선택 사항) 주제의 표시 이름을 입력합니다.
4. Data protection policy(데이터 보호 정책)를 확장합니다.
5. 다음 Configuration mode(구성 모드)를 선택합니다.

- Basic(기본) - 간단한 메뉴를 사용하여 데이터 보호 정책을 정의합니다.

- Advanced(고급) - JSON을 사용하여 사용자 지정 데이터 보호 정책을 정의합니다.

6. (선택 사항) 자체 사용자 지정 데이터 식별자를 생성하려면 사용자 지정 데이터 식별자 구성 섹션을 확장하고 다음을 수행합니다.

   a. 사용자 지정 데이터 식별자에 고유한 이름을 입력합니다. 사용자 지정 데이터 식별자는 영숫자, 밑줄(_) 및 하이픈(-) 문자를 지원합니다. 최대 128자까지 지원됩니다. 이 이름은 관리형 데이터 식별자와 동일한 이름을 공유할 수 없습니다. 사용자 지정 데이터 식별자 제한의 전체 목록은 사용자 지정 데이터 식별자 제약 섹션을 참조하세요.

   b. 사용자 지정 데이터 식별자의 정규 표현식 (RegEx) 을 입력합니다. RegEx영숫자, RegEx 예약 문자 및 기호를 지원합니다. RegEx 최대 길이는 200자입니다. RegEx 이 명령이 너무 복잡하면 Amazon SNS에서 API 호출이 실패합니다. RegEx제한 사항의 전체 목록은 을 참조하십시오사용자 지정 데이터 식별자 제약.

   c. (선택 사항) 사용자 지정 데이터 식별자 추가를 선택하여 필요에 따라 데이터 식별자를 더 추가합니다. 각 데이터 보호 정책에 최대 10개의 사용자 지정 데이터 식별자가 지원됩니다.

7. 데이터 보호 정책에 추가하려는 명령문을 선택합니다. 감사, 비식별(마스킹 또는 수정) 및 거부(차단) 명령문 유형을 동일한 데이터 보호 정책에 추가할 수 있습니다.

   a. Add audit statement(감사 명세서 추가) - 감사할 중요한 데이터, 해당 데이터에 대해 감사할 메시지 비율, 감사 로그를 보낼 위치를 구성합니다.

      > **Note**
      >
      > 데이터 보호 정책 또는 주제당 하나의 감사 명령문만 허용됩니다.

      i. 감사하려는 중요한 데이터를 정의하려면 데이터 식별자를 선택합니다.

      ii. Audit sample rate(감사 샘플 비율)에서는 중요한 정보를 감사할 메시지의 백분율을 최대 99%까지 입력합니다.

      iii. Audit destination(감사 대상)에서 감사 결과를 보낼 AWS 서비스를 선택하고 사용하는 각 AWS 서비스의 대상 이름을 입력합니다. 다음 Amazon Web Services 중에서 선택할 수 있습니다.

         - Amazon CloudWatch — CloudWatch Logs는 AWS 표준 로깅 솔루션입니다. CloudWatch 로그를 사용하면 Logs Insights (여기 샘플 참조) 를 사용하여 로그 분석

을 수행하고 지표와 경보를 생성할 수 있습니다. CloudWatch 로그는 많은 서비스가 로그를 게시하는 곳이므로 하나의 솔루션으로 모든 로그를 쉽게 집계할 수 있습니다. CloudWatchAmazon에 대한 자세한 내용은 Amazon CloudWatch 사용 설명서를 참조하십시오.

- Amazon 데이터 파이어호스 — Firehose는 스플렁크로의 실시간 스트리밍과 추가 로그 분석을 위한 OpenSearch Amazon Redshift에 대한 요구를 충족합니다. Amazon Data Firehose에 대한 자세한 내용은 Amazon Data Firehose 사용 설명서를 참조하십시오.
- Amazon Simple Storage Service - Amazon S3는 보관을 위한 경제적인 로그 대상입니다. 몇 년 동안 로그를 보관해야 할 수도 있습니다. 이 경우 Amazon S3에 로그를 입력하여 비용을 절약할 수 있습니다. Amazon Simple Storage Service에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서를 참조하세요.

b. Add a de-identify statement(비식별 명령문 추가) - 메시지에서 비식별할 민감한 데이터, 해당 데이터를 마스킹 또는 수정할지 여부, 해당 데이터의 전송을 중지할 계정을 구성합니다.

i. 데이터 식별자에서는 비식별하려는 민감한 데이터를 선택합니다.

ii. Define this de-identify statement for(이 비식별 명령문 대상 정의)에서는 이 비식별 명령문이 적용되는 AWS 계정 또는 IAM 보안 주체를 선택합니다. 모든 AWS 계정 또는 계정 ID 또는 IAM 엔터티 ARN을 사용하는 특정 AWS 계정 또는 IAM 엔터티(계정 루트, 역할 또는 사용자)에 적용할 수 있습니다. 여러 ID 또는 ARN을 쉼표(,)를 사용하여 구분합니다.

지원되는 IAM 보안 주체는 다음과 같습니다.

- IAM account principals(IAM 계정 보안 주체) - 예: `arn:aws:iam::AWS-account-ID:root`.
- IAM role principals(IAM 역할 주체) - 예:`arn:aws:iam::AWS-account-ID:role/role-name`.
- IAM user principals(IAM 사용자 주체) - 예:`arn:aws:iam::AWS-account-ID:user/user-name`.

iii. De-identify Option(비식별 옵션)에서는 민감한 데이터를 비식별할 방법을 선택합니다. 지원되는 옵션은 다음과 같습니다.

- Redact(수정) - 데이터를 완전히 제거합니다. 예를 들어 이메일: `classified@amazon.com`은 이메일: 이 됩니다.
- Mask(마스크) - 데이터를 단일 문자로 바꿉니다. 예를 들어 이메일: `classified@amazon.com`은 이메일: **********이 됩니다.

iv. (선택 사항)필요에 따라 비식별 명령문을 계속 추가합니다.

c. Add deny statement(거부 명령문 추가) – 주제를 통해 이동하는 것을 방지할 중요한 데이터와 해당 데이터 전달을 방지할 보안 주체를 구성합니다.

i. data direction(데이터 방향)에서는 거부 명령문의 메시지 방향을 선택합니다.

- Inbound messages(인바운드 메시지) - 이 거부 설명을 주제로 전송된 메시지에 적용합니다.
- Outbound messages(아웃바운드 메시지) - 이 거부 설명을 해당 주제가 구독 엔드포인트에 전달하는 메시지에 적용합니다.

ii. 거부하려는 중요한 데이터를 정의하려면 data identifiers(데이터 식별자)를 선택합니다.

iii. 이 거부 명령문이 적용되는 IAM principals(IAM 보안 주체)를 선택합니다. 모든 AWS 계정 또는 계정 ID 또는 IAM 엔터티 ARN을 사용하는 특정 AWS 계정 또는 IAM 엔터티(계정 루트, 역할 또는 사용자)에 적용할 수 있습니다. 여러 ID 또는 ARN을 쉼표(,)를 사용하여 구분합니다. 지원되는 IAM 보안 주체는 다음과 같습니다.

- IAM account principals(IAM 계정 보안 주체) - 예: `arn:aws:iam::AWS-account-ID:root`.
- IAM 역할 주체 - 예:`arn:aws:iam::AWS-account-ID:role/role-name`.
- IAM 사용자 주체 - 예:`arn:aws:iam::AWS-account-ID:user/user-name`.

iv. (선택 사항) 필요에 따라 거부 명령문을 계속 추가합니다.

메시지 데이터 보안을 위한 데이터 보호 정책 생성(SDK)

AWS 계정의 Amazon SNS 리소스 수와 크기는 제한되어 있습니다. 자세한 내용은 Amazon Simple Notification Service 엔드포인트 및 할당량을 참조하세요.

데이터 보호 정책 생성(AWS SDK)

AWS SDK를 사용하여 Amazon SNS 데이터 보호 정책을 생성할 수 있습니다.

Amazon SNS 주제와 함께 데이터 보호 정책 생성(AWS SDK)

다음 옵션을 사용하여 표준 Amazon SNS 주제와 함께 새 데이터 보호 정책을 생성합니다.

Java

```
/**
```

```
 * For information regarding CreateTopic see this documentation topic:
 *
 * https://docs.aws.amazon.com/code-samples/latest/catalog/javav2-sns-src-main-java-
com-example-sns-CreateTopic.java.html
 */

public static String createSNSTopicWithDataProtectionPolicy(SnsClient snsClient,
 String topicName, String dataProtectionPolicy) {

    try {
        CreateTopicRequest request = CreateTopicRequest.builder()
                .name(topicName)
                .dataProtectionPolicy(dataProtectionPolicy)
                .build();

        CreateTopicResponse result = snsClient.createTopic(request);
        return result.topicArn();
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
    return "";
}
```

JavaScript

```
// Import required AWS SDK clients and commands for Node.js
import {CreateTopicCommand } from "@aws-sdk/client-sns";
import {snsClient } from "./libs/snsClient.js";

// Set the parameters
const params = { Name: "TOPIC_NAME", DataProtectionPolicy:
 "DATA_PROTECTION_POLICY" };

const run = async () => {
  try {
    const data = await snsClient.send(new CreateTopicCommand(params));
    console.log("Success.",  data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
};
```

```
run();
```

기존 Amazon SNS 주제에 대한 데이터 보호 정책을 생성 또는 검색(AWS SDK)

다음 옵션을 사용하여 표준 Amazon SNS 주제와 함께 새 데이터 보호 정책을 생성 또는 검색합니다.

Java

```
public static void putDataProtectionPolicy(SnsClient snsClient, String topicName,
 String dataProtectionPolicy) {

    try {
        PutDataProtectionPolicyRequest request =
 PutDataProtectionPolicyRequest.builder()
                .resourceArn(topicName)
                .dataProtectionPolicy(dataProtectionPolicy)
                .build();

        PutDataProtectionPolicyResponse result =
 snsClient.putDataProtectionPolicy(request);
        System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode()
                + "\n\nTopic " + request.resourceArn()
                + " DataProtectionPolicy " + request.dataProtectionPolicy());
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}


public static void getDataProtectionPolicy(SnsClient snsClient, String topicName) {

    try {
        GetDataProtectionPolicyRequest request =
 GetDataProtectionPolicyRequest.builder()
                .resourceArn(topicName)
                .build();

        GetDataProtectionPolicyResponse result =
 snsClient.getDataProtectionPolicy(request);
```

```
        System.out.println("\n\nStatus is " + result.sdkHttpResponse().statusCode()
        + "\n\nDataProtectionPolicy: \n\n" + result.dataProtectionPolicy());
    } catch (SnsException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

JavaScript

```
// Import required AWS SDK clients and commands for Node.js
import {PutDataProtectionPolicyCommand, GetDataProtectionPolicyCommand } from "@aws-
sdk/client-sns";
import {snsClient } from "./libs/snsClient.js";

// Set the parameters
const putParams = { ResourceArn: "TOPIC_ARN", DataProtectionPolicy:
 "DATA_PROTECTION_POLICY" };

const runPut = async () => {
  try {
    const data = await snsClient.send(new
 PutDataProtectionPolicyCommand(putParams));
    console.log("Success.",  data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
};
runPut();

// Set the parameters
const getParams = { ResourceArn: "TOPIC_ARN" };

const runGet = async () => {
  try {
    const data = await snsClient.send(new
 GetDataProtectionPolicyCommand(getParams));
    console.log("Success.",  data);
    return data; // For unit tests.
  } catch (err) {
    console.log("Error", err.stack);
  }
```

```
};
runGet();
```

Amazon SNS 데이터 보호 정책 삭제

AWS API, AWS CLI, AWS CloudFormation 또는 AWS Management Console을 사용하여 Amazon SNS 데이터 보호 정책을 삭제할 수 있습니다.

Amazon SNS 데이터 보호 정책에 대한 일반 정보는 데이터 보호 정책 이해 단원을 참조하세요.

AWS 계정의 Amazon SNS 데이터 보호 정책 리소스의 수와 크기는 제한되어 있습니다. 자세한 내용은 AWS 일반 참조의 Amazon SNS API 제한을 참조하세요.

주제

- 데이터 보호 정책 삭제(콘솔)
- 빈 JSON 문자열을 사용하여 데이터 보호 정책 삭제
- AWS CLI를 사용하여 데이터 보호 정책 삭제

데이터 보호 정책 삭제(콘솔)

관리형 데이터 보호 정책을 삭제(콘솔)

1. Amazon SNS 콘솔에 로그인합니다.
2. 삭제하려는 데이터 보호 정책이 포함된 주제를 선택합니다.
3. Edit(편집)을 선택합니다.
4. Data protection policy(데이터 보호 정책) 섹션을 확장합니다.
5. 제거하려는 데이터 보호 정책 명령문 옆에 있는 Remove(제거)를 선택합니다.
6. Save changes(변경 사항 저장)를 선택합니다.

빈 JSON 문자열을 사용하여 데이터 보호 정책 삭제

데이터 보호 정책을 빈 JSON 문자열로 업데이트하여 삭제할 수 있습니다.

AWS CLI를 사용하여 데이터 보호 정책 삭제

AWS CLI를 사용하여 데이터 보호 정책을 삭제할 수 있습니다.

```
//aws sns put-data-protection-policy --resource-arn topic-arn --data-
protection-policy ""
```

데이터 식별자

Amazon SNS는 기계 학습 및 패턴 일치를 비롯한 다양한 기준과 기술을 사용하여 중요한 데이터를 탐지합니다. 데이터 식별자로 통칭되는 이러한 기준과 기술을 통해 많은 국가와 지역에서 점점 늘어나고 있는 민감한 데이터 유형을 탐지할 수 있습니다. Amazon SNS 관리형 데이터 식별자는 금융 데이터, 개인 건강 정보(PHI) 및 개인 식별 정보(PII)를 보호하기 위해 사전 구성된 데이터 유형을 제공합니다. 또한 사용자 지정 데이터 식별자를 사용하여 특정 사용 사례에 맞는 자체 데이터 식별자를 생성할 수 있습니다.

주제

Amazon SNS에서 관리형 데이터 식별자 사용

주제

관리형 데이터 식별자란 무엇입니까?

Amazon SNS 관리형 데이터 식별자는 신용카드 번호, AWS 비밀 액세스 키 또는 특정 국가 또는 지역의 여권 번호와 같은 특정 유형의 민감한 데이터를 탐지하도록 설계되었습니다. 데이터 보호 정책을 생성할 때 이러한 식별자를 사용하여 주제를 통과하는 메시지를 분석하고 탐지되면 조치를 취하도록 Amazon SNS를 구성할 수 있습니다.

Amazon SNS는 관리형 데이터 식별자를 사용하여 다음과 같은 중요한 데이터 범주를 탐지할 수 있습니다.

- 보안 인증 정보(예: 프라이빗 키 또는 AWS 보안 액세스 키)
- 디바이스 식별자, 예: IP 주소 또는 MAC 주소
- 금융 정보(예: 신용 카드 번호)
- PHI 관련 건강 정보, 예: 건강 보험 또는 의료 식별 번호
- PII용 개인 정보, 예: 운전면허증 또는 주민등록번호

Amazon SNS는 각 범주 내에서 여러 유형의 중요한 데이터를 탐지할 수 있습니다. 이 섹션의 항목에서는 각 유형과 이를 탐지하기 위한 관련 요구 사항을 나열하고 설명합니다. 각 유형에 대해 데이터를 탐지하도록 설계된 관리되는 데이터 식별자의 고유 식별자(ID)도 나타냅니다. 데이터 보호 정책을 만들 때 이 ID를 사용하여 메시지 데이터 보호에서 탐지할 관리형 데이터 식별자를 포함할 수 있습니다.

키워드 요구 사항

특정 유형의 중요한 데이터를 탐지하기 위해 Amazon SNS는 데이터 근처에서 키워드를 검색합니다. 특정 유형의 데이터에 대한 경우 이 섹션의 후속 항목은 해당 데이터에 대한 특정 키워드 요구 사항을 나타냅니다.

키워드는 대/소문자를 구분하지 않습니다 또한 키워드에 공백이 포함된 경우 Amazon SNS는 공백을 포함하지 않거나 공백 대신 밑줄(_) 또는 하이픈(-)이 포함된 유사 키워드를 자동으로 찾습니다. 경우에 따라 Amazon SNS는 키워드의 일반적인 변형을 해결하기 위해 키워드를 확장하거나 축약하기도 합니다.

중요한 데이터 유형에 대한 Amazon SNS 관리형 데이터 식별자

다음 표는 Amazon SNS가 관리형 데이터 식별자를 사용하여 탐지할 수 있는 보안 인증 정보, 디바이스, 금융, 의료 및 개인 건강 정보(PHI) 유형을 나열하고 설명합니다. 이는 개인 식별 정보(PII)로도 인정될 수 있는 특정 유형의 데이터에 추가됩니다.

리전별 데이터 식별자에는 대시가 포함된 식별자 이름과 두 글자(ISO 3166-1 alpha-2) 코드가 필요합니다. 예: DriversLicense -US.

식별자	범주	국가/언어
BankAccountNumber	금융	DE, ES, FR, GB, IT
CepCode	개인	BR
Cnpj	개인	BR

식별자	범주	국가/언어
CpfCode	개인	BR
DriversLicense	개인	AT, AU, BE, BG, CA, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IT, LT, LU, LV, MT, NL, PL, PT, RO, SE, SI, SK, US
DrugEnforcementAgencyNumber	상태	미국
ElectoralRollNumber	개인	GB
HealthInsuranceCardNumber	상태	EU
HealthInsuranceClaimNumber	상태	미국
HealthInsuranceNumber	상태	FR
HealthcareProcedureCode	상태	미국
IndividualTaxIdentificationNumber	개인	미국
InseeCode	개인	FR
MedicareBeneficiaryNumber	상태	미국
NationalDrugCode	상태	미국
NationalIdentificationNumber	개인	DE, ES, IT
NationalInsuranceNumber	개인	GB
NationalProviderId	상태	미국
NhsNumber	상태	GB
NieNumber	개인	ES

식별자	범주	국가/언어
NifNumber	개인	ES
PassportNumber	개인	CA, DE, ES, FR, GB, IT, US
PermanentResidenceNumber	개인	CA
PersonalHealthNumber	상태	CA
PhoneNumber	개인	BR, DE, ES, FR, GB, IT, US
PostalCode	개인	CA
RgNumber	개인	BR
SocialInsuranceNumber	개인	CA
Ssn	개인	ES, US
TaxId	개인	DE, ES, FR, GB
ZipCode	개인	미국

언어/지역에 구애받지 않는 지원되는 식별자

식별자	범주
Address	개인
AwsSecretKey	보안 인증 정보
CreditCardExpiration	금융
CreditCardNumber	금융
CreditCardSecurityCode	금융
EmailAddress	개인

식별자	범주
IpAddress	개인
LatLong	개인
명칭	개인
OpenSshPrivateKey	보안 인증
PgpPrivateKey	보안 인증
PkcsPrivateKey	보안 인증
PuttyPrivateKey	보안 인증 정보
VehicleIdentificationNumber	개인

중요한 데이터 유형: 보안 인증 정보

다음 테이블은 Amazon SNS가 관리형 데이터 식별자를 사용하여 탐지할 수 있는 보안 인증 정보 유형을 나열하고 설명합니다.

탐지 유형	관리형 데이터 식별자 ID	필수 키워드	국가 및 리전
AWS 비밀 액세스 키	AwsSecretKey	aws_secret_access_key, credentials, secret access key, secret key, set-awscredential	모두
OpenSSH 프라이빗 키	OpenSshPrivateKey	아니요	모두
PGP 프라이빗 키	PgpPrivateKey	아니요	모두
프라이빗 키 암호화 표준(PKCS) 프라이빗 키	PkcsPrivateKey	아니요	모두

탐지 유형	관리형 데이터 식별자 ID	필수 키워드	국가 및 리전
PuTTY 프라이빗 키	PuttyPrivateKey	아니요	모두

보안 인증 데이터 유형을 위한 데이터 식별자 ARN

다음은 데이터 보호 정책에 추가할 수 있는 데이터 식별자의 Amazon 리소스 이름(ARN) 목록입니다.

보안 인증 데이터 식별자 ARN
arn:AWS:데이터 보호: :aws:데이터 식별자/ AwsSecretKey
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ OpenSshPrivateKey
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ PgpPrivateKey
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ PkcsPrivateKey
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ PuttyPrivateKey

중요한 데이터 유형: 디바이스

다음 테이블은 Amazon SNS가 관리형 데이터 식별자를 사용하여 탐지할 수 있는 디바이스 식별자 유형을 나열하고 설명합니다.

탐지 유형	관리형 데이터 식별자 ID	필수 키워드	국가 및 리전
IP 주소	IpAddress	아니요	모두

디바이스 데이터 유형을 위한 데이터 식별자 ARN

다음은 데이터 보호 정책에 추가할 수 있는 데이터 식별자의 Amazon 리소스 이름(ARN) 목록입니다.

디바이스 데이터 식별자 ARN
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ IpAddress

중요한 데이터 유형: 금융

다음 테이블은 Amazon SNS가 관리형 데이터 식별자를 사용하여 탐지할 수 있는 금융 정보 유형을 나열하고 설명합니다.

탐지 유형	관리형 데이터 식별자 ID	필수 키워드	추가 정보	국가 및 리전
은행 계좌 번호	BankAccountNumber BankAccountNumber-US	예, 은행 계좌 번호 키워드 단원을 참조하세요.	여기에는 국가 코드와 같은 요소를 포함하여 최대 34개의 영숫자로 구성된 국제 은행 계좌 번호(IBAN)가 포함됩니다.	프랑스, 독일, 이탈리아, 스페인, 영국
신용 카드 유효 기간	CreditCardExpiration	만료일, 만료월, 만료 연도, 만료, 만기	–	모두
신용 카드 마그네틱 스트립 데이터	CreditCardMagneticStripe	예, 카드 데이터, iso7813, 마그네틱, 마그네틱 스트라이프, 스트라이프, 스와이프 포함.	여기에는 1 및 2 트랙이 포함됩니다.	모두
신용 카드 번호	CreditCardNumber	계좌 번호, american express, amex, 은행 카드, 카드, 카드 번호, 카드 번호, cc #, ccn,	적발 시 데이터는 Luhn check 공식을 준수하는 13~19자리 시퀀스여야 하며, 아메리칸 익스프레	모두

탐지 유형	관리형 데이터 식별자 ID	필수 키워드	추가 정보	국가 및 리전
		체크 카드, 신용, 신용 카드#, dankort, 직불, 직불 카드, diners club, discover, electron, elo 인증 코드, japanese card bureau, jcb, mastercard, mc, pan, 결제 계좌 번호, 결제 카드 번호, pcn, union pay, visa	스, 단코트, 다이너스 클럽, 디스커버, 일렉트론, 일본 카드 뷰로 (JCB), 마스터카드, 비자 (아래 위첨자 링크) 와 같은 신용 카드 유형에 대해 표준 카드 번호 접두사를 사용해야 합니다. UnionPay	
신용 카드 인증 코드	CreditCardSecurityCode	카드 ID, 카드 식별 코드, 카드 식별 번호, 카드 보안 코드, 카드 인증 코드, 카드 인증 번호, 카드 인증 데이터, 카드 인증 값, cvc, cvc2, elo 인증 코드	–	모두

1. Amazon SNS 신용 카드 발급 기관이 공개 테스트를 위해 예약한 다음 시퀀스는 보고하지 않습니다.

   122000000000003, 2222405343248877, 2222990905257051, 2223007648726984, 2223577120017656, 30569309025904, 34343434343434, 3528000700000000, 3530111333300000, 3566002020360505, 36148900647913, 36700102000000, 371449635398431, 378282246310005, 378734493671000, 38520000023237, 4012888888881881, 4111111111111111, 4222222222222, 4444333322221111,

4462030000000000, 4484070000000000, 4911830000000, 4917300800000000, 4917610000000000, 4917610000000000003, 5019717010103742, 5105105105105100, 5111010030175156, 5185540810000019, 5200828282828210, 5204230080000017, 5204740009900014, 5420923878724339, 5454545454545454, 5455330760000018, 5506900490000436, 5506900490000444, 5506900510000234, 5506920809243667, 5506922400634930, 5506927427317625, 5553042241984105, 5555553753048194, 5555555555554444, 5610591081018250, 6011000990139424, 6011000400000000, 6011111111111117, 630490017740292441, 630495060000000000, 6331101999990016, 6759649826438453, 6799990100000000019, 76009244561.

은행 계좌 번호 키워드

국가 코드와 같은 요소를 포함하여 최대 34개의 영숫자로 구성된 국제 은행 계좌 번호(IBAN)를 탐지하려면 다음 키워드를 사용하세요.

국가 또는 리전	키워드			
프랑스	account code, account number, accountno #, accountnumber#, bban, code bancaire, compte bancaire, customer account id, customer account number, customer bank account id, iban, numéro de compte			
독일	account code, account number, accountno #, accountnu			

국가 또는 리전	키워드			
	mber#, bankleitzahl, bban, customer account id, customer account number, customer bank account id, geheimzahl, iban, kartennummer, kontonummer, kreditkartennummer, sepa			
이탈리아	account code, account number, accountno#, accountnumber#, bban, codice bancario, conto bancario, customer account id, customer account number, customer bank account id, iban, numero di conto			

국가 또는 리전	키워드			
스페인	account code, account number, accountno #, accountnu mber#, bban, código cuenta, código cuenta bancaria, cuenta cliente id, customer account ID, customer account number, customer bank account id, iban, número cuenta bancaria cliente, número cuenta cliente			
영국	account code, account number, accountno #, accountnu mber#, bban, customer account id, customer account number, customer bank account id, iban, sepa			

국가 또는 리전	키워드			
미국	은행 계좌,당좌 계좌, 예금 계좌, 적금 계좌, 당좌 예금 계좌			

금융 데이터 유형을 위한 데이터 식별자 ARN

다음은 데이터 보호 정책에 추가할 수 있는 데이터 식별자의 Amazon 리소스 이름(ARN) 목록입니다.

금융 데이터 식별자 ARN
arn:AWS:데이터 보호: :aws:데이터 식별자/ BankAccountNumber -DE
arn:aws: 데이터 보호: :aws: 데이터 식별자/ -ES BankAccountNumber
arn:aws: 데이터 보호: :aws: 데이터 식별자/ -FR BankAccountNumber
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -GB BankAccountNumber
arn: AWS: 데이터 보호: :AWS: 데이터 식별자/ - IT BankAccountNumber
arn:AWS: 데이터 보호: :AWS: 데이터 식별자/ -미국 BankAccountNumber
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ CreditCardExpiration
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ CreditCardNumber
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ CreditCardSecurityCode

중요한 데이터 유형: 보호 대상 건강 정보(PHI)

다음 테이블은 Amazon SNS가 관리형 데이터 식별자를 사용하여 탐지할 수 있는 보호 대상 건강 정보(PHI) 유형을 나열하고 설명합니다.

탐지 유형	관리형 데이터 식별자 ID	필수 키워드	국가 및 리전
마약단속국(DEA) 등록 번호	DrugEnforcementAgencyNumber	dea number, dea registration	미국
건강 보험 카드 번호(EHIC)	HealthInsuranceCardNumber	assicurazione sanitaria numero, carta assicurazione numero, carte d'assurance maladie, carte européenne d'assurance maladie, ceam, ehic, ehic#, finlandehicnumber#, gesundheitskarte, Hälsokort, 건강 카드, 보험, 건강 카드 번호, 건강 카드 번호 보험 카드 번호, krankenversicherungskarte, krankenversicherungsnummer, 의료 계좌 번호, numero conto medico, numéro d'assurance maladie, numéro de carte d'assurance, numéro de compte Medical, numéro de cueenta de médica, segurojeta , sairaanhoitokortin, sairausvakuutuskortti, sairausvakuutusnumero, sjukförsäkring nummer, sjukförsä	EU

탐지 유형	관리형 데이터 식별자 ID	필수 키워드	국가 및 리전
		kringskort, suomi ehic-numero, tarjeta de salud, terveysko rtti, tessero,versanus	
건강 보험 청구 번호 (HICN)	HealthInsuranceCla imNumber	health insurance claim number, hic no, hic no., hic number, hic#, hicn, hicn#., hicno#	미국
건강 보험 또는 의료 식별 번호	HealthInsuranceNum ber	carte d'assuré social, carte vitale, insurance card	FR
Healthcare Common Procedure Coding System(HCPCS) 코드	HealthcareProcedur eCode	current procedural terminology, hcpcs, healthcare common procedure coding system	미국
메디케어 수혜자 번호 (MBN)	MedicareBeneficiar yNumber	mbi, medicare beneficiary	미국
국가 의약품 코드 (NDC)	NationalDrugCode	national drug code, ndc	미국
국가 공급자 식별자 (NPI)	NationalProviderId	hipaa, n.p.i, national provider, npi	미국
국가 의료 서비스 (NHS) 번호	NhsNumber	national health service, NHS	GB

탐지 유형	관리형 데이터 식별자 ID	필수 키워드	국가 및 리전
개인 건강 번호(PHN)	PersonalHealthNumber	canada healthcare number, msp number, personal healthcare number, phn, soins de santé	CA

건강 보험 또는 의료 식별 번호 키워드

다양한 유형의 건강 보험 및 의료 식별 번호를 탐지하기 위해 Amazon SNS는 숫자 근처에 키워드가 있어야 합니다. 여기에는 유럽 건강 보험 카드 번호(EU, 핀란드), 건강 보험 번호(프랑스), 메디케어 수혜자 식별자(미국), 국가 보험 번호(영국), NHS 번호(영국) 및 개인 건강 번호(캐나다)가 포함됩니다.

다음 표에는 특정 국가 및 리전에서 Amazon SNS 인식하는 키워드가 나와 있습니다.

국가 또는 리전	키워드
캐나다	Canada healthcare number, msp number, personal healthcare number, phn, soins de santé
EU	assicurazione sanitaria numero, carta assicurazione numero, carte d’assurance maladie, carte européenne d'assurance maladie, ceam, ehic, ehic#, finlandehicnumber#, gesundheitskarte, hälsokort, health card, health card number, health insurance card, health insurance number, insurance card number, krankenversicherungskarte, krankenversicherungsnummer, medical account number, numero conto medico, numéro d’assurance maladie, numéro de carte d’assurance, numéro de compte medical, número de cuenta médica, número de seguro de salud, número de tarjeta de seguro, sairaanhoitokortin, sairausva

국가 또는 리전	키워드
	kuutuskortti, sairausvakuutusnumero, sjukförsäkring nummer, sjukförsäkringskort, suomi ehic-numero, tarjeta de salud, terveyskortti, tessera sanitaria assicurazione numero, versicherungsnummer
핀란드	ehic, ehic#, finland health insurance card, finlandehicnumber#, finska sjukförsäkringskort, hälsokort, health card, health card number, health insurance card, health insurance number, sairaanhoitokortin, sairaanhoitokortin, sairausvakuutuskortti, sairausvakuutusnumero, sjukförsäkring nummer, sjukförsäkringskort, suomen sairausvakuutuskortti, suomi ehic-numero, terveyskortti
프랑스	carte d'assuré social, carte vitale, insurance card
영국	국가 의료 서비스, NHS
미국	mbi, medicare beneficiary

보호 대상 건강 정보(PHI) 데이터 유형에 대한 데이터 식별자 ARN

다음은 PHI 데이터 보호 정책에서 사용할 수 있는 데이터 식별자 Amazon 리소스 이름(ARN)을 나열합니다.

PHI 데이터 식별자 ARN
arn: AWS: 데이터 보호: :AWS: 데이터 식별자/ - 미국 DrugEnforcementAgencyNumber
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -미국 HealthcareProcedureCode
arn: AWS: 데이터 보호:: AWS: HealthInsuranceCardNumber 데이터 식별자/ - EU

PHI 데이터 식별자 ARN
arn: AWS: 데이터 보호: :AWS: 데이터 식별자/ - 미국 HealthInsuranceClaimNumber
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -FR HealthInsuranceNumber
arn:AWS: 데이터 보호: :AWS: 데이터 식별자/ -미국 MedicareBeneficiaryNumber
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -미국 NationalDrugCode
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -GB NationalInsuranceNumber
arn: AWS: 데이터 보호: :AWS: 데이터 식별자/ - 미국 NationalProviderId
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -GB NhsNumber
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -CA PersonalHealthNumber

중요한 데이터 유형: 개인 식별 정보(PII)

다음 테이블은 Amazon SNS가 관리형 데이터 식별자를 사용하여 탐지할 수 있는 개인 식별 정보(PII) 유형을 나열하고 설명합니다.

탐지 유형	관리형 데이터 식별자 ID	필수 키워드	추가 정보	국가 및 리전
생년월일	DateOfBirth	dob, date of birth, birthdate, birth date, birthday, b-day, bday	지원에는 모든 숫자, 숫자 및 월 이름의 조합과 같은 대부분의 날짜 형식이 포함됩니다. 날짜 구성 요소는 공백, 슬래시(/) 또는 하이픈(-)으로 구분할 수 있습니다.	모두

탐지 유형	관리형 데이터 식별자 ID	필수 키워드	추가 정보	국가 및 리전
Código de Endereçamento Postal(CEP)	CepCode	cep, código de endereçamento postal, codigo de endereçamento postal	–	브라질
Cadastro Nacional da Pessoa Jurídica(CNPJ)	Cnpj	cadastro nacional da pessoa jurídica, cadastro nacional da pessoa juridica, cnpj	–	브라질
Cadastro de Pessoas Físicas(CPF)	CpfCode	Cadastro de pessoas fisicas, cadastro de pessoas físicas, cadastro de pessoa física, cadastro de pessoa fisica, cpf	–	브라질

탐지 유형	관리형 데이터 식별자 ID	필수 키워드	추가 정보	국가 및 리전
운전면허증 식별 번호	DriversLicense	예, 운전면허증 식별 번호 키워드 단원을 참조하세요.	–	호주, 오스트리아, 벨기에, 불가리아, 캐나다, 크로아티아, 사이프러스, 체코 공화국, 덴마크, 에스토니아, 핀란드, 프랑스, 독일, 그리스, 헝가리, 아일랜드, 이탈리아, 라트비아, 리투아니아, 룩셈부르크, 몰타, 네덜란드, 폴란드, 포르투갈, 루마니아, 슬로바키아, 슬로베니아, 스페인, 스웨덴, 영국, 미국
선거인단 번호	Electoral RollNumber	electoral#, electoral #, electoralnumber, electoral number, electoralroll#, electoral roll#, electoral roll #, electoral roll no., electoral roll number, electoralrollno	–	영국

탐지 유형	관리형 데이터 식별자 ID	필수 키워드	추가 정보	국가 및 리전
개인 납세자 식별	IndividualTaxIdentificationNumber	예, 납세자 식별 및 참조 번호 키워드 단원을 참조하세요.	–	미국
국립 통계 경제 연구소(INSEE)	InseeCode	예, 국가별 식별 번호 키워드 단원을 참조하세요.	–	프랑스
국적 식별 번호	NationalIdentificationNumber	예, 국가별 식별 번호 키워드 단원을 참조하세요.	여기에는 DNI(Documento Nacional de Identidad) 식별자(스페인), Codice fiscale 코드(이탈리아) 및 국가 신분증 번호(독일어)가 포함됩니다.	독일, 이탈리아, 스페인
National Insurance Number(NINO)	NationalInsuranceNumber	insurance no., insurance number, insurance #, national insurance number, nationalinsurance#, nationalinsurancenumber, nin, nino	–	영국

탐지 유형	관리형 데이터 식별자 ID	필수 키워드	추가 정보	국가 및 리전
Número de identidad de extranjero(NIE)	NieNumber	예, 납세자 식별 및 참조 번호 키워드 단원을 참조하세요.	–	스페인
Número de Identificación Fiscal(NIF)	NifNumber	예, 납세자 식별 및 참조 번호 키워드 단원을 참조하세요.	–	스페인
여권 번호	PassportNumber	예, 여권 번호 키워드 단원을 참조하세요.	–	캐나다, 프랑스, 독일, 이탈리아, 스페인, 영국, 미국

탐지 유형	관리형 데이터 식별자 ID	필수 키워드	추가 정보	국가 및 리전
영주권 번호	Permanent Residence Number	carte résident permanent , numéro carte résident permanent, numéro résident permanent , permanent resident card, permanent resident card number, permanent resident no, permanent resident no., permanent resident number, pr no, pr no., pr non, pr number, résident permanent no., résident permanent non	–	캐나다

탐지 유형	관리형 데이터 식별자 ID	필수 키워드	추가 정보	국가 및 리전
전화번호	PhoneNumber	브라질: 추가로 포함되는 키워드: cel, celular, fone, móvel, número residencial, numero residencial, telefone 기타: cell, contact, fax, fax number, mobile, phone, phone number, tel, telephone, telephone number	여기에는 미국 내 수신자 부담 전화 번호와 팩스 번호가 포함됩니다. 키워드가 데이터에 근접한 경우 번호에 국가 코드를 포함할 필요가 없습니다. 키워드가 데이터에 근접하지 않은 경우 숫자에 국가 코드가 포함되지 않아도 됩니다.	브라질, 캐나다, 프랑스, 독일, 이탈리아, 스페인, 영국, 미국
우편 번호	PostalCode	No	–	캐나다
Registro Geral(RG)	RgNumber	예, 국가별 식별 번호 키워드 단원을 참조하세요.	–	브라질
Social Insurance Number(SIN)	SocialInsuranceNumber	canadian id, numéro d'assurance sociale, social insurance number, sin	–	캐나다

탐지 유형	관리형 데이터 식별자 ID	필수 키워드	추가 정보	국가 및 리전
사회 보장 번호	Ssn	스페인 – número de la seguridad social, 사회 보장 번호, social security no. número de la seguridad social, 사회 보장 번호, socialsecurityno#, ssn, ssn# 미국 – social security, ss#, ssn	–	스페인, 미국
납세자 식별 번호 또는 참조 번호	TaxId	예, 납세자 식별 및 참조 번호 키워드 단원을 참조하세요.	여기에는 TIN(프랑스), Steueridentifikationsnummer(독일), CIF(스페인) 및 UTR TRN(영국)이 포함됩니다.	프랑스, 독일, 스페인, 영국
미국 우편번호	ZipCode	zip code, zip+4	–	미국
우편 주소	Address	No	키워드가 필수 항목은 아니지만 검색하려면 주소에 도시 또는 장소의 이름과 우편번호를 포함해야 합니다.	호주, 캐나다, 프랑스, 독일, 이탈리아, 스페인, 영국, 미국

탐지 유형	관리형 데이터 식별자 ID	필수 키워드	추가 정보	국가 및 리전
전자 메일 주소	EmailAddress	이메일, 이메일 주소, 이메일, 이메일 주소	–	모두
위성 항법 시스템(GPS) 좌표	LatLong	coordinate, coordinates, lat long, latitude longitude, location, position	Amazon SNS는 위도 및 경도 좌표가 쌍으로 저장되고 DD(10진수도) 형식인 경우 GPS 좌표를 탐지할 수 있습니다(예: 41.948614,-87.655311). 지원에는 10진수 도(DDM) 형식의 좌표(예: 41°56.9168'N 87°39.3187'W) 또는 도, 분, 초(DMS) 형식의 좌표(예: 41°56'55.0104"N 87°39'19.1196"W)가 포함되지 않습니다.	모두
전체 이름	Name	No	Amazon SNS는 전체 이름만 탐지할 수 있습니다. 라틴 문자 집합만 지원합니다.	모두

탐지 유형	관리형 데이터 식별자 ID	필수 키워드	추가 정보	국가 및 리전
차량 식별 번호 (VIN)	VehicleIdentificationNumber	Fahrgestellnummer, niv, numarul de identificare, numarul seriei de sasiu, serie sasiu, numer VIN, Número de Identificação do Veículo, Número de Identificación de Automóviles, numéro d'identification du véhicule, vehicle identification number, vin, VIN numeris	Amazon SNS는 17자 시퀀스로 구성되고 ISO 3779 및 3780 표준을 준수하는 VIN을 탐지할 수 있습니다. 이 표준은 전 세계에서 사용할 수 있도록 설계되었습니다.	모두

운전면허증 식별 번호 키워드

다양한 유형의 운전면허증 식별 번호를 탐지하기 위해 Amazon SNS에 해당 번호에 가까운 키워드가 있어야 합니다. 다음 표에는 특정 국가 및 리전에서 Amazon SNS 인식하는 키워드가 나와 있습니다.

국가 또는 리전	키워드
호주	dl# dl:, dl :, dlno# driver licence, driver license, driver permit, drivers lic., drivers licence, driver's licence, drivers license, driver's license, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit

국가 또는 리전	키워드
오스트리아	führerschein, fuhrerschein, führerschein republik österreich, fuhrerschein republik osterreich
벨기에	fuehrerschein, fuehrerschein- nr, fuehrersc heinnummer, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrersch einnummer, führerscheinnummer, numéro permis conduire, permis de conduire, rijbewijs, rijbewijsnummer
불가리아	превозно средство, свидетелство за управление на моторно, свидетелство за управление на мпс, сумпс, шофьорска книжка
캐나다	dl#, dl:, dlno#, driver licence, driver licences, driver license, driver licenses, driver permit, drivers lic., drivers licence, driver's licence, drivers licences, driver's licences, drivers license, driver's license, drivers licenses, driver's licenses, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit, permis de conduire
크로아티아	vozačka dozvola
사이프러스	άδεια οδήγησης
체코 공화국	číslo licence, číslo licence řidiče, číslo řidičskéh o průkazu, ovladače lic., povolení k jízdě, povolení řidiče, řidiči povolení, řidičský prúkaz, řidičský průkaz
덴마크	kørekort, kørekortnummer

국가 또는 리전	키워드
에스토니아	juhi litsentsi number, juhiloa number, juhiluba, juhiluba number
핀란드	ajokortin numero, ajokortti, förare lic., körkort, körkort nummer, kuljettaja lic., permis de conduire
프랑스	permis de conduire
독일	fuehrerschein, fuehrerschein- nr, fuehrersc heinnummer, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrersch einnummer, führerscheinnummer
그리스	δεια οδήγησης, adeia odigisis
헝가리	illesztőprogramok lic, jogosítvány, jogsi, licencszám, vezető engedély, vezetői engedély
아일랜드	ceadúnas tiomána
이탈리아	patente di guida, patente di guida numero, patente guida, patente guida numero
라트비아	autovadītāja apliecība, licences numurs, vadītāja apliecība, vadītāja apliecības numurs, vadītāja atļauja, vadītāja licences numurs, vadītāji lic.
리투아니아	vairuotojo pažymėjimas
룩셈부르크	fahrerlaubnis, führerschäin
몰타	liċenzja tas-sewqan
네덜란드	permis de conduire, rijbewijs, rijbewijsnummer

국가 또는 리전	키워드
폴란드	numer licencyjny, prawo jazdy, zezwolenie na prowadzenie
포르투갈	carta de condução, carteira de habilitação, carteira de motorist, carteira habilitação, carteira motorist, licença condução, licença de condução, número de licença, número licença, permissão condução, permissão de condução
루마니아	numărul permisului de conducere, permis de conducere
슬로바키아	číslo licencie, číslo vodičského preukazu, ovládače lic., povolenia vodičov, povolenie jazdu, povolenie na jazdu, povolenie vodiča, vodičský preukaz
슬로베니아	vozniško dovoljenje
스페인	carnet conducer, el carnet de conducer, licencia conducer, licencia de manejo, número carnet conducer, número de carnet de conducer, número de permiso conducer, número de permiso de conducer, número licencia conducer, número permiso conducer, permiso conducción, permiso conducer, permiso de conducción
스웨덴	ajokortin numero, dlno# ajokortti, drivere lic., förare lic., körkort, körkort nummer, körkortsn ummer, kuljettajat lic.

국가 또는 리전	키워드
영국	dl#, dl:, dlno#, driver licence, driver licences, driver license, driver licenses, driver permit, drivers lic., drivers licence, driver's licence, drivers licences, driver's licences, drivers license, driver's license, drivers licenses, driver's licenses, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit
미국	dl#, dl:, dlno#, driver licence, driver licences, driver license, driver licenses, driver permit, drivers lic., drivers licence, driver's licence, drivers licences, driver's licences, drivers license, driver's license, drivers licenses, driver's licenses, drivers permit, driver's permit, drivers permit number, driving licence, driving license, driving permit

국가별 식별 번호 키워드

다양한 유형의 주민등록번호를 탐지하기 위해 Amazon SNS에 해당 번호에 가까운 키워드가 있어야 합니다. 여기에는 Documento Nacional de Identidad(DNI) 식별자(스페인), 프랑스 국립 통계 및 경제 연구소(INSEE) 코드, 독일 신분증 번호 및 Registro Geral(RG) 번호(브라질)이 포함됩니다.

다음 표에는 특정 국가 및 리전에서 Amazon SNS 인식하는 키워드가 나와 있습니다.

국가 또는 리전	키워드
브라질	registro geral, rg
프랑스	assurance sociale, carte nationale d’identit é, cni, code sécurité sociale, French social security number, fssn#, insee, insurance number, national id number, nationalid#, numéro d'assurance, sécurité sociale, sécurité

국가 또는 리전	키워드
	sociale non., sécurité sociale numéro, social, social security, social security number, socialsecuritynumber, ss#, ssn, ssn#
독일	ausweisnummer, id number, identification number, identity number, insurance number, personal id, personalausweis
이탈리아	codice fiscal, dati anagrafici, ehic, health card, health insurance card, p. iva, partita i.v.a., personal data, tax code, tessera sanitaria
스페인	dni, dni#, dninúmero#, documento nacional de identidad, identidad único, identidadúnico#, insurance number, national identification number, national identity, nationalid#, nationali dno#, número nacional identidad, personal identification number, personal identity no, unique identity number, uniqueid#

여권 번호 키워드

다양한 유형의 여권 번호를 탐지하기 위해 Amazon SNS에 해당 번호에 가까운 키워드가 있어야 합니다. 다음 표에는 특정 국가 및 리전에서 Amazon SNS 인식하는 키워드가 나와 있습니다.

국가 또는 리전	키워드
캐나다	passeport, passeport#, passport, passport#, passportno, passportno#
프랑스	numéro de passeport, passeport, passeport #, passeport #, passeportn °, passeport n °, passeportNon, passeport non

국가 또는 리전	키워드
독일	ausstellungsdatum, ausstellungsort, geburtsdatum, passport, passports, reisepass, reisepass–nr, reisepassnummer
이탈리아	italian passport number, numéro passeport, numéro passeport italien, passaporto, passaporto italiana, passaporto numero, passport number, repubblica italiana passaporto
스페인	españa pasaporte, libreta pasaporte, número pasaporte, pasaporte, passport, passport book, passport no, passport number, spain passport
영국	passeport #, passeport n °, passeportNon, passeport non, passeportn °, passport #, passport no, passport number, passport#, passportid
미국	passport, travel document

납세자 식별 및 참조 번호 키워드

다양한 유형의 납세자 식별 및 참조 번호를 탐지하기 위해 Amazon SNS는 숫자 근처에 키워드가 있어야 합니다. 다음 표에는 특정 국가 및 리전에서 Amazon SNS 인식하는 키워드가 나와 있습니다.

국가 또는 리전	키워드
브라질	cadastro de pessoa física, cadastro de pessoa fisica, cadastro de pessoas físicas, cadastro de pessoas fisicas, cadastro nacional da pessoa jurídica, cadastro nacional da pessoa juridica, cnpj, cpf
프랑스	numéro d'identification fiscale, tax id, tax identification number, tax number, tin, tin#

국가 또는 리전	키워드
독일	identifikationsnummer, steuer id, steueride ntifikationsnummer, steuernummer, tax id, tax identification number, tax number
스페인	cif, cif número, cifnúmero#, nie, nif, número de contribuyente, número de identidad de extranjero, número de identificación fiscal, número de impuesto corporativo, personal tax number, tax id, tax identification number, tax number, tin, tin#
영국	paye, tax id, tax id no., tax id number, tax identification, tax identification#, tax no., tax number, tax reference, tax#, taxid#, temporary reference number, tin, trn, unique tax reference, unique taxpayer reference, utr
미국	개인 납세자 식별 번호(itin, i.t.i.n.)

개인 식별 정보(PII) 데이터 식별자 ARN

다음 표는 데이터 보호 정책에 추가할 수 있는 데이터 식별자의 Amazon 리소스 이름(ARN) 목록입니다.

PII 및 데이터 식별자 ARN
arn:aws:dataprotection::aws:data-identifier/Address
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -BR CepCode
arn:aws:dataprotection::aws:data-identifier/Cnpj-BR
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -BR CpfCode
arn: AWS: 데이터 보호: :aws: 데이터 식별자/DateOfBirth

PII 및 데이터 식별자 ARN
arn:AWS: 데이터 보호: :AWS: 데이터 식별자/ -AT DriversLicense
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -AU DriversLicense
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -BE DriversLicense
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -BG DriversLicense
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -CA DriversLicense
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -CY DriversLicense
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -CZ DriversLicense
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -DE DriversLicense
arn:aws: 데이터 보호: :aws: 데이터 식별자/ -DK DriversLicense
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -EE DriversLicense
arn:aws: 데이터 보호: :aws: 데이터 식별자/ -ES DriversLicense
arn: AWS: 데이터 보호: :aws: 데이터 식별자/ FI DriversLicense
arn: AWS: 데이터 보호: :aws: 데이터 식별자/ -FR DriversLicense
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -GB DriversLicense
arn: AWS: 데이터 보호: :aws: 데이터 식별자/ -GR DriversLicense
arn:AWS: 데이터 보호: :AWS: 데이터 식별자/ -HR DriversLicense
arn: AWS: 데이터 보호:: AWS: DriversLicense 데이터 식별자/ -HU
arn:aws: 데이터 보호: :aws: 데이터 식별자/ - IE DriversLicense
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -IT DriversLicense
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -LT DriversLicense

PII 및 데이터 식별자 ARN
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -LU DriversLicense
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -LV DriversLicense
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -MT DriversLicense
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -NL DriversLicense
arn:aws: 데이터 보호: :aws: 데이터 식별자/ -PL DriversLicense
arn:aws: 데이터 보호: :aws: 데이터 식별자/ -PT DriversLicense
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -RO DriversLicense
arn:aws: 데이터 보호: :aws: 데이터 식별자/ -SE DriversLicense
arn:aws: 데이터 보호: :aws: 데이터 식별자/ -SI DriversLicense
arn:aws: 데이터 보호: :aws: 데이터 식별자/ -SK DriversLicense
arn:AWS: 데이터 보호: :AWS: 데이터 식별자/ -미국 DriversLicense
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -GB ElectoralRollNumber
arn: AWS: 데이터 보호: :aws: 데이터 식별자/EmailAddress
arn: AWS: 데이터 보호: :AWS: 데이터 식별자/ - 미국 IndividualTaxIdentificationNumber
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -FR InseeCode
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ LatLong
arn:aws:dataprotection::aws:data-identifier/Name
arn:aws: 데이터 보호: :aws: 데이터 식별자/ - DE NationalIdentificationNumber
arn:aws: 데이터 보호: :aws: 데이터 식별자/ -ES NationalIdentificationNumber
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -IT NationalIdentificationNumber

PII 및 데이터 식별자 ARN
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -ES NieNumber
arn:aws: 데이터 보호: :aws: 데이터 식별자/ -ES NifNumber
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -CA PassportNumber
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ - DE PassportNumber
arn:aws: 데이터 보호: :aws: 데이터 식별자/ -ES PassportNumber
arn:aws: 데이터 보호: :aws: 데이터 식별자/ -FR PassportNumber
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -GB PassportNumber
arn: AWS: 데이터 보호: :AWS: 데이터 식별자/ - IT PassportNumber
arn:AWS: 데이터 보호: :AWS: 데이터 식별자/ -미국 PassportNumber
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -CA PermanentResidenceNumber
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -BR PhoneNumber
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -DE PhoneNumber
arn:aws: 데이터 보호: :aws: 데이터 식별자/ -ES PhoneNumber
arn:aws: 데이터 보호: :aws: 데이터 식별자/ -FR PhoneNumber
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -GB PhoneNumber
arn: AWS: 데이터 보호: :AWS: 데이터 식별자/ - IT PhoneNumber
arn:AWS: 데이터 보호: :AWS: 데이터 식별자/ -미국 PhoneNumber
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -CA PostalCode
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -BR RgNumber
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -CA SocialInsuranceNumber

PII 및 데이터 식별자 ARN
arn:aws:dataprotection::aws:data-identifier/Ssn-ES
arn:aws:dataprotection::aws:data-identifier/Ssn-US
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ - DE TaxId
arn:aws: 데이터 보호: :aws: 데이터 식별자/ -ES TaxId
arn:aws: 데이터 보호: :aws: 데이터 식별자/ -FR TaxId
arn:AWS: 데이터 보호: :aws: 데이터 식별자/ -GB TaxId
arn: AWS: 데이터 보호: :aws: 데이터 식별자/VehicleIdentificationNumber
arn: AWS: 데이터 보호: :AWS: 데이터 식별자/ - 미국 ZipCode

Amazon SNS에서 사용자 지정 데이터 식별자 사용

주제

사용자 지정 데이터 식별자란?

사용자 지정 데이터 식별자(CDI)를 통해 데이터 보호 정책에 사용할 수 있는 사용자 지정 정규 표현식을 정의할 수 있습니다. 사용자 지정 데이터 식별자를 사용하면 관리형 데이터 식별자로는 제공할 수 없는 비즈니스별 개인 식별 정보(PII) 사용 사례를 대상으로 지정할 수 있습니다. 예를 들어 사용자 지정 데이터 식별자를 사용하여 회사별 직원 ID를 찾을 수 있습니다. 사용자 지정 데이터 식별자는 관리형 데이터 식별자와 함께 사용할 수 있습니다.

데이터 보호 정책에서 사용자 지정 데이터 식별자 사용

다음 데이터 보호 정책은 Amazon SNS 주제에 회사별 직원 ID가 포함된 페이로드를 탐지한 다음 해시 기호(#)를 사용하여 해당 ID를 마스킹하도록 지시합니다.

1. 데이터 보호 정책 내에 Configuration 블록을 생성합니다.
2. 사용자 지정 데이터 식별자에 Name을 입력합니다. 예를 들어 **EmployeeId**입니다.
3. 사용자 지정 데이터 식별자에 Regex을 입력합니다. 예를 들어 **EID-\d{9}-US**입니다.
4. 정책 설명에서 다음 사용자 지정 데이터 식별자를 참조하세요.

```
{
 "Name": "__example_data_protection_policy",
 "Description": "Example data protection policy",
 "Version": "2021-06-01",
 "Configuration": {
   "CustomDataIdentifier": [
     {"Name": "EmployeeId", "Regex": "EID-\d{9}-US"}
   ]
 },
 "Statement": [
   {
     "DataDirection": "Inbound",
     "Principal": ["*"],
     "DataIdentifier": [
        "EmployeeId"
     ],
     "Operation": {
       "Deidentify": {
         "MaskConfig": {
           "MaskWithCharacter": "#"
         }
       }
     }
   }
 ]
}
```

5. (선택 사항) 필요에 따라 사용자 지정 데이터 식별자를 Configuration 블록에 계속 추가합니다. 데이터 보호 정책은 현재 최대 10개의 사용자 지정 데이터 식별자를 지원합니다.

사용자 지정 데이터 식별자 제약

Amazon SNS 사용자 지정 데이터 식별자에는 다음과 같은 제한 사항이 있습니다.

- 각 데이터 보호 정책에 최대 10개의 사용자 지정 데이터 식별자가 지원됩니다.
- 사용자 지정 데이터 식별자 이름의 최대 길이는 128자입니다. 다음 문자가 지원됩니다.

  - 영숫자: (a-zA-Z0-9)
  - 기호: ('_' | '-')
- RegEx의 최대 길이는 200자입니다. 다음 문자가 지원됩니다.
  - 영숫자: (a-zA-Z0-9)
  - 기호: ('_' | '#' | '=' | '@' |'/' | ';' | ',' | '-' | ' ')
  - RegEx 예약 문자: ('^' | '$' | '?' | '[' | ']' | '{' | '}' | '|' | '\\' | '*' | '+' | '.')
- 사용자 지정 데이터 식별자는 관리형 데이터 식별자와 동일한 이름을 공유할 수 없습니다.
- 각 Amazon SNS 주제에 대한 모든 데이터 보호 정책에는 사용자 지정 데이터 식별자를 지정해야 합니다.

Amazon SNS 메시지 전송

이 섹션에서는 메시지 전송 방식을 설명합니다.

주제

Amazon SNS 원시 메시지 전송

Amazon Data Firehose, Amazon SQS 및 HTTP/S 엔드포인트에서 메시지의 JSON 형식을 처리하지 않도록 하기 위해 Amazon SNS는 원시 메시지 전송을 허용합니다.

- Amazon Data Firehose 또는 Amazon SQS 엔드포인트에 대해 원시 메시지 전송을 활성화하면 게시된 메시지에서 모든 Amazon SNS 메타데이터가 제거되고 메시지가 있는 그대로 전송됩니다.
- HTTP/S 엔드포인트에 대해 원시 메시지 전송을 활성화하면 값이 `true`로 설정된 HTTP 헤더 `x-amz-sns-rawdelivery`가 메시지에 추가됩니다. 이는 메시지가 JSON 형식 지정 없이 게시되었음을 나타냅니다.
- HTTP/S 엔드포인트에 대해 원시 메시지 전송을 활성화하면 메시지 본문, 클라이언트 IP 및 필수 헤더가 전송되지만, 메시지 속성을 지정하면 전송되지 않습니다.
- Firehose 엔드포인트에 대해 원시 메시지 전송을 활성화하면 메시지 본문이 전달됩니다. 메시지 속성을 지정하면 전송되지 않습니다.

AWS SDK를 사용하여 원시 메시지 전송을 활성화하려면 `SetSubscriptionAttribute` API 작업을 사용하고 속성 값을 로 설정해야 합니다. `RawMessageDelivery` `true`

AWS Management Console을 사용한 원시 메시지 전송 활성화

1. Amazon SNS 콘솔에 로그인합니다.
2. 탐색 창에서 주제(Topics)를 선택합니다.

3. 주제 페이지에서 Firehose, Amazon SQS 또는 HTTP/S 엔드포인트에 구독하는 주제를 선택합니다.
4. ***MyTopic***페이지의 구독 섹션에서 구독을 선택하고 편집을 선택합니다.
5. Edit ***EXAMPLE1-23bc-4567-d890-ef12g3hij456***(예제 1-23bc-4567-d890-ef12g3hij456 편집) 페이지의 세부 정보 섹션에서 Enable raw message delivery(원시 메시지 전송 활성화)를 선택합니다.
6. 변경 사항 저장를 선택합니다.

메시지 형식 예제

다음 예에서는 동일한 메시지가 동일한 Amazon SQS 대기열로 두 번 전송됩니다. 유일한 차이점은 원시 메시지 전송이 첫 번째 메시지에 대해 사용 중지되고 두 번째 메시지에 대해 사용된다는 것입니다.

- 원시 메시지 전송이 사용 중지됨

```
{
  "Type": "Notification",
  "MessageId": "dc1e94d9-56c5-5e96-808d-cc7f68faa162",
  "TopicArn": "arn:aws:sns:us-east-2:111122223333:ExampleTopic1",
  "Subject": "TestSubject",
  "Message": "This is a test message.",
  "Timestamp": "2021-02-16T21:41:19.978Z",
  "SignatureVersion": "1",
  "Signature":
 "FMG5tlZhJNHLHUXvZgtZzlk24FzVa7oX0T4P03neeXw8ZEXZx6z35j2FOTuNYShn2h0bKNC/
zLTnMyIxEzmi2X1shOBWsJHkrW2xkR58ABZF+4uWHEE73yDVR4SyYAikP9jstZzDRm
+bcVs8+T0yaLiEGLrIIIL4esi1llhIkgErCuy5btPcWXBdio2fpCRD5x9oR6gmE/
rd5O7lX1c1uvnv4r1Lkk4pqP2/iUfxFZva1xLSRvgyfm6D9hNklVyPfy
+7TalMD0lzmJuOrExtnSIbZew3foxgx8GT+lbZkLd0ZdtdRJlIyPRP44eyq78sU0Eo/
LsDr0Iak4ZDpg8dXg==",
  "SigningCertURL": "https://sns.us-east-2.amazonaws.com/
SimpleNotificationService-010a507c1833636cd94bdb98bd93083a.pem",
  "UnsubscribeURL": "https://sns.us-east-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
east-2:111122223333:ExampleTopic1:e1039402-24e7-40a3-a0d4-797da162b297"
}
```

- 원시 메시지 전송이 사용됨

```
This is a test message.
```

Amazon SQS 구독의 메시지 속성 및 원시 메시지 전송

Amazon SNS는 메시지 속성 전송을 지원하므로 메시지에 대한 타임스탬프, 지리 공간 데이터, 서명 및 식별자와 같은 정형 메타데이터 항목을 제공할 수 있습니다. 원시 메시지 전송이 활성화된 Amazon SQS 구독의 경우 최대 10개의 메시지 속성을 전송할 수 있습니다. 10개 이상의 메시지 속성을 전송하려면 원시 메시지 전송을 비활성화해야 합니다. 하지만 Amazon SNS는 원시 메시지 전송이 활성화된 상태에서 Amazon SQS 구독으로 향하는 메시지 속성이 10개 이상인 메시지를 삭제하여 클라이언트 측 오류로 간주합니다.

다른 계정의 Amazon SQS 대기열로 Amazon SNS 메시지 전송

이 문서에서는 다른 계정의 Amazon SQS 대기열에 하나 이상의 구독을 가진 Amazon SNS 주제에 알림을 게시하는 방법을 설명합니다. 동일한 계정에서와 같은 방법으로 주제 및 대기열을 설정합니다 (Amazon SQS 대기열로 팬아웃 참조). 주요 차이점은 구독 확인을 처리하는 방법이며 주제에 대한 대기열을 구독하는 방법에 따라 다릅니다.

대기열 소유자가 구독을 생성할 때 확인이 자동으로 이루어지므로, 가능하다면 대기열 소유자가 구독 생성 섹션에서 참조하는 단계를 따르는 것이 좋습니다.

> **Note**
>
> Amazon SQS 대기열의 메시지 양이 많은 경우 대기열 소유자가 구독을 생성하는 것이 좋습니다.

주제

- 대기열 소유자 구독 생성
- 대기열을 소유하지 않지만 구독을 생성하는 사용자
- 구독 취소 요청에 대한 인증을 요구하도록 구독을 강제하려면 어떻게 해야 합니까?

대기열 소유자 구독 생성

Amazon SQS 대기열을 생성한 계정이 대기열 소유자입니다. 대기열 소유자가 구독을 생성할 경우 구독 확인은 필요하지 않습니다. 대기열은 `Subscribe` 작업이 완료되자마자 주제로부터의 알림 수신을 시작합니다. 대기열 소유자가 주제 소유자의 주제를 구독하려면 해당 주제 소유자가 대기열 소유자의 계정에 해당 주제에 대한 `Subscribe` 작업을 호출할 수 있는 권한을 부여해야 합니다.

1단계: AWS Management Console을 사용하여 주제 정책을 설정하려면

1. Amazon SNS 콘솔에 로그인합니다.
2. 탐색 창에서 주제(Topics)를 선택합니다.
3. 주제를 선택한 다음 편집을 선택합니다.
4. ***MyTopic*** 편집 페이지에서 액세스 정책 섹션을 확장합니다.
5. 다음 정책을 입력합니다.

```
{
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": {
            "AWS": "111122223333"
         },
         "Action": "sns:Subscribe",
         "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic"
      }
   ]
}
```

이 정책은 계정 111122223333에 계정 123456789012에서 MyTopic에 대해 sns:Subscribe를 호출할 수 있는 권한을 부여합니다.

계정 111122223333에 대한 자격 증명을 가진 사용자가 MyTopic을 구독할 수 있습니다. 이 권한을 사용하면 계정 ID에서 해당 IAM 사용자/역할에 권한을 위임할 수 있습니다. 루트 계정 또는 관리자 사용자만 sns:Subscribe를 호출할 수 있습니다. IAM 사용자/역할도 대기열에서 구독하려면 sns:subscribe 권한이 있어야 합니다.

6. 변경 사항 저장을 선택합니다.

계정 111122223333에 대한 자격 증명을 가진 사용자가 MyTopic을 구독할 수 있습니다.

2단계: AWS Management Console을 사용하여 다른 AWS 계정의 주제에 Amazon SQS 대기열 구독을 추가하려면

시작하기 전에 주제 및 대기열에 대한 ARN이 있고 대기열에 메시지를 보낼 수 있는 주제에 대한 권한을 부여했는지 확인합니다.

1. Amazon SQS 콘솔에 로그인합니다.
2. 탐색 창에서 대기열(Queues)을 선택합니다.
3. 대기열 목록에서 Amazon SNS 주제를 구독할 대기열을 선택합니다.
4. Subscribe to Amazon SNS topic(Amazon SNS 주제 구독)을 선택합니다.
5. 이 대기열 메뉴에 사용 가능한 Amazon SNS 주제 지정(Specify an Amazon SNS topic available for this queue menu)에서 대기열에 대해 Amazon SNS 주제(Amazon SNS topic)를 선택합니다.
6. Amazon SNS 주제 ARN 입력(Enter Amazon SNS topic ARN)을 선택한 다음 주제의 Amazon 리소스 이름(ARN)을 입력합니다.
7. 저장을 선택합니다.

> **Note**
>
> - 서비스와 통신할 수 있으려면 대기열에 Amazon SNS에 대한 권한이 있어야 합니다.
> - 사용자가 대기열의 소유자이므로 구독을 확인할 필요가 없습니다.

대기열을 소유하지 않지만 구독을 생성하는 사용자

구독을 생성하지만 대기열 소유자가 아닌 사용자는 구독을 확인해야 합니다.

`Subscribe` 작업을 사용하면 Amazon SNS에서 대기열에 구독 확인을 보냅니다. 구독이 Amazon SNS 콘솔에 표시되고 해당 구독 ID가 확인 보류 중으로 설정되어 있습니다.

구독을 확인하려면 대기열로부터의 메시지를 읽을 수 있는 권한이 있는 사용자가 구독 확인 URL을 검색해야 하며, 구독 소유자는 구독 확인 URL을 사용하여 구독을 확인해야 합니다. 구독이 확인되기 전에는 주제에 게시된 알림은 대기열로 전송되지 않습니다. 구독을 확인하려면 Amazon SQS 콘솔 또는 `ReceiveMessage` 작업을 사용합니다.

> **Note**
>
> 주제에 엔드포인트를 구독하기 전에 대기열에 대한 `sqs:SendMessage` 권한을 설정하여 대기열이 주제로부터 메시지를 받을 수 있는지 확인합니다. 자세한 내용은 2단계: Amazon SQS 대기열에 메시지를 전송하도록 Amazon SNS 주제에 권한 부여 섹션을 참조하세요.

1단계: AWS Management Console을 사용하여 다른 AWS 계정의 주제에 Amazon SQS 대기열 구독을 추가하려면

시작하기 전에 주제 및 대기열에 대한 ARN이 있고 대기열에 메시지를 보낼 수 있는 주제에 대한 권한을 부여했는지 확인합니다.

1. Amazon SNS 콘솔에 로그인합니다.
2. 탐색 창에서 구독을 선택합니다.
3. 구독 페이지에서 구독 생성을 선택합니다.
4. 구독 생성 페이지의 세부 정보 섹션에서 다음을 수행합니다.
   a. 주제 ARN에 주제의 ARN을 입력합니다.
   b. 프로토콜에서 Amazon SQS를 선택합니다.
   c. 엔드포인트에 대기열의 ARN을 입력합니다.
   d. 구독 생성을 선택합니다.

> **Note**
>
> - 서비스와 통신할 수 있으려면 대기열에 Amazon SNS에 대한 권한이 있어야 합니다.

다음은 Amazon SNS 주제가 Amazon SQS 대기열에 메시지를 전송하도록 허용하는 정책 문 예제입니다.

```
{
   "Sid": "Stmt1234",
   "Effect": "Allow",
   "Principal": "*",
   "Action": "sqs:SendMessage",
   "Resource": "arn:aws:sqs:us-west-2:111111111111:QueueName",
   "Condition": {
      "ArnEquals": {
         "aws:SourceArn": "arn:aws:sns:us-west-2:555555555555:TopicName"
      }
   }
}
```

2단계: AWS Management Console을 사용하여 구독을 확인하려면

1. Amazon SQS 콘솔에 로그인합니다.
2. 주제에 대한 대기 중인 구독을 보유한 대기열을 선택합니다.
3. Send and receive messages(메시지 보내기 및 받기)를 선택한 다음 Poll for messages(메시지 폴링)를 선택합니다.

   대기열에서 구독 확인 메시지가 수신됩니다.
4. 본문 열에서 다음을 수행합니다.

   a. 추가 정보를 선택합니다.

   b. 메시지 세부 정보(Message Details) 대화 상자에서 SubscribeURL 값을 찾아서 확인합니다. 이 링크는 구독 링크입니다(아래 예). API 토큰 유효성 검사에 대한 자세한 내용은 Amazon SNS API 참조의 `ConfirmSubscription` 섹션을 참조하세요.

   ```
   https://sns.us-west-2.amazonaws.com/?
   Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-
   east-2:123456789012:MyTopic&Token=2336412f37fb...
   ```

   c. 구독 확인 링크를 기록해 둡니다. URL을 대기열 소유자에서 구독 소유자에게로 전달해야 합니다. 구독 소유자는 URL을 Amazon SNS 콘솔에 입력해야 합니다.
5. 구독 소유자로 Amazon SNS 콘솔에 로그인하여 구독 소유자가 확인을 수행합니다.
6. 관련 주제를 선택합니다.
7. 주제의 구독 목록 테이블에서 관련 구독을 선택합니다. '확인 보류 중(Pending confirmation)'으로 표시됩니다.
8. 구독 확인(Confirm subscription)을 선택합니다.
9. 구독 확인 링크를 요청하는 모달이 나타납니다. 구독 확인 링크를 붙여넣습니다.
10. 모달에서 구독 확인(Confirm subscription)을 선택합니다.

    XML 응답이 표시됩니다. 예를 들면 다음과 같습니다.

```
<ConfirmSubscriptionResponse>
   <ConfirmSubscriptionResult>
      <SubscriptionArn>arn:aws:sns:us-east-2:123456789012:MyTopic:1234a567-
bc89-012d-3e45-6fg7h890123i</SubscriptionArn>
   </ConfirmSubscriptionResult>
   <ResponseMetadata>
```

```
      <RequestId>abcd1efg-23hi-jkl4-m5no-p67q8rstuvw9</RequestId>
   </ResponseMetadata>
</ConfirmSubscriptionResponse>
```

구독된 대기열은 주제로부터 메시지를 수신할 준비가 되었습니다.

11. (선택 사항) Amazon SNS 콘솔에서 주제 구독을 보는 경우 확인 보류 중 메시지가 구독 ID 열의 구독 ARN으로 대체되었음을 확인할 수 있습니다.

구독 취소 요청에 대한 인증을 요구하도록 구독을 강제하려면 어떻게 해야 합니까?

구독 소유자는 구독 확인 시 `AuthenticateOnUnsubscribe` 플래그를 true로 설정해야 합니다.

- `AuthenticateOnUnsubscribe`는 대기열 소유자가 구독을 생성할 때 자동으로 true로 설정됩니다.
- 인증 없이 구독 확인 링크로 이동하는 경우 `AuthenticateOnUnsubscribe`를 true로 설정할 수 없습니다.

Amazon SNS 메시지를 다른 리전의 Amazon SQS 대기열 또는 AWS Lambda 함수로 전송

Amazon SNS는 기본적으로 사용되는 리전과 옵트인 리전 모두에 대해 교차 리전 전송을 지원합니다. 옵트인 리전을 포함하여 Amazon SNS가 지원하는 AWS 리전의 현재 목록은 Amazon Web Services 일반 참조에서 Amazon Simple Notification Service 엔드포인트 및 할당량을 참조하세요.

Amazon SNS는 Amazon SQS 대기열 및 AWS Lambda 함수에 대한 알림의 교차 리전 전송을 지원합니다. 리전 중 하나가 옵트인 리전인 경우 구독 리소스의 정책에서 다른 Amazon SNS 서비스 보안 주체를 지정해야 합니다.

Amazon SNS 구독 명령은 Amazon SNS가 호스팅되는 리전의 대상 계정에서 실행해야 합니다. 예를 들어 Amazon SNS가 us-east-1 리전의 계정'A'에 있고 Lambda 함수가 us-east-2 리전의 계정 'B'에 있는 경우 구독 CLI 명령은 us-east-1 리전의 계정 'A'에서 실행되어야 합니다.

옵트인 리전

Amazon SNS는 다음과 같은 옵트인 리전을 지원합니다.

지역명	리전
아프리카(케이프타운) 리전	af-south-1
Asia Pacific (Hong Kong) Region	ap-east-1
아시아 태평양(하이데라바드) 리전	ap-south-2
Asia Pacific (Jakarta) Region	ap-southeast-3
Asia Pacific (Melbourne) Region	ap-southeast-4
Asia Pacific (Osaka) Region	ap-northeast-3
Europe (Milan) Region	eu-south-1
유럽(스페인) 리전	eu-south-2
유럽(취리히) 리전	eu-central-2
Israel (Tel Aviv) Region	il-central-1
Middle East (Bahrain) Region	me-south-1
Middle East (UAE) Region	me-central-1

옵트인 지역을 활성화하는 방법에 대한 자세한 내용은 의 AWSAmazon Web Services 일반 참조지역 관리를 참조하십시오.

옵트인 리전에서 기본적으로 사용되는 리전으로 메시지를 전송하는 데 Amazon SNS를 사용하는 경우, 대기열에 생성된 리소스 정책을 변경해야 합니다. 보안 주체 `sns.amazonaws.com`을 `sns.<opt-in-region>.amazonaws.com`으로 바꿉니다. 예:

- 미국 동부(버지니아 북부)의 Amazon SQS 대기열을 아시아 태평양(홍콩)의 Amazon SNS 주제에 구독하려면 대기열 정책의 보안 주체를 `sns.ap-east-1.amazonaws.com`으로 변경하세요. 옵트인 리전은 2019년 3월 20일 이후에 시작된 모든 리전으로 구성됩니다. 여기에는 아시아 태평양(홍콩), 아시아 태평양(자카르타), 중동(바레인), 유럽(밀라노), 아프리카(케이프타운)가 포함됩니다. 2019년 3월 20일 이전에 시작된 리전은 기본적으로 활성화되어 있습니다.

Amazon SQS에 대한 교차 리전 전송 지원

교차 리전 전송 유형	지원됨/지원되지 않음	
기본 지원 리전에서 옵트인 리전으로	대기열의 서비스 보안 주체에서 `sns.<opt-in-region>.amazonaws.com` 을 사용하여 지원됨	
옵트인 리전에서 기본 지원 리전으로	대기열의 서비스 보안 주체에서 `sns.<opt-in-region>.amazonaws.com` 을 사용하여 지원됨	
옵트인 리전에서 옵트인 리전으로	지원되지 않음	

다음은 옵트인 지역 (af-south-1) 의 Amazon SNS 주제를 특정 지역의 Amazon SQS 대기열 (us-east-1) 에 전송할 수 있도록 허용하는 액세스 정책 설명의 예입니다. enabled-by-default 경로 `Statement/Principal/Service` 아래에 지역화된 필수 서비스 보안 주체 구성이 포함되어 있습니다.

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "allow_sns_arn:aws:sns:af-south-1:111111111111:source_topic_name",
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.af-south-1.amazonaws.com"
      },
      "Action": "SQS:SendMessage",
      "Resource": "arn:aws:sqs:us-east-1:111111111111:destination_queue_name",
      "Condition": {
        "ArnLike": {
          "aws:SourceArn": "arn:aws:sns:af-south-1:111111111111:source_topic_name"
        }
      }
    },
    ...
```

```
  ]
}
```

- 미국 동부 (버지니아 북부) 의 AWS Lambda 함수를 아시아 태평양 (홍콩) 의 Amazon SNS 주제에 구독하려면 AWS Lambda 함수 정책의 보안 주체를 로 `sns.ap-east-1.amazonaws.com` 변경하십시오. 옵트인 리전은 2019년 3월 20일 이후에 시작된 모든 리전으로 구성됩니다. 여기에는 아시아 태평양(홍콩), 아시아 태평양(자카르타), 중동(바레인), 유럽(밀라노), 아프리카(케이프타운)가 포함됩니다. 2019년 3월 20일 이전에 시작된 리전은 기본적으로 활성화되어 있습니다.

  지역 간 전송 지원: AWS Lambda

교차 리전 전송 유형	지원됨/지원되지 않음	
기본 지원 리전에서 옵트인 리전으로	지원되지 않음	
옵트인 리전에서 기본 지원 리전으로	Lambda 함수의 서비스 보안 주체에서 `sns.<opt-in-region>.amazonaws.com` 을 사용하여 지원됨	
옵트인 리전에서 옵트인 리전으로	지원되지 않음	

Amazon SNS 메시지 전송 상태

Amazon SNS는 다음 Amazon SNS 엔드포인트가 있는 주제에 전송된 알림 메시지의 전송 상태를 로깅하기 위한 지원을 제공합니다.

- HTTP
- Amazon Data Firehose
- AWS Lambda
- 플랫폼 애플리케이션 엔드포인트
- Amazon Simple Queue Service

메시지 전송 상태 속성을 구성하면 주제 구독자에게 전송되는 메시지에 대한 CloudWatch 로그 항목이 로그로 전송됩니다. 메시지 전송 상태를 로깅하면 다음과 같이 더욱 확장된 운영 이해를 제공할 수 있습니다.

- 메시지가 Amazon SNS 엔드포인트에 전송되었는지 확인
- Amazon SNS 엔드포인트에서 Amazon SNS로 전송된 응답 식별
- 메시지 유지 시간(게시 타임스탬프 시간부터 Amazon SNS 엔드포인트에 넘겨주기 직전까지의 시간) 결정

메시지 전송 상태를 위한 주제 속성을 구성하려면, AWS 소프트웨어 개발 키트 (SDK) AWS Management Console, 쿼리 API 또는 AWS CloudFormation를 사용할 수 있습니다.

주제

- AWS Management Console을 사용하여 전송 상태 로깅 구성
- AWS SDK를 사용하여 전송 상태 로깅 구성
- AWS 주제 속성을 구성하기 위한 SDK 예제
- AWS CloudFormation을 사용하여 전송 상태 로깅 구성

AWS Management Console을 사용하여 전송 상태 로깅 구성

1. Amazon SNS 콘솔에 로그인합니다.
2. 탐색 창에서 주제(Topics)를 선택합니다.
3. 주제 페이지에서 주제를 선택한 다음 편집을 선택합니다.
4. 편집 ***MyTopic***페이지에서 전송 상태 로깅 섹션을 확장합니다.
5. 전송 상태를 로그할 프로토콜을 선택합니다(예: AWS Lambda).
6. 성공 샘플 비율 (CloudWatch 로그를 수신하려는 성공 메시지의 백분율) 을 입력합니다.
7. IAM 역할 섹션에서 다음 중 하나를 수행합니다.

   - 계정에서 기존 서비스 역할을 선택하려면 Use existing service role(기존 서비스 역할 사용)을 선택한 후 성공한 전송과 실패한 전송의 IAM 역할을 지정합니다.

   - 계정에 새 서비스 역할을 생성하려면 Create new service role(새 서비스 역할 생성)을 선택하고 Create new roles(새 역할 생성)를 선택하여 IAM 콘솔에서 성공한 전송과 실패한 전송의 IAM 역할을 정의합니다.

     Amazon SNS에서 사용자 대신 CloudWatch 로그를 사용할 수 있는 쓰기 액세스 권한을 부여하려면 Allow (허용) 를 선택합니다.
8. 변경 사항 저장를 선택합니다.

이제 메시지 전송 상태가 포함된 CloudWatch 로그를 보고 분석할 수 있습니다. 사용에 CloudWatch 대한 자세한 내용은 CloudWatch설명서를 참조하십시오.

AWS SDK를 사용하여 전송 상태 로깅 구성

AWS SDK는 Amazon SNS의 메시지 전송 상태 속성을 사용하기 위한 여러 언어의 API를 제공합니다.

주제 속성

메시지 전송 상태를 위해 다음 주제 속성 이름 값을 사용할 수 있습니다.

HTTP

- `HTTPSuccessFeedbackRoleArn` - HTTP 엔드포인트에 가입된 Amazon SNS 주제에 대한 메시지 전송 성공 상태를 나타냅니다.
- `HTTPSuccessFeedbackSampleRate` - HTTP 엔드포인트에 가입된 Amazon SNS 주제에 대해 샘플링에 성공한 메시지의 비율을 나타냅니다.
- `HTTPFailureFeedbackRoleArn` - HTTP 엔드포인트에 가입된 Amazon SNS 주제에 대한 메시지 전송 실패 상태를 나타냅니다.

Amazon Data Firehose

- `FirehoseSuccessFeedbackRoleArn` - Amazon Kinesis Data Firehose 엔드포인트에 가입된 Amazon SNS 주제에 대한 메시지 전송 성공 상태를 나타냅니다.
- `FirehoseSuccessFeedbackSampleRate` - Amazon Kinesis Data Firehose 엔드포인트에 가입된 Amazon SNS 주제에 대해 샘플링에 성공한 메시지의 비율을 나타냅니다.
- `FirehoseFailureFeedbackRoleArn` - Amazon Kinesis Data Firehose 엔드포인트에 가입된 Amazon SNS 주제에 대한 메시지 전송 실패 상태를 나타냅니다.

AWS Lambda

- `LambdaSuccessFeedbackRoleArn` - Lambda 엔드포인트에 가입된 Amazon SNS 주제에 대한 메시지 전송 성공 상태를 나타냅니다.
- `LambdaSuccessFeedbackSampleRate` - Lambda 엔드포인트에 가입된 Amazon SNS 주제에 대해 샘플링에 성공한 메시지의 비율을 나타냅니다.

- `LambdaFailureFeedbackRoleArn` - Lambda 엔드포인트에 가입된 Amazon SNS 주제에 대한 메시지 전송 실패 상태를 나타냅니다.

플랫폼 애플리케이션 엔드포인트

- `ApplicationSuccessFeedbackRoleArn`— AWS 애플리케이션 엔드포인트에 가입된 Amazon SNS 주제의 성공적인 메시지 전송 상태를 나타냅니다.
- `ApplicationSuccessFeedbackSampleRate`— AWS 애플리케이션 엔드포인트에 가입된 Amazon SNS 주제에 대해 샘플링할 수 있는 성공 메시지의 비율을 나타냅니다.
- `ApplicationFailureFeedbackRoleArn`— AWS 애플리케이션 엔드포인트에 가입되어 있는 Amazon SNS 주제의 메시지 전송 실패 상태를 나타냅니다.

> **Note**
>
> Amazon SNS 애플리케이션 엔드포인트에 전송된 알림 메시지의 메시지 전송 상태를 위해 주제 속성을 구성할 수 있을 뿐만 아니라, 푸시 알림 서비스에 전송되는 푸시 알림 메시지의 전송 상태를 위해 애플리케이션 속성을 구성할 수도 있습니다. 자세한 정보는 메시지 전송 상태를 위한 Amazon SNS 애플리케이션 속성 사용을 참조하세요.

Amazon SQS

- `SQSSuccessFeedbackRoleArn` - Amazon SQS 엔드포인트에 가입된 Amazon SNS 주제에 대한 메시지 전송 성공 상태를 나타냅니다.
- `SQSSuccessFeedbackSampleRate` - Amazon SQS 엔드포인트에 가입된 Amazon SNS 주제에 대해 샘플링에 성공한 메시지의 비율을 나타냅니다.
- `SQSFailureFeedbackRoleArn` - Amazon SQS 엔드포인트에 가입된 Amazon SNS 주제에 대한 메시지 전송 실패 상태를 나타냅니다.

> **Note**
>
> `<ENDPOINT>SuccessFeedbackRoleArn`및 `<ENDPOINT>FailureFeedbackRoleArn` 속성은 Amazon SNS에서 사용자를 대신하여 CloudWatch 로그를 사용할 수 있는 쓰기 액세스 권한을 부여하는 데 사용됩니다. `<ENDPOINT>SuccessFeedbackSampleRate` 속성은 성공적으로 전송된 메시지의 샘플 비율(0-100)을 지정할 때 사용됩니다.

<ENDPOINT>FailureFeedbackRoleArn속성을 구성하고 나면 실패한 모든 메시지 전송으로 CloudWatch 로그가 생성됩니다.

AWS 주제 속성을 구성하기 위한 SDK 예제

다음 코드 예제는 SetTopicAttributes의 사용 방법을 보여 줍니다.

CLI

AWS CLI

주제에 대한 속성을 설정하려면

다음 set-topic-attributes 예제에서는 지정된 주제에 DisplayName 속성을 설정합니다.

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name DisplayName \
    --attribute-value MyTopicDisplayName
```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 SetTopicAttributes참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetTopicAttributes {

    public static void main(String[] args) {
        final String usage = """

                Usage:    <attribute> <topicArn> <value>

                Where:
                   attribute - The attribute action to use. Valid parameters are:
 Policy | DisplayName | DeliveryPolicy .
                   topicArn - The ARN of the topic.\s
                   value - The value for the attribute.
                """;

        if (args.length < 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String attribute = args[0];
        String topicArn = args[1];
        String value = args[2];

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        setTopAttr(snsClient, attribute, topicArn, value);
        snsClient.close();
    }

    public static void setTopAttr(SnsClient snsClient, String attribute, String
 topicArn, String value) {
        try {
```

```
            SetTopicAttributesRequest request =
 SetTopicAttributesRequest.builder()
                    .attributeName(attribute)
                    .attributeValue(value)
                    .topicArn(topicArn)
                    .build();

            SetTopicAttributesResponse result =
 snsClient.setTopicAttributes(request);
            System.out.println(
                    "\n\nStatus was " + result.sdkHttpResponse().statusCode() +
 "\n\nTopic " + request.topicArn()
                            + " updated " + request.attributeName() + " to " +
 request.attributeValue());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API SetTopicAttributes참조를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

별도의 모듈에서 클라이언트를 생성하고 내보냅니다.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
```

```
export const snsClient = new SNSClient({});
```

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
import { SetTopicAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

export const setTopicAttributes = async (
  topicArn = "TOPIC_ARN",
  attributeName = "DisplayName",
  attributeValue = "Test Topic",
) => {
  const response = await snsClient.send(
    new SetTopicAttributesCommand({
      AttributeName: attributeName,
      AttributeValue: attributeValue,
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'd1b08d0e-e9a4-54c3-b8b1-d03238d2b935',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API SetTopicAttributes참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun setTopAttr(attribute: String?, topicArnVal: String?, value: String?)
 {

    val request = SetTopicAttributesRequest {
        attributeName = attribute
        attributeValue = value
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.setTopicAttributes(request)
        println("Topic ${request.topicArn} was updated.")
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 SetTopicAttributes.

PHP

SDK for PHP

> **Note**
>
> 자세한 내용은 여기에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'vendor/autoload.php';
```

```
use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Configure the message delivery status attributes for an Amazon SNS Topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
$attribute = 'Policy | DisplayName | DeliveryPolicy';
$value = 'First Topic';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->setTopicAttributes([
        'AttributeName' => $attribute,
        'AttributeValue' => $value,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- API 세부 정보는 AWS SDK for PHP API SetTopicAttributes참조를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# Service class to enable an SNS resource with a specified policy
class SnsResourceEnabler
  # Initializes the SnsResourceEnabler with an SNS resource client
  #
  # @param sns_resource [Aws::SNS::Resource] The SNS resource client
  def initialize(sns_resource)
    @sns_resource = sns_resource
    @logger = Logger.new($stdout)
  end

  # Sets a policy on a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource to include in the policy
  # @param policy_name [String] The name of the policy attribute to set
  def enable_resource(topic_arn, resource_arn, policy_name)
    policy = generate_policy(topic_arn, resource_arn)
    topic = @sns_resource.topic(topic_arn)

    topic.set_attributes({
                           attribute_name: policy_name,
                           attribute_value: policy
                         })
    @logger.info("Policy #{policy_name} set successfully for topic
 #{topic_arn}.")
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Failed to set policy: #{e.message}")
  end

  private

  # Generates a policy string with dynamic resource ARNs
```

```
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource
  # @return [String] The policy as a JSON string
  def generate_policy(topic_arn, resource_arn)
    {
      Version: "2008-10-17",
      Id: "__default_policy_ID",
      Statement: [{
                   Sid: "__default_statement_ID",
                   Effect: "Allow",
                   Principal: { "AWS": "*" },
                   Action: ["SNS:Publish"],
                   Resource: topic_arn,
                   Condition: {
                     ArnEquals: {
                       "AWS:SourceArn": resource_arn
                     }
                   }
                 }]
    }.to_json
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = "MY_TOPIC_ARN"     # Should be replaced with a real topic ARN
  resource_arn = "MY_RESOURCE_ARN" # Should be replaced with a real resource ARN
  policy_name = "POLICY_NAME"    # Typically, this is "Policy"

  sns_resource = Aws::SNS::Resource.new
  enabler = SnsResourceEnabler.new(sns_resource)

  enabler.enable_resource(topic_arn, resource_arn, policy_name)
end
```

- 자세한 정보는 AWS SDK for Ruby 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for Ruby API SetTopicAttributes참조를 참조하십시오.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        lo_sns->settopicattributes(
            iv_topicarn = iv_topic_arn
            iv_attributename  = iv_attribute_name
            iv_attributevalue = iv_attribute_value
        ).
        MESSAGE 'Set/updated SNS topic attributes.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 SAP용AWS SDK ABAP API 참조를 참조하십시오 SetTopicAttributes.

AWS CloudFormation을 사용하여 전송 상태 로깅 구성

사용을 구성하려면 JSON 또는 YAML 템플릿을 `DeliveryStatusLogging` 사용하여 AWS CloudFormation스택을 생성하십시오. AWS CloudFormation 자세한 내용은 AWS CloudFormation 사용 설명서의 `AWS::SNS::Topic` 리소스 `DeliveryStatusLogging` 속성을 참조하십시오. 다음은 Amazon SQS AWS CloudFormation 프로토콜의 모든 `DeliveryStatusLogging` 속성을 사용하여 새 주제를 생성하거나 기존 주제를 업데이트하기 위한 JSON 및 YAML 템플릿의 예입니다.

JSON

```
"Resources": {
    "MySNSTopic" : {
        "Type" : "AWS::SNS::Topic",
        "Properties" : {
            "TopicName" : "TestTopic",
            "DisplayName" : "TEST",
```

```
            "SignatureVersion" : "2",
            "DeliveryStatusLogging" : [{
                "Protocol": "sqs",
                "SuccessFeedbackSampleRate": "45",
                "SuccessFeedbackRoleArn": "arn:aws:iam::123456789012:role/
SNSSuccessFeedback_test1",
                "FailureFeedbackRoleArn": "arn:aws:iam::123456789012:role/
SNSFailureFeedback_test2"
            }]
        }
    }
}
```

YAML

```
Resources:
  MySNSTopic:
    Type: AWS::SNS::Topic
    Properties:
      TopicName:TestTopic
      DisplayName:TEST
      SignatureVersion:2
      DeliveryStatusLogging:
       - Protocol: sqs
         SuccessFeedbackSampleRate: 45
         SuccessFeedbackRoleArn: arn:aws:iam::123456789012:role/
SNSSuccessFeedback_test1
         FailureFeedbackRoleArn: arn:aws:iam::123456789012:role/
SNSFailureFeedback_test2
```

Amazon SNS 메시지 전송 재시도

Amazon SNS에서는 각 전송 프로토콜에 대한 전송 정책을 정의합니다. 서버 측 오류가 발생할 경우(즉, 구독 엔드포인트를 호스팅하는 시스템을 사용할 수 없게 될 경우) Amazon SNS가 메시지 전송을 재시도하는 방식이 전송 정책에 따라 결정됩니다. 전송 정책이 소진되면 Amazon SNS는 전송 재시도를 중지하고 배달 못한 편지 대기열이 구독에 연결되어 있지 않는 한 메시지를 삭제합니다. 자세한 정보는 Amazon SNS 배달 못한 편지 대기열(DLQ)에서 확인하세요.

주제

전송 프로토콜 및 정책

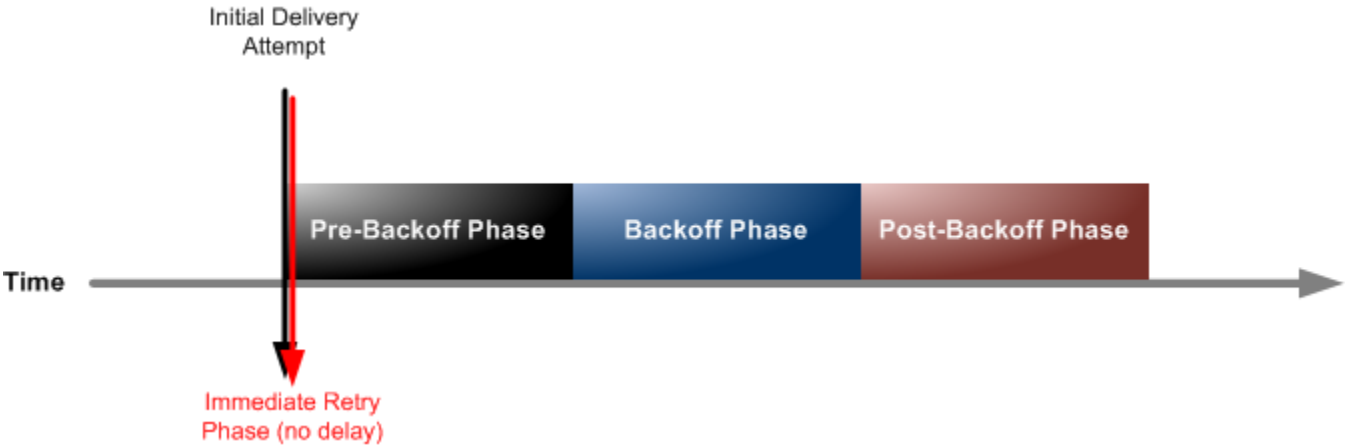
> **Note**
>
> - HTTP/S를 제외하고 Amazon SNS에서 정의하는 전송 정책을 변경할 수 없습니다. HTTP/S만 사용자 지정 정책을 지원합니다. HTTP/S 전송 정책 생성에서 확인하세요.
> - Amazon SNS는 전송 재시도에 지터링을 적용합니다. 자세한 정보는 AWS 아키텍처 블로그의 지수 백오프 및 지터 게시물을 참조하세요.
> - HTTP/S 엔드포인트의 총 정책 재시도 시간은 3,600초를 초과할 수 없습니다. 이 값은 고정 한도이며 늘릴 수 없습니다.

[엔드포인트 유형]	전송 프로토콜	즉각 재시도(지연 없음) 단계	프리 백오프 단계	백오프 단계	포스트 백오프 단계	총 시도 횟수
AWS 관리형 엔드포인트	아마존 데이터 파이어호스[1]	3회, 지연 없음	2회, 1초 간격	10회, 1초~20초 범위에서 지수 백오프 사용	100,000회, 20초 간격	100,015회, 23일 동안
	AWS Lambda					
	Amazon SQS					
고객 관리형 엔드포인트	SMTP	0회, 지연 없음	2회, 10초 간격	10회, 10초~600초(10분) 범위에서 지수 백오프 사용	38회, 600초(10분) 간격	50회 시도, 6시간 동안
	SMS					
	모바일 푸시					

[1] Firehose 프로토콜의 제한 오류의 경우 Amazon SNS는 고객 관리형 엔드포인트와 동일한 전송 정책을 사용합니다.

전송 정책 단계

다음 다이어그램은 전송 정책의 단계를 보여줍니다.



각 전송 정책은 4단계로 구성됩니다.

1. 즉각 재시도 단계(지연 없음) – 이 단계는 첫 전송 시도 직후에 발생합니다. 이 단계에서는 재시도 간에 지연이 없습니다.
2. 프리 백오프 단계 – 이 단계는 즉시 재시도 단계 후에 이어집니다. Amazon SNS는 이 단계를 사용하여 백오프 기능을 적용하기 전에 일련의 재시도를 시도합니다. 이 단계에서는 재시도 횟수와 재시도 간의 지연 시간을 지정합니다.
3. 백오프 단계 – 이 단계에서는 재시도 백오프 함수를 사용하여 재시도 간 지연을 제어합니다. 최소 지연 시간과 최대 지연 시간을 설정한 다음 재시도 백오프 함수를 설정하여 최소 지연 시간부터 최대 지연 시간까지 어느 정도 간격으로 지연을 늘릴 것인지를 정의합니다. 백오프 함수는 Arithmetic, Exponential, Geometric 또는 Linear가 될 수 있습니다.
4. 포스트 백오프 단계 – 백오프 단계 뒤에 오는 이 단계에서는 재시도 횟수와 재시도 간의 지연 시간을 지정합니다. 이것이 마지막 단계입니다.

HTTP/S 전송 정책 생성

전송 정책과 해당 4단계를 사용하여 Amazon SNS가 HTTP/S 엔드포인트로 메시지 전송을 재시도하는 방법을 정의할 수 있습니다. Amazon SNS를 사용하면 예를 들어 HTTP 서버의 용량을 기반으로 정책을 사용자 지정하려는 경우 HTTP 엔드포인트에 대한 기본 재시도 정책을 재정의할 수 있습니다.

구독 또는 주제 수준에서 HTTP/S 전송 정책을 JSON 객체로 설정할 수 있습니다. 주제 수준에서 정책을 정의하면 주제에 연결된 모든 HTTP/S 구독에 정책이 적용됩니다. 구독 수준에서 전송 정책을 설정하려면 `Subscribe` 또는 `SetSubscriptionAttributes` API 작업을 사용할 수 있습니다. 주제 수준에서 전송 정책을 설정하려면 `CreateTopic` 또는 `SetTopicAttributes` API 작업을 사용할 수 있습니다. 또는 템플릿의 AWS::SNS::Subscription리소스를 사용할 수도 있습니다. AWS CloudFormation

HTTP/S 서버의 용량에 따라 전송 정책을 사용자 지정해야 합니다. 정책은 주제 속성 또는 구독 속성으로 설정할 수 있습니다. 주제의 모든 HTTP/S 구독이 동일한 HTTP/S 서버를 대상으로 하는 경우 전송 정책을 주제 속성으로 설정하여 주제의 모든 HTTP/S 구독에 정책이 적용되도록 하는 것이 좋습니다. 이렇게 하지 않으면 정책이 대상으로 하는 HTTP/S 서버의 용량에 따라 주제의 각 HTTP/S 구독에 대한 전송 정책을 작성해야 합니다.

요청 정책에서 Content-Type 헤더에 대해서도 알림의 미디어 유형을 지정할 수 있습니다. 기본적으로 Amazon SNS 콘텐츠 유형이 `text/plain; charset=UTF-8`로 설정된 HTTP/S 엔드포인트로 모든 알림을 보냅니다. Amazon SNS를 사용하면 기본 요청 정책을 재정의할 수 있습니다. 지원 `headerContentType` 및 제한 사항은 아래 표를 참조하세요.

다음 JSON 객체는 실패한 HTTP/S 전송을 아래와 같이 재시도하도록 Amazon SNS에 지시하는 전송 정책을 나타냅니다.

1. 지연 없음 단계에서 즉시 3회
2. 프리 백오프 단계에서 2회(1초 간격)
3. 10회(1초~60초 범위에서 지수 백오프 사용)
4. 포스트 백오프 단계에서 35회(60초 간격)

이 샘플 전송 정책에서 Amazon SNS는 메시지를 삭제하기 전에 총 50번의 시도를 합니다. 전송 정책에 지정된 재시도가 소진된 후에도 메시지를 유지하려면 전송할 수 없는 메시지를 배달 못한 편지 대기열(DLQ)로 이동하도록 구독을 구성합니다. 자세한 정보는 Amazon SNS 배달 못한 편지 대기열(DLQ)에서 확인하세요.

> **Note**
>
> 또한 이 전송 정책은 `maxReceivesPerSecond` 속성을 이용하여 초당 10개 이하로 전송을 제한하도록 Amazon SNS에 지시합니다. 이 자체 조절 속도로 인해 전송된 메시지 (아웃바운드 트래픽) 보다 게시된 메시지 (인바운드 트래픽) 수가 더 많을 수 있습니다. 아웃바운드 트래픽보다 인바운드 트래픽이 더 많은 경우 구독에 큰 메시지 백로그가 누적되어 메시지 전송 대기

시간이 길어질 수 있습니다. 전송 정책에서는 maxReceivesPerSecond에 대해 워크로드에 부정적인 영향을 미치지 않는 값을 지정해야 합니다.

> **Note**
>
> 이 전송 정책은 application/json에 HTTP/S 알림을 위한 기본 콘텐츠 유형을 재정의합니다.

```
{
    "healthyRetryPolicy": {
        "minDelayTarget": 1,
        "maxDelayTarget": 60,
        "numRetries": 50,
        "numNoDelayRetries": 3,
        "numMinDelayRetries": 2,
        "numMaxDelayRetries": 35,
        "backoffFunction": "exponential"
    },
    "throttlePolicy": {
        "maxReceivesPerSecond": 10
    },
    "requestPolicy": {
        "headerContentType": "application/json"
    }
}
```

전송 정책은 재시도 정책과 제한 정책, 요청 정책으로 구성됩니다. 전송 정책에는 총 9개의 속성이 있습니다.

정책	설명	Constraint
minDelayTarget	재시도의 최소 지연 시간입니다. 단위: 초	1~최대 지연 시간 기본값: 20
maxDelayTarget	재시도의 최대 지연 시간입니다.	최소 지연 시간~3,600

정책	설명	Constraint
	단위: 초	기본값: 20
numRetries	즉각 재시도, 프리 백오프, 백오프, 포스트 백오프 단계의 재시도를 모두 포함한 총 재시도 횟수입니다.	0~100 기본값: 3
numNoDelayRetries	재시도 간 지연 없이 즉각 수행되는 재시도 횟수입니다.	0 이상 기본값: 0
numMinDelayRetries	재시도 간 지정된 최소 지연 시간으로 프리 백오프 단계에서 수행되는 재시도 횟수입니다.	0 이상 기본값: 0
numMaxDelayRetries	재시도 간 지정된 최대 지연 시간으로 포스트 백오프 단계에서 수행되는 재시도 횟수입니다.	0 이상 기본값: 0
backoffFunction	재시도 간 백오프에 사용되는 모델입니다.	다음 4가지 옵션 중 하나: • Arithmetic • Exponential • Geometric • Linear 기본값: Linear
maxReceivesPerSecond	구독별 초당 최대 전송 수입니다.	1 이상 기본값: 제한 없음

정책	설명	Constraint
`headerContentType`	HTTP/S 엔드포인트로 전송되는 알림의 콘텐츠 유형입니다.	요청 정책이 정의되지 않은 경우 콘텐츠 유형의 기본값은 `text/plain; charset=UTF-8` 입니다. 구독에 대해 원시 메시지 전송을 비활성화하거나(기본값) 전송 정책이 주제 수준에서 정의된 경우 지원되는 헤더 콘텐츠 유형은 `application/json` 및 `text/plain` 입니다. 구독에서 원시 메시지 전송을 활성화하면 다음과 같은 콘텐츠 유형이 지원됩니다. • text/css • text/csv • text/html • text/plain • text/xml • application/atom+xml • application/json • application/octet-stream • application/soap+xml • 애플리케이션/ x-www-form-urlencoded • application/xhtml+xml • application/xml

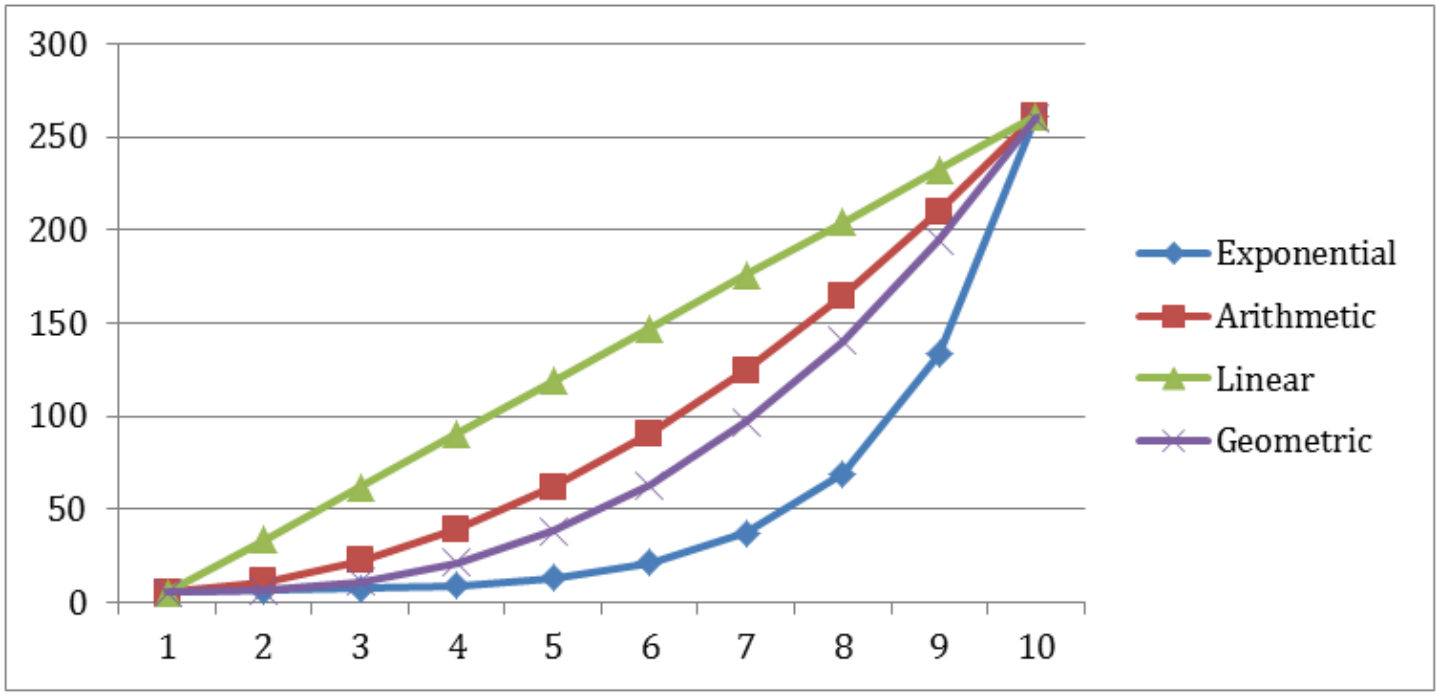
Amazon SNS에서는 다음 수식을 사용하여 백오프 단계의 재시도 횟수를 계산합니다.

```
numRetries - numNoDelayRetries - numMinDelayRetries - numMaxDelayRetries
```

3가지 파라미터를 사용하여 백오프 단계의 재시도 빈도를 제어할 수 있습니다.

- `minDelayTarget` – 백오프 단계에서 처음 수행하는 재시도에 대한 지연을 정의합니다.
- `maxDelayTarget` – 백오프 단계에서 마지막으로 수행하는 재시도에 대한 지연을 정의합니다.
- `backoffFunction` – Amazon SNS가 백오프 단계의 첫 재시도부터 마지막 재시도까지의 모든 재시도에 대한 지연을 계산하는 데 사용할 알고리즘을 정의합니다. 4가지 재시도 백오프 함수 중 하나를 사용할 수 있습니다.

다음 다이어그램은 백오프 단계에서 각 재시도 백오프 함수에 따라 재시도 관련 지연이 어떻게 달라지는지를 보여줍니다. 전송 정책의 총 재시도 횟수는 10회, 최소 지연 시간은 5초, 최대 지연 시간은 260초로 설정되어 있습니다. 세로 축은 10번의 재시도 각각에 대한 지연(초)을 나타냅니다. 가로 축은 첫 번째 시도에서 열 번째 시도까지의 재시도 횟수를 나타냅니다.



Amazon SNS 배달 못한 편지 대기열(DLQ)

배달 못한 편지 대기열은 Amazon SNS 구독이 구독자에게 성공적으로 배달할 수 없는 메시지를 저장하는 Amazon SQS 대기열입니다. 클라이언트 오류 또는 서버 오류로 인해 배달할 수 없는 메시지는 추가 분석 또는 재처리를 위해 배달 못한 편지 대기열에 보관됩니다. 자세한 정보는 구독에 대한 Amazon SNS 배달 못한 편지 대기열 구성 및 Amazon SNS 메시지 전송 재시도에서 확인하세요.

> **Note**
>
> - Amazon SNS 구독 및 Amazon SQS 대기열은 동일한 AWS 계정 및 리전에 있어야 합니다.
> - FIFO 주제의 경우 Amazon SQS 대기열을 Amazon SNS 구독에 대한 DLQ(Dead Letter Queue)로 사용할 수 있습니다. FIFO 주제 구독은 FIFO 대기열을 사용하고, 표준 주제 구독은 표준 대기열을 사용합니다.
> - 암호화된 Amazon SQS 대기열을 DLQ(Dead Letter Queue)로 사용하려면 Amazon SNS 서비스 보안 주체에 AWS KMS API 작업에 대한 액세스 권한을 부여하는 키 정책과 함께 사용자 지정 KMS를 사용해야 합니다. 자세한 정보는 이 가이드의 저장 시 암호화 및 Amazon Simple Queue Service 개발자 안내서의 서버 측 암호화(SSE) 및 AWS KMS를 사용하여 Amazon SQS 데이터 보호를 참조하세요.

주제

메시지 전송이 실패하는 이유

일반적으로 Amazon SNS가 클라이언트 측 또는 서버 측 오류로 인해 구독된 엔드포인트에 액세스할 수 없으면 메시지 전송이 실패합니다. Amazon SNS 에서 클라이언트 측 오류를 수신하거나 해당 재시도 정책에 지정된 재시도 횟수 이상의 메시지에 대해 서버 측 오류가 계속 수신되면 Amazon SNS는 구독에 배달 못한 편지 대기열이 연결되어 있지 않는 한 메시지를 삭제합니다. 전송이 실패해도 구독의 상태는 바뀌지 않습니다. 자세한 내용은 Amazon SNS 메시지 전송 재시도 섹션을 참조하세요.

클라이언트 측 오류

Amazon SNS에 부실 구독 메타데이터가 있으면 클라이언트 측 오류가 발생할 수 있습니다. 이러한 오류는 일반적으로 소유자가 엔드포인트(예: Amazon SNS 주제를 구독하는 Lambda 함수)를 삭제하거나 소유자가 구독 엔드포인트에 연결된 정책을 변경하여 Amazon SNS에서 엔드포인트에 메시지를 전

송하지 못하게 될 경우에 발생합니다. Amazon SNS는 클라이언트 측 오류로 인해 실패한 메시지 전송을 다시 시도하지 않습니다.

서버 측 오류

구독 엔드포인트를 담당하는 시스템이 사용 불가능하게 되거나 Amazon SNS의 유효한 요청을 처리할 수 없음을 나타내는 예외를 반환하는 경우 서버 측 오류가 발생할 수 있습니다. 서버 측 오류가 발생하면 Amazon SNS는 선형 또는 지수 백오프 함수를 사용하여 실패한 전송을 재시도합니다. 서버 측 오류가 Amazon SQS 또는 AWS Lambda 기반의 AWS 관리형 엔드포인트로 인해 발생하는 경우 Amazon SNS는 23일 동안 최대 100,015회까지 전송을 재시도합니다.

고객 관리형 엔드포인트(예: HTTP, SMTP, SMS 또는 모바일 푸시)도 서버 측 오류가 발생하는 원인이 될 수 있습니다. Amazon SNS는 이러한 유형의 엔드포인트로 전송 역시 재시도합니다. HTTP 엔드포인트에서 고객이 정의하는 재시도 정책이 지원되지만 Amazon SNS는 SMTP, SMS 및 모바일 푸시 엔드포인트에 대해 내부 전송 재시도 정책을 6시간 동안 50회로 설정합니다.

배달 못한 편지 대기열의 작동 방식

메시지 전송은 구독 수준에서 이루어지기 때문에 배달 못한 편지 대기열은 주제가 아닌 Amazon SNS 구독에 연결됩니다. 이를 통해 각 메시지의 원래 대상 엔드포인트를 보다 쉽게 식별할 수 있습니다.

Amazon SNS 구독에 연결된 배달 못한 편지 대기열은 일반 Amazon SQS 대기열입니다. 메시지 보존 기간에 대한 자세한 정보는 Amazon Simple Queue Service 개발자 안내서의 메시지 관련 할당량을 참조하세요. Amazon SQS `SetQueueAttributes` API 작업을 사용하여 메시지 보존 기간을 변경할 수 있습니다. 애플리케이션의 복원력을 높이려면 배달 못한 편지 대기열의 최대 보존 기간을 14일로 설정하는 것이 좋습니다.

메시지가 배달 못한 편지 대기열로 이동하는 방식

메시지는 리드라이브 정책을 사용하여 배달 못한 편지 대기열로 이동합니다. 리드라이브 정책은 배달 못한 편지 대기열의 ARN을 참조하는 JSON 객체입니다. `deadLetterTargetArn` 속성을 통해 ARN이 지정됩니다. ARN은 Amazon SNS 구독과 동일한 AWS 계정 및 리전의 Amazon SQS 대기열을 가리켜야 합니다. 자세한 내용은 구독에 대한 Amazon SNS 배달 못한 편지 대기열 구성 섹션을 참조하세요.

다음 JSON 객체는 SNS 구독에 연결된 샘플 리드라이브 정책입니다.

```
{
  "deadLetterTargetArn": "arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue"
```

```
}
```

배달 못한 편지 대기열로부터 메시지를 이동하는 방법

다음 두 가지 방법을 사용하여 메시지를 배달 못한 편지 대기열로부터 이동할 수 있습니다.

- Amazon SQS 소비자 로직 작성 방지 – 배달 못한 편지 대기열을 Lambda 함수에 대한 이벤트 소스로 설정하여 배달 못한 편지 대기열을 비웁니다.
- Amazon SQS 소비자 로직 작성 – Amazon SQS API, AWS SDK 또는 AWS CLI를 사용하여 사용자 지정 소비자 로직을 작성함으로써 배달 못한 편지 대기열의 메시지를 폴링, 처리 및 삭제합니다.

배달 못한 편지 대기열을 모니터링 및 로깅하는 방법

Amazon CloudWatch 지표를 사용하여 Amazon SNS 구독에 연결된 배달 못한 편지 대기열을 모니터링할 수 있습니다. 모든 Amazon SQS 대기열은 1분 간격으로 CloudWatch 지표를 보냅니다. 자세한 정보는 Amazon Simple Queue Service 개발자 안내서의 Amazon SQS에 사용 가능한 CloudWatch 지표를 참조하세요. 배달 못한 편지 대기열이 있는 모든 Amazon SNS 구독에서도 CloudWatch 지표를 내보냅니다. 자세한 내용은 CloudWatch를 사용하여 Amazon SNS 주제 모니터링 섹션을 참조하세요.

배달 못한 편지 대기열의 활동에 대한 알림을 받으려면 CloudWatch 지표와 경보를 사용할 수 있습니다. 예를 들어 배달 못한 편지 대기열이 항상 비어 있기를 원하는 경우 `NumberOfMessagesSent` 지표에 대한 CloudWatch 경보를 생성할 수 있습니다. 경보 임계값을 0으로 설정하고 경보가 울릴 때 알림을 받을 Amazon SNS 주제를 지정할 수 있습니다. 이 Amazon SNS 주제는 모든 엔드포인트 유형(예: 이메일 주소, 전화번호 또는 모바일 호출기 앱)에 경보 알림을 전송할 수 있습니다.

CloudWatch Logs를 사용하면 Amazon SNS 전송이 실패하고 메시지가 배달 못한 편지 대기열로 이동한 원인이 되는 예외적인 상황을 조사할 수 있습니다. Amazon SNS는 CloudWatch에 성공한 전송과 실패한 전송을 모두 로그할 수 있습니다. 자세한 내용은 Amazon SNS 메시지 전송 상태 섹션을 참조하세요.

구독에 대한 Amazon SNS 배달 못한 편지 대기열 구성

배달 못한 편지 대기열은 Amazon SNS 구독이 구독자에게 성공적으로 배달할 수 없는 메시지를 저장하는 Amazon SQS 대기열입니다. 클라이언트 오류 또는 서버 오류로 인해 배달할 수 없는 메시지는 추가 분석 또는 재처리를 위해 배달 못한 편지 대기열에 보관됩니다. 자세한 정보는 Amazon SNS 배달 못한 편지 대기열(DLQ) 및 Amazon SNS 메시지 전송 재시도에서 확인하세요.

이 페이지에서는, AWS SDK AWS Management Console AWS CLI, 및 를 사용하여 Amazon SNS 구독을 위한 데드레터 대기열을 구성하는 AWS CloudFormation 방법을 보여줍니다.

> **Note**
>
> FIFO 주제의 경우 Amazon SQS 대기열을 Amazon SNS 구독에 대한 DLQ(Dead Letter Queue)로 사용할 수 있습니다. FIFO 주제 구독은 FIFO 대기열을 사용하고, 표준 주제 구독은 표준 대기열을 사용합니다.

필수 조건

배달 못한 편지 대기열을 구성하려면 먼저 다음 사전 조건을 완료합니다.

1. `MyTopic`이라는 Amazon SNS 주제 생성
2. Amazon SNS 구독의 엔드포인트로 사용할 `MyEndpoint`이라는 Amazon SQS 대기열을 생성합니다.
3. (건너뛰기 AWS CloudFormation) 해당 주제 대기열을 구독하십시오.
4. Amazon SNS 구독의 배달 못한 편지 대기열로 사용할 `MyDeadLetterQueue`이라는 다른 Amazon SQS 대기열을 생성합니다.
5. Amazon SNS 보안 주체에 Amazon SQS API 작업에 대한 액세스 권한을 부여하려면 `MyDeadLetterQueue`에 대해 다음 대기열 정책을 설정합니다.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "Service": "sns.amazonaws.com"
    },
    "Action": "SQS:SendMessage",
    "Resource": "arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue",
    "Condition": {
      "ArnEquals": {
        "aws:SourceArn": "arn:aws:sns:us-east-2:123456789012:MyTopic"
      }
    }
  }]
}
```

주제

- 를 사용하여 Amazon SNS 구독을 위한 데드레터 대기열을 구성하려면 AWS Management Console

- SDK를 사용하여 Amazon SNS 구독을 위한 데드레터 대기열을 구성하려면 AWS
- 를 사용하여 Amazon SNS 구독을 위한 데드레터 대기열을 구성하려면 AWS CLI
- 를 사용하여 Amazon SNS 구독을 위한 데드레터 대기열을 구성하려면 AWS CloudFormation

를 사용하여 Amazon SNS 구독을 위한 데드레터 대기열을 구성하려면 AWS Management Console

이 자습서를 시작하기 전에 사전 조건을 완료해야 합니다.

1. Amazon SQS 콘솔에 로그인합니다.
2. Amazon SQS 대기열을 생성하거나 기존 대기열을 사용하여 대기열의 세부 정보 탭에서 대기열의 ARN을 확인합니다.

   ```
   arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue
   ```

3. Amazon SNS 콘솔에 로그인합니다.
4. 탐색 창에서 구독을 선택합니다.
5. [구독] 페이지에서 기존 구독을 선택한 다음 [편집]을 선택합니다.
6. 편집 ***1234a567-bc89-012d-3e45-6fg7h890123i*** 페이지 에서 리드라이브 정책(배달 못한 편지 대기열 섹션을 확장하고 다음을 수행합니다.

   a. 활성을 선택합니다.

   b. Amazon SQS 대기열의 ARN을 지정합니다.
7. 변경 사항 저장를 선택합니다.

   구독이 배달 못한 편지 대기열을 사용하도록 구성됩니다.

SDK를 사용하여 Amazon SNS 구독을 위한 데드레터 대기열을 구성하려면 AWS

이 예제를 실행하기 전에 사전 조건을 완료해야 합니다.

AWS SDK를 사용하려면 사용자 인증 정보로 구성해야 합니다. 자세한 정보는 AWS SDK 및 도구 참조 가이드의 공유 구성 및 자격 증명 파일을 참조하세요.

다음 코드 예제는 사용 `SetSubscriptionAttributesRedrivePolicy` 방법을 보여줍니다.

Java

SDK for Java 1.x

> **Note**
>
> 더 많은 내용이 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Specify the ARN of the Amazon SNS subscription.
String subscriptionArn =
    "arn:aws:sns:us-east-2:123456789012:MyEndpoint:1234a567-
bc89-012d-3e45-6fg7h890123i";

// Specify the ARN of the Amazon SQS queue to use as a dead-letter queue.
String redrivePolicy =
    "{\"deadLetterTargetArn\":\"arn:aws:sqs:us-
east-2:123456789012:MyDeadLetterQueue\"}";

// Set the specified Amazon SQS queue as a dead-letter queue
// of the specified Amazon SNS subscription by setting the RedrivePolicy
 attribute.
SetSubscriptionAttributesRequest request = new SetSubscriptionAttributesRequest()
    .withSubscriptionArn(subscriptionArn)
    .withAttributeName("RedrivePolicy")
    .withAttributeValue(redrivePolicy);
sns.setSubscriptionAttributes(request);
```

를 사용하여 Amazon SNS 구독을 위한 데드레터 대기열을 구성하려면 AWS CLI

이 자습서를 시작하기 전에 사전 조건을 완료해야 합니다.

1. AWS CLI를 설치하고 구성합니다. 자세한 정보는 AWS Command Line Interface 사용 설명서를 참조하세요.
2. 다음 명령을 사용합니다.

```
aws sns set-subscription-attributes \
```

```
--subscription-arn arn:aws:sns:us-east-2:123456789012:MyEndpoint:1234a567-
bc89-012d-3e45-6fg7h890123i
--attribute-name RedrivePolicy
--attribute-value "{\"deadLetterTargetArn\": \"arn:aws:sqs:us-
east-2:123456789012:MyDeadLetterQueue\"}"
```

를 사용하여 Amazon SNS 구독을 위한 데드레터 대기열을 구성하려면 AWS CloudFormation

이 자습서를 시작하기 전에 사전 조건을 완료해야 합니다.

1. 다음 JSON 코드를 `MyDeadLetterQueue.json`이라는 파일에 복사합니다.

```
{
  "Resources": {
    "mySubscription": {
      "Type" : "AWS::SNS::Subscription",
      "Properties" : {
        "Protocol": "sqs",
        "Endpoint": "arn:aws:sqs:us-east-2:123456789012:MyEndpoint",
        "TopicArn": "arn:aws:sns:us-east-2:123456789012:MyTopic",
        "RedrivePolicy": {
          "deadLetterTargetArn":
            "arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue"
        }
      }
    }
  }
}
```

2. AWS CloudFormation 콘솔에 로그인합니다.
3. [Select Template] 페이지에서 [Upload a template to Amazon S3]를 선택하고, `MyDeadLetterQueue.json` 파일을 선택하고 나서 [Next]를 선택합니다.
4. 세부 정보 지정 페이지에서 스택 이름에 `MyDeadLetterQueue`를 입력한 후 다음을 선택합니다.
5. 옵션 페이지에서 다음을 선택합니다.
6. 검토(Review) 페이지에서 생성(Create)을 선택합니다.

AWS CloudFormation `MyDeadLetterQueue`스택 생성을 시작하고 CREATE_IN_PROGRESS 상태를 표시합니다. 프로세스가 완료되면 CREATE_COMPLETE 상태가 표시됩니다 AWS CloudFormation .

Amazon SNS 메시지 아카이브, 재생 및 분석

Amazon SNS 표준 주제는 Amazon Data Firehose를 통한 메시지 보관을 지원합니다. Firehose 전송 스트림으로 알림을 팬아웃하여 Amazon Simple Storage Service (Amazon S3), Amazon Redshift 등 Firehose가 지원하는 스토리지 및 분석 대상으로 알림을 보낼 수 있습니다.

Amazon SNS FIFO 주제는 주제 소유자가 주제에 게시된 메시지를 최대 365일 동안 저장(또는 아카이브)할 수 있는 코드 없는 인플레이스 메시지 아카이브를 지원합니다. 활성 `ArchivePolicy`이 있는 주제의 경우, 구독자는 구독한 엔드포인트로 아카이브된 메시지를 다시 가져오기(또는 재생하기) 위한 `ReplayPolicy`을 생성할 수 있습니다. 이 기능에 대해 자세히 알아보려면 FIFO 주제의 메시지 아카이브 및 재생 섹션을 참조하세요.

특성	표준 주제	FIFO 주제
메시지 아카이브	팬아웃에서 Firehose로의 전송 스트림	FIFO 주제 소유자의 메시지 아카이브
메시지 재생	표준 주제에 대한 재생은 기본 제공 기능이 아닙니다. 많은 고객이 메시지 아카이브를 기반으로 자체 메시지 아카이브를 구축합니다.	FIFO 주제 구독자를 위한 메시지 재생

애플리케이션 간(A2A) 메시징에 Amazon SNS 사용

이 섹션에서는 구독자와의 애플리케이션 간 메시징에 Amazon SNS를 사용하는 방법에 대한 정보를 제공합니다.

주제

팬아웃에서 Firehose로의 전송 스트림

Amazon Data Firehose 전송 스트림을 Amazon SNS 주제로 구독하면 추가 스토리지 및 분석 엔드포인트에 알림을 보낼 수 있습니다. Amazon SNS 주제에 게시된 메시지는 구독된 Firehose 전송 스트림으로 전송되고 Firehose에 구성된 대로 대상으로 전송됩니다. 구독 소유자는 Amazon SNS 주제에 대한 Firehose 전송 스트림을 최대 5개까지 구독할 수 있습니다. 각 Firehose 전송 스트림에는 요청 할당량과 초당 처리량에 대한 기본 할당량이 있습니다. 이 제한으로 인해 전달된 메시지(아웃바운드 트래픽)보다 게시된 메시지(인바운드 트래픽)가 더 많을 수 있습니다. 아웃바운드 트래픽보다 인바운드 트래픽이 더 많은 경우 구독에 큰 메시지 백로그가 누적되어 메시지 전송 대기 시간이 길어집니다. 워크로드에 부정적인 영향을 미치지 않도록 게시 속도에 따라 할당량 증가를 요청할 수 있습니다.

Firehose 전송 스트림을 통해 Amazon SNS 알림을 아마존 심플 스토리지 서비스 (Amazon S3), 아마존 Redshift, 아마존 서비스 (서비스) 및 데이터독, OpenSearch 뉴렐릭, MongoDB, 스플렁크와 같은 타사 서비스 제공업체로 전송할 수 있습니다. OpenSearch

예를 들어 이 기능을 사용하여 규정 준수, 아카이브 또는 기타 목적을 위해 Amazon S3 버킷의 주제로 전송된 메시지를 영구 저장할 수 있습니다. 이렇게 하려면 S3 버킷 대상으로 Firehose 전송 스트림을 생성하고 해당 전송 스트림을 Amazon SNS 주제에 구독하십시오. 또 다른 예로, Amazon SNS 주제로 전송된 메시지를 분석하려면 OpenSearch 서비스 인덱스 대상을 사용하여 전송 스트림을 생성하십시오. 그런 다음 Firehose 전송 스트림을 구독하여 Amazon SNS 주제를 구독할 수 있습니다.

또한 Amazon SNS는 Firehose 엔드포인트로 전송되는 알림에 대한 메시지 전송 상태 로깅을 지원합니다. 자세한 내용은 Amazon SNS 메시지 전송 상태을(를) 참조하세요.

주제

Amazon SNS 주제에 대한 Firehose 전송 스트림을 구독하기 위한 사전 요구 사항

Amazon Data Firehose 전송 스트림을 SNS 주제로 구독하려면 다음이 AWS 계정 있어야 합니다.

- 표준 SNS 주제. 자세한 설명은 Amazon SNS 주제 생성 섹션을 참조하세요.
- Firehose 전송 스트림. 자세한 내용은 Amazon Data Firehose 개발자 안내서의 Amazon Data Firehose 전송 스트림 생성 및 애플리케이션에 Firehose 리소스에 대한 액세스 권한 부여를 참조하십시오.
- Amazon SNS 서비스 주체를 신뢰하고 전송 스트림에 대한 쓰기 권한이 있는 AWS Identity and Access Management(IAM) 역할입니다. 구독을 생성할 때 이 역할의 Amazon 리소스 이름(ARN)을 `SubscriptionRoleARN`으로 입력합니다. Amazon SNS가 이 역할을 맡아 Amazon SNS가 Firehose 전송 스트림에 레코드를 넣을 수 있습니다.

  다음 정책 예는 권장 권한을 보여 줍니다.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "firehose:DescribeDeliveryStream",
        "firehose:ListDeliveryStreams",
        "firehose:ListTagsForDeliveryStream",
        "firehose:PutRecord",
        "firehose:PutRecordBatch"
      ],
      "Resource": [
        "arn:aws:firehose:us-east-1:111111111111:deliverystream/firehose-sns-
delivery-stream"
      ],
      "Effect": "Allow"
```

```
    }
  ]
}
```

Firehose 사용에 대한 전체 권한을 제공하려면 AWS 관리형 정책을 사용할 수도 있습니다. `AmazonKinesisFirehoseFullAccess` 또는 Firehose를 사용하기 위한 더 엄격한 권한을 제공하기 위해 자체 정책을 만들 수 있습니다. 정책은 최소한 특정 전송 스트림에서 `PutRecord` 작업을 실행할 수 있는 권한을 제공해야 합니다.

어떤 경우든 Amazon SNS 서비스 주체를 포함하도록 신뢰 관계를 편집해야 합니다. 예:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

역할 생성에 대한 자세한 정보는 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요.

이러한 요구 사항을 완료한 후 전송 스트림에서 SNS 주제를 구독할 수 있습니다.

Firehose 전송 스트림을 Amazon SNS 주제에 구독하기

Amazon SNS 알림을 Amazon Data Firehose 전송 스트림으로 전송하려면 먼저 모든 사전 요구 사항을 충족했는지 확인하십시오. 지원되는 엔드포인트 목록은 의 Amazon Data Firehose 엔드포인트 및 할당량을 참조하십시오. Amazon Web Services 일반 참조

Firehose 전송 스트림을 구독하여 주제를 구독하려면

1. Amazon SNS 콘솔에 로그인합니다.
2. 탐색 창에서 구독을 선택합니다.
3. 구독 페이지에서 구독 생성을 선택합니다.

4. 구독 생성 페이지의 세부 정보 섹션에서 다음을 수행합니다.

   a. 주제 ARN에서 표준 주제의 Amazon 리소스 이름(ARN)을 선택합니다.

   b. 프로토콜에서 Firehose를 선택합니다.

   c. 엔드포인트의 경우 Amazon SNS로부터 알림을 수신할 수 있는 Firehose 전송 스트림의 ARN을 선택합니다.

   d. 구독 역할 ARN의 경우 Firehose 전송 스트림에 쓰기 위해 만든 AWS Identity and Access Management (IAM) 역할의 ARN을 지정합니다. 자세한 설명은 Amazon SNS 주제에 대한 Firehose 전송 스트림을 구독하기 위한 사전 요구 사항 섹션을 참조하세요.

   e. (선택 사항) 게시된 메시지에서 Amazon SNS 메타데이터를 제거하려면 원시 메시지 전송 사용을 선택합니다. 자세한 설명은 Amazon SNS 원시 메시지 전송 섹션을 참조하세요.

5. (선택 사항) 필터 정책을 구성하려면 구독 필터 정책 섹션을 확장합니다. 자세한 설명은 Amazon SNS 구독 필터 정책 섹션을 참조하세요.

6. (선택 사항) 구독에 대한 배달 못한 편지 대기열을 구성하려면 리드라이브 정책(배달 못한 편지 대기열) 섹션을 확장합니다. 자세한 설명은 Amazon SNS 배달 못한 편지 대기열(DLQ) 섹션을 참조하세요.

7. 구독 생성을 선택합니다.

콘솔에서 구독을 만들고 구독의 세부 정보 페이지를 엽니다.

전송 스트림 대상 작업

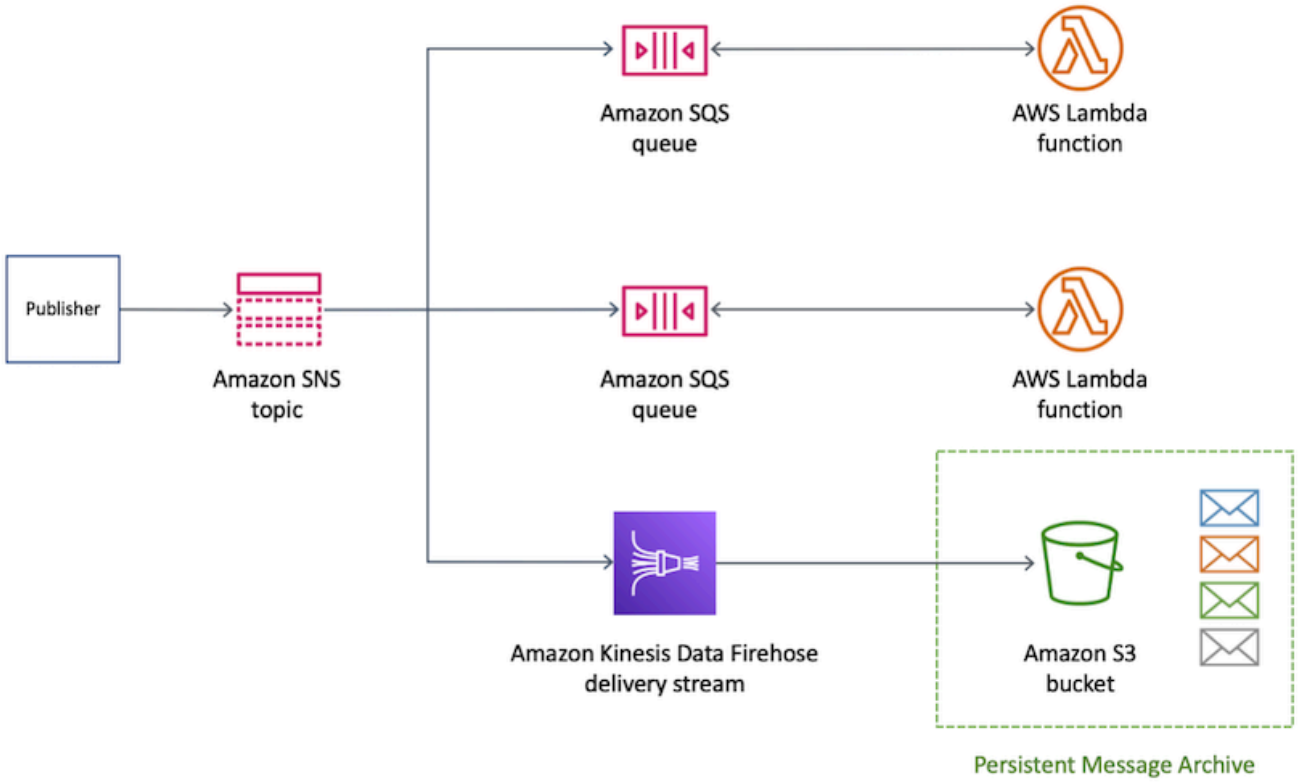
Amazon Data Firehose 전송 스트림을 통해 추가 엔드포인트로 메시지를 보낼 수 있습니다. 이 섹션에서는 지원되는 대상으로 작업하는 방법을 설명합니다.

주제

- Amazon S3 대상
- OpenSearch 서비스 목적지
- Amazon Redshift 대상
- HTTP 대상

Amazon S3 대상

이 섹션에서는 Amazon Simple Storage Service (Amazon S3) 에 데이터를 게시하는 Amazon Data Firehose 전송 스트림에 대한 정보를 제공합니다.



주제

- Amazon S3 대상에 대해 아카이브된 메시지 형식
- Amazon S3 대상에 대한 메시지 분석

Amazon S3 대상에 대해 아카이브된 메시지 형식

다음 예에서는 가독성을 위해 들여쓰기를 사용하여 Amazon Simple Storage Service(Amazon S3) 버킷으로 전송된 Amazon SNS 알림을 보여줍니다.

> **Note**
>
> 이 예에서는 게시된 메시지에 대해 원시 메시지 전송 기능이 사용 중지됩니다. 원시 메시지 전송이 사용 중지되면 Amazon SNS는 이러한 속성을 포함하여 메시지에 JSON 메타데이터를 추가합니다.
>
> - `Type`
> - `MessageId`
> - `TopicArn`

- Subject
- Timestamp
- UnsubscribeURL
- MessageAttributes

원시 전송에 대한 자세한 정보는 Amazon SNS 원시 메시지 전송에서 확인하세요.

```
{
    "Type": "Notification",
    "MessageId": "719a6bbf-f51b-5320-920f-3385b5e9aa56",
    "TopicArn": "arn:aws:sns:us-east-1:333333333333:my-kinesis-test-topic",
    "Subject": "My 1st subject",
    "Message": "My 1st body",
    "Timestamp": "2020-11-26T23:48:02.032Z",
    "UnsubscribeURL": "https://sns.us-east-1.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:333333333333:my-kinesis-test-
topic:0b410f3c-ee5e-49d8-b59b-3b4aa6d8fcf5",
    "MessageAttributes": {
        "myKey1": {
            "Type": "String",
            "Value": "myValue1"
        },
        "myKey2": {
            "Type": "String",
            "Value": "myValue2"
        }
    }
 }
```

다음 예는 Amazon Data Firehose 전송 스트림을 통해 동일한 Amazon S3 버킷으로 전송된 세 개의 SNS 메시지를 보여줍니다. 버퍼링이 고려되고 줄 바꿈으로 메시지를 구분합니다.

```
{"Type":"Notification","MessageId":"d7d2513e-6126-5d77-
bbe2-09042bd0a03a","TopicArn":"arn:aws:sns:us-east-1:333333333333:my-
kinesis-test-topic","Subject":"My 1st subject","Message":"My 1st
 body","Timestamp":"2020-11-27T00:30:46.100Z","UnsubscribeURL":"https://
sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
east-1:313276652360:my-kinesis-test-topic:0b410f3c-ee5e-49d8-
```

```
b59b-3b4aa6d8fcf5","MessageAttributes":{"myKey1":
{"Type":"String","Value":"myValue1"},"myKey2":{"Type":"String","Value":"myValue2"}}}
{"Type":"Notification","MessageId":"0c0696ab-7733-5bfb-b6db-
ce913c294d56","TopicArn":"arn:aws:sns:us-east-1:333333333333:my-
kinesis-test-topic","Subject":"My 2nd subject","Message":"My 2nd
 body","Timestamp":"2020-11-27T00:31:22.151Z","UnsubscribeURL":"https://
sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
east-1:313276652360:my-kinesis-test-topic:0b410f3c-ee5e-49d8-
b59b-3b4aa6d8fcf5","MessageAttributes":{"myKey1":{"Type":"String","Value":"myValue1"}}}
{"Type":"Notification","MessageId":"816cd54d-8cfa-58ad-91c9-8d77c7d173aa","TopicArn":"arn:aws:s
east-1:333333333333:my-kinesis-test-topic","Subject":"My 3rd subject","Message":"My
 3rd body","Timestamp":"2020-11-27T00:31:39.755Z","UnsubscribeURL":"https://
sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
east-1:313276652360:my-kinesis-test-topic:0b410f3c-ee5e-49d8-b59b-3b4aa6d8fcf5"}
```


Amazon S3 대상에 대한 메시지 분석

이 페이지에서는 Amazon Data Firehose 전송 스트림을 통해 Amazon Simple Storage Service (Amazon S3) 대상으로 전송된 Amazon SNS 메시지를 분석하는 방법을 설명합니다.

Firehose 전송 스트림을 통해 Amazon S3 대상으로 전송된 SNS 메시지를 분석하려면

1. Amazon S3 리소스를 구성합니다. 지침은 Amazon Simple Storage Service 사용 설명서의 버킷 생성 및 Amazon Simple Storage Service 사용 설명서의 Amazon S3 버킷 작업을 참조하세요.
2. 전송 스트림을 구성합니다. 지침은 Amazon Data Firehose 개발자 안내서에서 Amazon S3를 대상으로 선택하기 참조하십시오.
3. Amazon Athena를 사용하여 표준 SQL로 Amazon S3 객체를 쿼리합니다. 자세한 정보는 Amazon Athena 사용 설명서의 시작하기를 참조하세요.

쿼리 예

이번 쿼리 예에서는 다음과 같이 가정합니다.

- 메시지는 `default` 스키마의 `notifications` 테이블에 저장됩니다.
- `notifications` 테이블에는 유형이 `string`인 `timestamp` 열이 포함되어 있습니다.

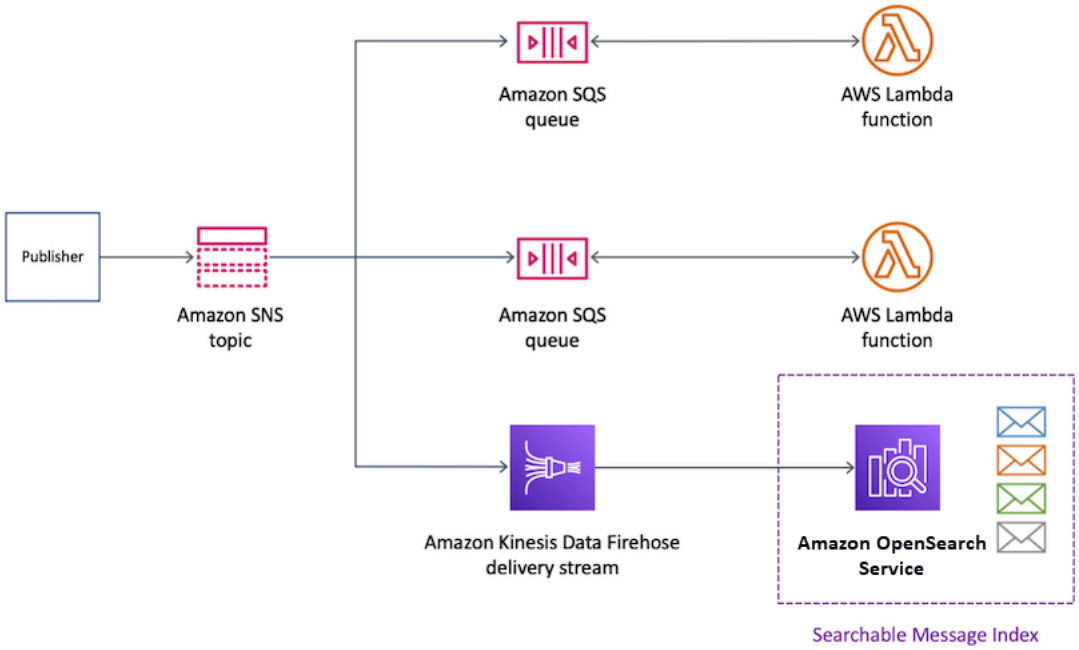
다음 쿼리는 지정된 날짜 범위에 수신된 모든 SNS 메시지를 반환합니다.

```
SELECT *
FROM default.notifications
```

```
WHERE from_iso8601_timestamp(timestamp) BETWEEN TIMESTAMP '2020-12-01 00:00:00' AND
 TIMESTAMP '2020-12-02 00:00:00';
```

OpenSearch 서비스 목적지

이 섹션에서는 Amazon OpenSearch 서비스 (OpenSearch 서비스) 에 데이터를 게시하는 Amazon Data Firehose 전송 스트림에 대한 정보를 제공합니다.



주제

- OpenSearch Service 인덱스의 아카이브된 메시지 형식
- OpenSearch 서비스 대상에 대한 메시지 분석

OpenSearch Service 인덱스의 아카이브된 메시지 형식

다음 예에서는 `my-index`라는 Amazon OpenSearch Service(OpenSearch Service) 인덱스로 전송된 Amazon SNS 알림을 보여줍니다. 이 인덱스에는 `Timestamp` 필드에 시간 필터 필드가 있습니다. SNS 알림은 페이로드의 `_source` 속성에 있습니다.

> **Note**
>
> 이 예에서는 게시된 메시지에 대해 원시 메시지 전송 기능이 사용 중지됩니다. 원시 메시지 전송이 사용 중지되면 Amazon SNS는 이러한 속성을 포함하여 메시지에 JSON 메타데이터를 추가합니다.

- Type
- MessageId
- TopicArn
- Subject
- Timestamp
- UnsubscribeURL
- MessageAttributes

원시 전송에 대한 자세한 정보는 Amazon SNS 원시 메시지 전송에서 확인하세요.


```
{
  "_index": "my-index",
  "_type": "_doc",
  "_id": "4961310096311132320325040540219328379477388655098593280 2.0",
  "_version": 1,
  "_score": null,
  "_source": {
    "Type": "Notification",
    "MessageId": "bf32e294-46e3-5dd5-a6b3-bad65162e136",
    "TopicArn": "arn:aws:sns:us-east-1:111111111111:my-topic",
    "Subject": "Sample subject",
    "Message": "Sample message",
    "Timestamp": "2020-12-02T22:29:21.189Z",
    "UnsubscribeURL": "https://sns.us-east-1.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:111111111111:my-
topic:b5aa9bc1-9c3d-452b-b402-aca2cefc63c9",
    "MessageAttributes": {
      "my_attribute": {
        "Type": "String",
        "Value": "my_value"
      }
    }
  },
  "fields": {
    "Timestamp": [
      "2020-12-02T22:29:21.189Z"
    ]
```

```
  },
  "sort": [
    1606948161189
  ]
}
```

OpenSearch 서비스 대상에 대한 메시지 분석

이 페이지에서는 Amazon Data Firehose 전송 스트림을 통해 Amazon OpenSearch 서비스(OpenSearch 서비스) 대상으로 전송되는 Amazon SNS 메시지를 분석하는 방법을 설명합니다.

Firehose 전송 스트림을 통해 서비스 대상으로 전송된 SNS 메시지를 분석하려면 OpenSearch

1. OpenSearch 서비스 리소스를 구성하세요. 지침은 Amazon 서비스 개발자 안내서의 Amazon OpenSearch OpenSearch Service 시작하기를 참조하십시오.
2. 전송 스트림을 구성합니다. 지침은 Amazon Data Firehose 개발자 안내서의 대상 OpenSearch 서비스 선택을 참조하십시오.
3. OpenSearch 서비스 쿼리와 Kibana를 사용하여 쿼리를 실행하십시오. 자세한 내용은 Amazon OpenSearch 서비스 개발자 안내서의 3단계: OpenSearch 서비스 도메인 및 Kibana에서 문서 검색을 참조하십시오.

쿼리 예

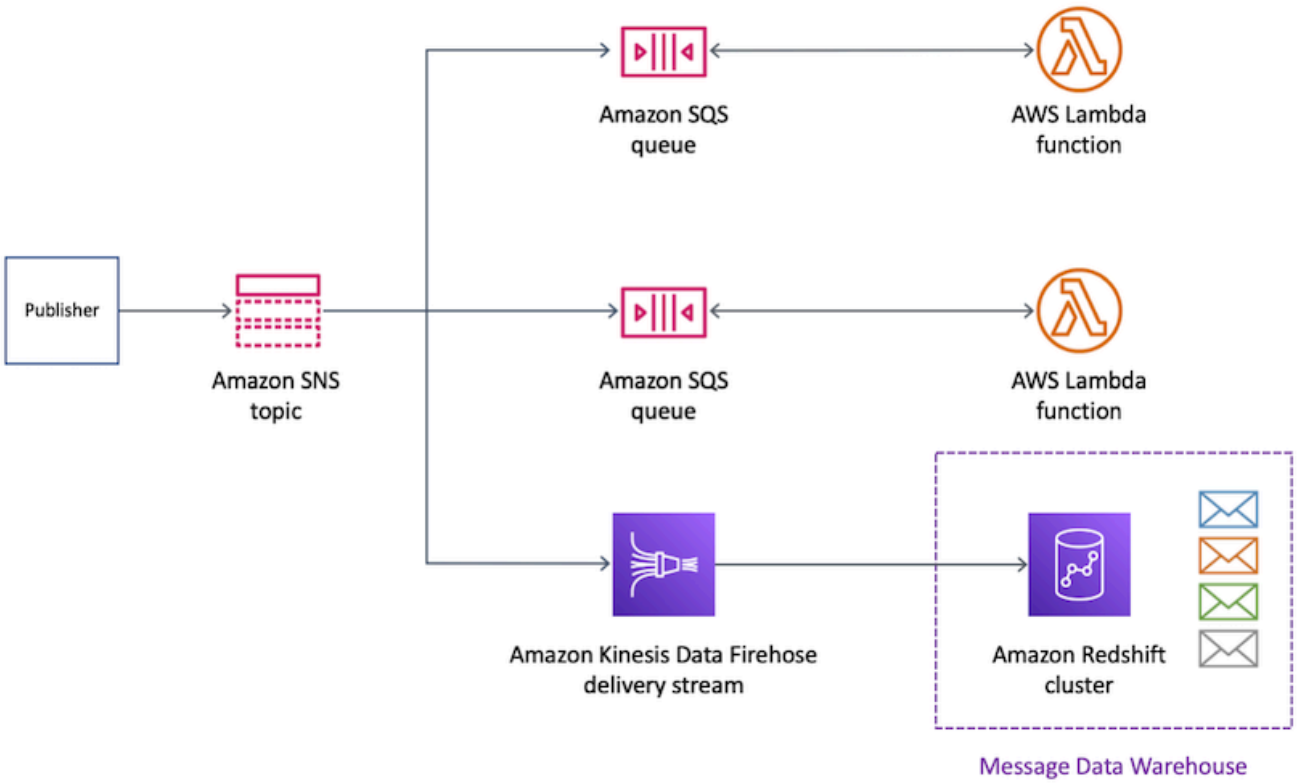
다음 예에서는 지정된 날짜 범위에 수신된 모든 SNS 메시지에 대한 `my-index` 인덱스를 쿼리합니다.

```
POST https://search-my-domain.us-east-1.es.amazonaws.com/my-index/_search
{
  "query": {
    "bool": {
      "filter": [
        {
          "range": {
            "Timestamp": {
              "gte": "2020-12-08T00:00:00.000Z",
              "lte": "2020-12-09T00:00:00.000Z",
              "format": "strict_date_optional_time"
            }
          }
        }
      ]
```

```
    }
  }
}
```

Amazon Redshift 대상

이 섹션에서는 Amazon Redshift에 데이터를 게시하는 Amazon Data Firehose 전송 스트림으로 Amazon SNS 알림을 팬아웃하는 방법을 설명합니다. 이 구성을 사용하면 Amazon Redshift 데이터베이스에 연결하고 SQL 쿼리 도구를 사용하여 데이터베이스에서 특정 기준을 충족하는 Amazon SNS 메시지를 쿼리할 수 있습니다.



주제

- Amazon Redshift 대상에 대한 아카이브 테이블 구조
- Amazon Redshift 대상에 대한 메시지 분석

Amazon Redshift 대상에 대한 아카이브 테이블 구조

Amazon Redshift 엔드포인트의 경우 게시된 Amazon SNS 메시지는 테이블의 행으로 아카이브됩니다. 다음은 예제입니다.

> **Note**
>
> 이 예에서는 게시된 메시지에 대해 원시 메시지 전송 기능이 사용 중지됩니다. 원시 메시지 전송이 사용 중지되면 Amazon SNS는 이러한 속성을 포함하여 메시지에 JSON 메타데이터를 추가합니다.
>
> - `Type`
> - `MessageId`
> - `TopicArn`
> - `Subject`
> - `Message`
> - `Timestamp`
> - `UnsubscribeURL`
> - `MessageAttributes`
>
> 원시 전송에 대한 자세한 정보는 Amazon SNS 원시 메시지 전송에서 확인하세요.
> Amazon SNS는 이 목록에 표시된 대문자를 사용하여 메시지에 속성을 추가하지만 Amazon Redshift 테이블의 열 이름은 모두 소문자로 표시됩니다. Amazon Redshift 엔드포인트에 대한 JSON 메타데이터를 변환하려면 `COPY` 명령을 사용할 수 있습니다. 자세한 정보는 Amazon Redshift 데이터베이스 개발자 안내서의 JSON 예에서 복사 및 'auto ignorecase' 옵션을 사용하여 JSON 데이터에서 로드를 참조하세요.

type	messageid	topicarn	subject	message	timestamp	unsubscribeurl	messageattributes
알림	ea544832-a0d8-581d-9275-108243c46103	arn:aws:sns:us-east-1:111111111111:my-topic	샘플 제목	샘플 메시지	2020-12-02T00:33:32.272Z	https://sns.us-east-1.amazonaws.com/?Action=Unsubscribe&Subscri	{\"my_attribute\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

type	messageid	topicarn	subject	message	timestamp	unsubscribeurl	messageattributes
						ptionArn=arn:aws:sns:us-east-1:111111111111:my-topic:326deeeb-cbf4-45da-b92b-ca77a247813b	

type	messageid	topicarn	subject	message	timestamp	unsubscribeurl	messageattributes
알림	ab124832-a0d8-581d-9275-108243c46114	arn:aws:sns:us-east-1:111111111111:my-topic	샘플 제목 2	샘플 메시지 2	2020-12-03T00:18:11.129Z	https://sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:111111111111:my-topic:326deeeb-cbf4-45da-b92b-ca77a247813b	{\"my_attribute2\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

type	messageid	topicarn	subject	message	timestamp	unsubscribeurl	messageattributes
알림	ce644832-a0d8-581d-9275-108243c46125	arn:aws:sns:us-east-1:111111111111:my-topic	샘플 제목 3	샘플 메시지 3	2020-12-09T00:08:44.405Z	https://sns.us-east-1.amazonaws.com/?Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-east-1:111111111111:my-topic:326deeeb-cbf4-45da-b92b-ca77a247813b	{\"my_attribute3\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

Amazon Redshift 엔드포인트로 알림을 팬아웃하는 방법에 대한 자세한 정보는 Amazon Redshift 대상에서 확인하세요.

Amazon Redshift 대상에 대한 메시지 분석

이 페이지에서는 Amazon Data Firehose 전송 스트림을 통해 Amazon Redshift 대상으로 전송되는 Amazon SNS 메시지를 분석하는 방법을 설명합니다.

Firehose 전송 스트림을 통해 Amazon Redshift 대상으로 전송된 SNS 메시지를 분석하려면

1. Amazon Redshift 리소스를 구성합니다. 지침은 Amazon Redshift 시작 안내서의 Amazon Redshift 시작하기를 참조하세요.
2. 전송 스트림을 구성합니다. 자세한 지침은 Amazon Data Firehose 개발자 안내서의 목적지로 Amazon Redshift를 선택하십시오.
3. 쿼리를 실행합니다. 자세한 정보는 Amazon Redshift 관리 안내서의 쿼리 편집기를 사용하여 데이터베이스 쿼리를 참조하세요.

쿼리 예

이번 쿼리 예에서는 다음과 같이 가정합니다.

- 메시지는 기본 `public` 스키마의 `notifications` 테이블에 저장됩니다.
- SNS 메시지의 `Timestamp` 속성은 열 데이터 유형이 `timestamptz`인 테이블의 `timestamp` 열에 저장됩니다.

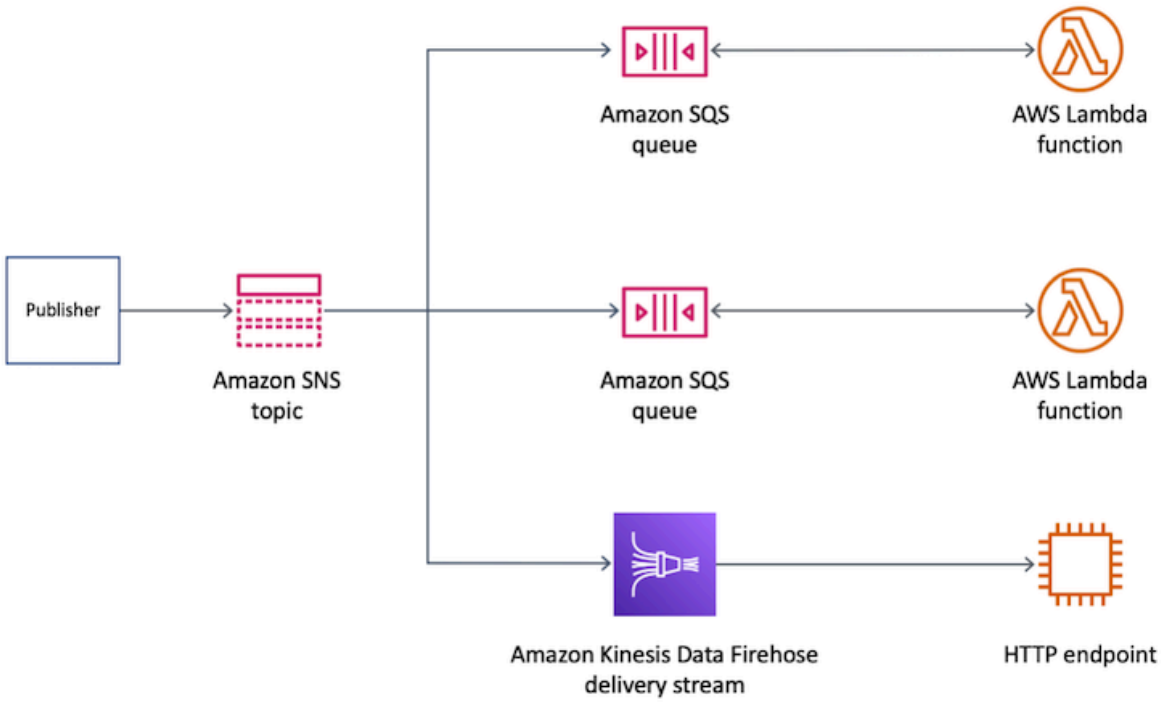
> **Note**
>
> Amazon Redshift 엔드포인트에 대한 JSON 메타데이터를 변환하려면 `COPY` 명령을 사용할 수 있습니다. 자세한 정보는 Amazon Redshift 데이터베이스 개발자 안내서의 JSON 예에서 복사 및 'auto ignorecase' 옵션을 사용하여 JSON 데이터에서 로드를 참조하세요.

다음 쿼리는 지정된 날짜 범위에 수신된 모든 SNS 메시지를 반환합니다.

```
SELECT *
FROM public.notifications
WHERE timestamp > '2020-12-01T09:00:00.000Z' AND timestamp <
 '2020-12-02T09:00:00.000Z';
```

HTTP 대상

이 섹션에서는 HTTP 엔드포인트에 데이터를 게시하는 Amazon Data Firehose 전송 스트림에 대한 정보를 제공합니다.



주제

- HTTP 대상에 대해 전송된 메시지 형식

HTTP 대상에 대해 전송된 메시지 형식

다음은 Amazon Data Firehose 전송 스트림이 HTTP 엔드포인트로 전송할 수 있는 Amazon SNS의 HTTP POST 요청 본문의 예시입니다. SNS 알림은 `records` 속성에서 base64 페이로드로 인코딩됩니다.

> **Note**
>
> 이 예에서는 게시된 메시지에 대해 원시 메시지 전송 기능이 사용 중지됩니다. 원시 전송에 대한 자세한 정보는 Amazon SNS 원시 메시지 전송에서 확인하세요.

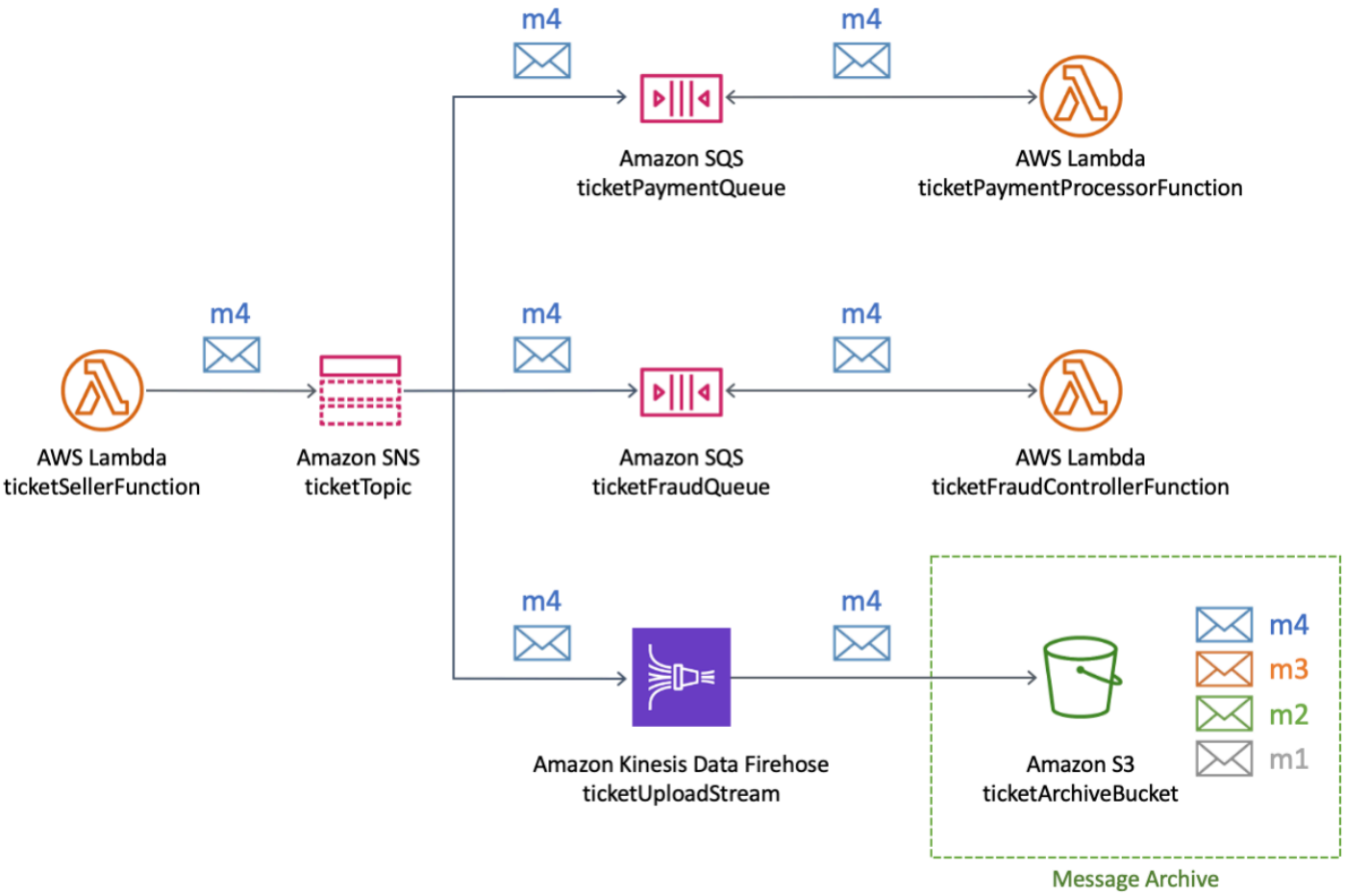
```
"body": {
    "requestId": "ebc9e8b2-fce3-4aef-a8f1-71698bf8175f",
    "timestamp": 1606255960435,
    "records": [
```

```
      {
        "data":
"eyJUeXBlIjoiTm90aWZpY2F0aW9uIiwiTWVzc2FnZUlkIjoiMjFkMmUzOGQtMmNhYi01ZjYxLTliYTItYmJiYWFhYzg0M
      }
    ]
 }
```

메시지 아카이브 및 분석 사용 사례 예

이 섹션에서는 Amazon SNS 메시지를 아카이브 및 분석하기 위한 일반적인 사용 사례에 대한 자습서를 제공합니다.

이 사용 사례의 배경은 규제된 환경에서 작동하는 항공권 발권 플랫폼입니다. 이 플랫폼은 회사가 모든 티켓 판매 내역을 최소 5년 동안 아카이브하도록 요구하는 규정 준수 프레임워크의 적용을 받습니다. 데이터 보존에 대한 규정 준수 목표를 달성하기 위해 회사는 Amazon Data Firehose 전송 스트림을 기존 SNS 주제에 구독합니다. 전송 스트림의 대상은 Amazon Simple Storage Service(Amazon S3) 버킷입니다. 이 구성을 사용하면 SNS 주제에 게시된 모든 이벤트가 Amazon S3 버킷에 아카이브됩니다. 다음 다이어그램은 이 구성의 아키텍처를 보여줍니다.



분석을 실행하고 티켓 판매에 대한 인사이트를 얻기 위해 회사는 Amazon Athena를 사용하여 SQL 쿼리를 실행합니다. 예를 들어, 회사는 가장 인기 있는 목적지와 가장 자주 이용하는 항공편에 대해 알아보기 위해 쿼리를 작성할 수 있습니다.

이 사용 사례에 대한 AWS 리소스를 생성하기 위해 AWS Management Console 또는 AWS CloudFormation 템플릿을 사용할 수 있습니다.

주제

- 초기 리소스 생성
- Firehose 전송 스트림 생성
- Firehose 전송 스트림을 Amazon SNS 주제에 구독하기
- 구성 테스트 및 쿼리
- AWS CloudFormation 템플릿 사용

초기 리소스 생성

이 페이지에서는 메시지 아카이브 및 분석 예 사용 사례에 대해 다음과 같은 리소스를 만드는 방법을 설명합니다.

- Amazon Simple Storage Service(Amazon S3) 버킷
- Amazon Simple Queue Service(Amazon SQS) 대기열 2개
- Amazon SNS 주제
- Amazon SNS 주제에 대한 Amazon SQS 구독 2개

초기 리소스를 생성하려면

1. 다음과 같이 Amazon S3 버킷을 생성합니다.

   a. Amazon S3 콘솔을 엽니다.
   b. 버킷 만들기를 선택합니다.
   c. 버킷 이름에 전 세계적으로 고유한 이름을 입력합니다. 다른 필드는 기본값으로 유지합니다.
   d. 버킷 만들기를 선택합니다.

   Amazon S3 버킷에 대한 자세한 정보는 Amazon Simple Storage Service 사용 설명서의 버킷 생성 및 Amazon Simple Storage Service 사용 설명서의 Amazon S3 버킷 작업을 참조하세요.
2. 다음과 같이 Amazon SQS 대기열 2개를 생성합니다.

   a. Amazon SQS 콘솔을 엽니다.
   b. 대기열 생성을 선택합니다.
   c. 유형에서 표준을 선택합니다.
   d. 이름에 **`ticketPaymentQueue`**를 입력합니다.
   e. 액세스 정책에서 메서드 선택에 고급을 선택합니다.
   f. JSON 정책 상자에 다음 정책을 붙여넣습니다.

```
{
  "Version": "2008-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
```

```
        "Service": "sns.amazonaws.com"
      },
      "Action": "sqs:SendMessage",
      "Resource": "*",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:sns:us-east-1:123456789012:ticketTopic"
        }
      }
    }
  ]
}
```

이 액세스 정책에서 AWS 계정 번호(*123456789012*)를 자신의 번호로 바꾸고 그에 따라 AWS 리전(*us-east-1*)을 변경합니다.

g. 대기열 생성을 선택합니다.

h. 이 단계를 반복하여 **ticketFraudQueue**라는 두 번째 SQS 대기열을 만듭니다.

SQS 대기열 생성에 대한 자세한 정보는 Amazon Simple Queue Service 개발자 안내서의 Amazon SQS 대기열 생성(콘솔)을 참조하세요.

3. SNS 주제를 생성하려면

a. Amazon SNS 콘솔의 주제 페이지를 엽니다.

b. [Create topic]을 선택합니다.

c. 세부 정보에서 유형에 표준을 선택합니다.

d. 이름에 **ticketTopic**을 입력합니다.

e. 주제 생성을 선택합니다.

Amazon SNS 주제 생성에 대한 자세한 정보는 Amazon SNS 주제 생성에서 확인하세요.

4. 두 SQS 대기열 모두에서 SNS 주제를 구독합니다.

a. Amazon SNS 콘솔의 ticketTopic 주제 세부 정보 페이지에서 구독 생성을 선택합니다.

b. 세부 정보에서 프로토콜에 대해 Amazon SQS를 선택합니다.

c. 엔드포인트에서 ticketPaymentQueue 대기열의 Amazon 리소스 이름(ARN)을 선택합니다.

d. 구독 생성을 선택합니다.

e. 이 단계를 반복하여 ticketFraudQueue 대기열의 ARN으로 두 번째 구독을 생성합니다.

SNS 주제 구독에 대한 자세한 정보는 Amazon SNS 주제 구독에서 확인하세요. Amazon SQS 콘솔에서 SNS 주제에 대한 SQS 대기열을 구독할 수도 있습니다. 자세한 정보는 Amazon Simple Queue Service 개발자 안내서의 Amazon SQS 대기열에서 Amazon SNS 주제 구독(콘솔)을 참조하세요.

이 사용 사례 예에 대한 초기 리소스를 만들었습니다. 계속하려면 Firehose 전송 스트림 생성에서 확인하세요.

Firehose 전송 스트림 생성

이 페이지에서는 메시지 보관 및 분석 예제 사용 사례를 위해 Amazon Data Firehose 전송 스트림을 생성하는 방법을 설명합니다.

Firehose 전송 스트림을 만들려면

1. Amazon Kinesis 서비스 콘솔을 엽니다.
2. Firehose를 선택한 다음 전송 스트림 생성을 선택합니다.
3. 새 전송 스트림 페이지에서 전송 스트림 이름에 **`ticketUploadStream`**을 입력하고 다음을 선택합니다.
4. 프로세스 레코드 페이지에서 다음을 선택합니다.
5. 대상 선택 페이지에서 다음을 수행합니다.
   a. 대상에서 Amazon S3을 선택합니다.
   b. S3 대상에서 S3 버킷에 대해 최초로 생성한 S3 버킷을 선택합니다.
   c. 다음을 선택합니다.
6. 설정 구성 페이지에서 S3 버퍼 조건에 대해 다음을 수행합니다.
   - 버퍼 크기에 **`1`**을 입력합니다.
   - 버퍼 간격에 **`60`**을 입력합니다.

   Amazon S3 버퍼에 이 값을 사용하면 구성을 빠르게 테스트할 수 있습니다. 가장 먼저 만족되는 조건에 의해 S3 버킷으로의 데이터 전송이 트리거됩니다.
7. 설정 구성 페이지의 권한에서 필요한 권한이 자동으로 할당된 AWS Identity and Access Management(IAM) 역할을 생성하도록 선택합니다. 그런 다음, 다음을 선택합니다.
8. 검토 페이지에서 전송 스트림 생성을 선택합니다.

9. Kinesis Data Firehose 전송 스트림 페이지에서 방금 ticketUploadStream생성한 전송 스트림을 선택합니다 (). 세부 정보 탭에서 나중을 위해 스트림의 Amazon 리소스 이름(ARN)을 적어둡니다.

전송 스트림을 생성하는 방법에 대한 자세한 내용은 Amazon Data Firehose 개발자 안내서의 Amazon Data Firehose 전송 스트림 생성을 참조하십시오. IAM 역할 생성에 대한 자세한 정보는 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요.

필요한 권한을 사용하여 Firehose 전송 스트림을 만들었습니다. 계속하려면 Firehose 전송 스트림을 Amazon SNS 주제에 구독하기에서 확인하세요.

Firehose 전송 스트림을 Amazon SNS 주제에 구독하기

이 페이지에서는 메시지 아카이브 및 분석 예 사용 사례에 대해 다음을 만드는 방법을 설명합니다.

- Amazon SNS 구독으로 Amazon Data Firehose 전송 스트림에 레코드를 올릴 수 있게 해주는 AWS Identity and Access Management (IAM) 역할
- SNS 주제에 대한 Firehose 전송 스트림 구독

Amazon SNS 구독에 대한 IAM 역할을 생성하려면

1. IAM 콘솔에서 역할 페이지를 엽니다.
2. 역할 생성을 선택합니다.
3. 신뢰할 수 있는 엔터티 유형 선택(Select type of trusted entity)에서 AWS 서비스(service)를 선택합니다.
4. 사용 사례 선택에 SNS를 선택합니다. 그런 다음 다음: 권한을 선택합니다.
5. 다음: 태그를 선택합니다.
6. 다음: 검토를 선택합니다.
7. 검토 페이지의 역할 이름에 **ticketUploadStreamSubscriptionRole**을 입력합니다. 그런 다음 역할 생성을 선택합니다.
8. 역할을 생성하면 이름 (ticketUploadStreamSubscriptionRole) 을 선택합니다.
9. 역할의 요약 페이지에서 인라인 정책 추가를 선택합니다.
10. 정책 생성 페이지에서 JSON 탭을 선택한 후 상자에 다음 정책을 붙여넣습니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
        {
            "Action": [
                "firehose:DescribeDeliveryStream",
                "firehose:ListDeliveryStreams",
                "firehose:ListTagsForDeliveryStream",
                "firehose:PutRecord",
                "firehose:PutRecordBatch"
            ],
            "Resource": [
                "arn:aws:firehose:us-east-1:123456789012:deliverystream/
ticketUploadStream"
            ],
            "Effect": "Allow"
        }
    ]
}
```

이 정책에서 AWS 계정 번호(*123456789012*)를 자신의 번호로 바꾸고 그에 따라 AWS 리전(*us-east-1*)을 변경합니다.

11. 정책 검토를 선택합니다.
12. 정책 검토 페이지의 이름에 **FirehoseSnsPolicy**를 입력합니다. 그런 다음 정책 생성을 선택합니다.
13. 역할의 요약 페이지에서 나중을 위해 역할 ARN을 기록합니다.

IAM 역할 생성에 대한 자세한 정보는 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요.

Firehose 전송 스트림을 SNS 주제로 구독하려면

1. Amazon SNS 콘솔의 주제 페이지를 엽니다.
2. 구독 탭에서 구독 생성을 선택합니다.
3. 세부 정보에서 프로토콜에 대해 Amazon Data Firehose를 선택합니다.
4. Endpoint에 앞서 생성한 ticketUploadStream전송 스트림의 Amazon 리소스 이름 (ARN) 을 입력합니다. 예를 들면 **arn:aws:firehose:us-east-1:123456789012:deliverystream/ticketUploadStream**를 입력합니다.
5. 구독 역할 ARN에는 이전에 생성한 ticketUploadStreamSubscriptionRoleIAM 역할의 ARN을 입력합니다. 예를 들면 **arn:aws:iam::123456789012:role/ticketUploadStreamSubscriptionRole**을(를) 입력합니다.

6. 원시 메시지 전송 사용 확인란을 선택합니다.
7. 구독 생성을 선택합니다.

IAM 역할 및 SNS 주제 구독을 만들었습니다. 계속하려면 구성 테스트 및 쿼리에서 확인하세요.

구성 테스트 및 쿼리

이 페이지에서는 Amazon SNS 주제에 메시지를 게시하여 메시지 아카이브 및 분석 예 사용 사례를 테스트하는 방법을 설명합니다. 지침에는 실행하고 필요에 맞게 조정할 수 있는 예제 쿼리가 포함되어 있습니다.

구성을 테스트하려면

1. Amazon SNS 콘솔의 주제 페이지를 엽니다.
2. **ticketTopic** 주제를 선택을 선택합니다.
3. 메시지 게시를 선택합니다.
4. 주제에 메시지 게시 페이지에서 메시지 본문에 대해 다음을 입력합니다. 메시지 끝에 줄바꿈 문자를 추가합니다.

   ```
   {"BookingDate":"2020-12-15","BookingTime":"2020-12-15
    04:15:05","Destination":"Miami","FlyingFrom":"Vancouver","TicketNumber":"abcd1234"}
   ```

   다른 모든 옵션을 기본값으로 유지합니다.
5. 메시지 게시를 선택합니다.

   메시지 게시에 대한 자세한 정보는 Amazon SNS 메시지 게시에서 확인하세요.
6. 60초의 전송 스트림 간격 후에 Amazon Simple Storage Service(Amazon S3) 콘솔을 열고 최초로 생성한 Amazon S3 버킷을 선택합니다.

   버킷에 게시된 메시지가 표시됩니다.

데이터를 쿼리하려면

1. Amazon Athena 콘솔을 엽니다.
2. 쿼리를 실행합니다.

예를 들어 default 스키마의 notifications 테이블에 다음 데이터가 포함되어 있다고 가정합니다.

```
{"BookingDate":"2020-12-15","BookingTime":"2020-12-15
 04:15:05","Destination":"Miami","FlyingFrom":"Vancouver","TicketNumber":"abcd1234"}
{"BookingDate":"2020-12-15","BookingTime":"2020-12-15
 11:30:15","Destination":"Miami","FlyingFrom":"Omaha","TicketNumber":"efgh5678"}
{"BookingDate":"2020-12-15","BookingTime":"2020-12-15
 3:30:10","Destination":"Miami","FlyingFrom":"NewYork","TicketNumber":"ijkl9012"}
{"BookingDate":"2020-12-15","BookingTime":"2020-12-15
 12:30:05","Destination":"Delhi","FlyingFrom":"Omaha","TicketNumber":"mnop3456"}
```

최상위 대상을 찾으려면 다음 쿼리를 실행합니다.

```
SELECT destination
FROM default.notifications
GROUP BY destination
ORDER BY count(*) desc
LIMIT 1;
```

특정 날짜 및 시간 범위 동안 판매된 티켓을 쿼리하려면 다음과 같은 쿼리를 실행합니다.

```
SELECT *
FROM default.notifications
WHERE bookingtime
  BETWEEN TIMESTAMP '2020-12-15 10:00:00'
  AND TIMESTAMP '2020-12-15 12:00:00';
```

필요에 따라 두 샘플 쿼리를 모두 적용할 수 있습니다. Athena를 사용하여 쿼리를 실행하는 방법에 대한 자세한 정보는 Amazon Athena 사용 설명서의 시작하기를 참조하세요.

정리

테스트를 완료한 후 사용 요금이 발생하지 않도록 하려면 자습서 중에 생성한 다음 리소스를 삭제하세요.

- Amazon SNS 구독
- Amazon SNS 주제

- Amazon Simple Queue Service(Amazon SQS) 대기열
- Amazon S3 버킷
- Amazon Data Firehose 전송 스트림
- AWS Identity and Access Management(IAM) 역할 및 정책

AWS CloudFormation 템플릿 사용

Amazon SNS 메시지 아카이브 및 분석 예 사용 사례의 배포를 자동화하려면 다음 YAML 템플릿을 사용할 수 있습니다.

```
---
AWSTemplateFormatVersion: '2010-09-09'
Description: Template for creating an SNS archiving use case
Resources:
  ticketUploadStream:
    DependsOn:
    - ticketUploadStreamRolePolicy
    Type: AWS::KinesisFirehose::DeliveryStream
    Properties:
      S3DestinationConfiguration:
        BucketARN: !Sub 'arn:${AWS::Partition}:s3:::${ticketArchiveBucket}'
        BufferingHints:
          IntervalInSeconds: 60
          SizeInMBs: 1
        CompressionFormat: UNCOMPRESSED
        RoleARN: !GetAtt ticketUploadStreamRole.Arn
  ticketArchiveBucket:
    Type: AWS::S3::Bucket
  ticketTopic:
    Type: AWS::SNS::Topic
  ticketPaymentQueue:
    Type: AWS::SQS::Queue
  ticketFraudQueue:
    Type: AWS::SQS::Queue
  ticketQueuePolicy:
    Type: AWS::SQS::QueuePolicy
    Properties:
      PolicyDocument:
        Statement:
          Effect: Allow
          Principal:
```

```
          Service: sns.amazonaws.com
        Action:
          - sqs:SendMessage
        Resource: '*'
        Condition:
          ArnEquals:
            aws:SourceArn: !Ref ticketTopic
    Queues:
      - !Ref ticketPaymentQueue
      - !Ref ticketFraudQueue
ticketUploadStreamSubscription:
  Type: AWS::SNS::Subscription
  Properties:
    TopicArn: !Ref ticketTopic
    Endpoint: !GetAtt ticketUploadStream.Arn
    Protocol: firehose
    SubscriptionRoleArn: !GetAtt ticketUploadStreamSubscriptionRole.Arn
ticketPaymentQueueSubscription:
  Type: AWS::SNS::Subscription
  Properties:
    TopicArn: !Ref ticketTopic
    Endpoint: !GetAtt ticketPaymentQueue.Arn
    Protocol: sqs
ticketFraudQueueSubscription:
  Type: AWS::SNS::Subscription
  Properties:
    TopicArn: !Ref ticketTopic
    Endpoint: !GetAtt ticketFraudQueue.Arn
    Protocol: sqs
ticketUploadStreamRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: '2012-10-17'
      Statement:
      - Sid: ''
        Effect: Allow
        Principal:
          Service: firehose.amazonaws.com
        Action: sts:AssumeRole
ticketUploadStreamRolePolicy:
  Type: AWS::IAM::Policy
  Properties:
    PolicyName: FirehoseticketUploadStreamRolePolicy
```

```
      PolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Action:
          - s3:AbortMultipartUpload
          - s3:GetBucketLocation
          - s3:GetObject
          - s3:ListBucket
          - s3:ListBucketMultipartUploads
          - s3:PutObject
          Resource:
          - !Sub 'arn:aws:s3:::${ticketArchiveBucket}'
          - !Sub 'arn:aws:s3:::${ticketArchiveBucket}/*'
      Roles:
      - !Ref ticketUploadStreamRole
  ticketUploadStreamSubscriptionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - sns.amazonaws.com
          Action:
          - sts:AssumeRole
      Policies:
      - PolicyName: SNSKinesisFirehoseAccessPolicy
        PolicyDocument:
          Version: '2012-10-17'
          Statement:
          - Action:
            - firehose:DescribeDeliveryStream
            - firehose:ListDeliveryStreams
            - firehose:ListTagsForDeliveryStream
            - firehose:PutRecord
            - firehose:PutRecordBatch
            Effect: Allow
            Resource:
            - !GetAtt ticketUploadStream.Arn
```

Lambda 함수로 팬아웃

Amazon SNS와 AWS Lambda가 통합되어 Amazon SNS 알림으로 Lambda 함수를 호출할 수 있습니다. Lambda 함수가 구독하는 SNS 주제에 메시지가 게시되면 게시된 메시지의 페이로드와 함께 Lambda 함수가 호출됩니다. Lambda 함수가 입력 파라미터로 메시지 페이로드를 수신하고 메시지에서 정보를 조작하거나, 다른 SNS 주제에 메시지를 게시하거나, 다른 AWS 서비스에 메시지를 보낼 수 있습니다.

또한 Amazon SNS는 Lambda 엔드포인트에 전송된 메시지 알림을 위해 메시지 전송 상태 속성도 지원합니다. 자세한 정보는 Amazon SNS 메시지 전송 상태에서 확인하세요.

Prerequisites

Amazon SNS 알림을 사용하여 Lambda 함수를 호출하려면 다음이 필요합니다.

- Lambda 함수
- Amazon SNS 주제

Amazon SNS로 사용할 Lambda 함수 생성에 대한 자세한 내용은 Amazon SNS로 Lambda 사용을 참조하세요. Amazon SNS 주제 생성에 대한 자세한 내용은 주제 생성을 참조하세요.

옵트인 리전에서 기본적으로 활성화된 리전으로 메시지를 전송하는 데 Amazon SNS를 사용하는 경우, 보안 주체 `sns.amazonaws.com`을 `sns.<opt-in-region>.amazonaws.com`으로 대체하여 AWS Lambda 함수에서 생성된 정책을 변경해야 합니다.

예를 들어 미국 동부(버지니아 북부)의 Lambda 함수가 아시아 태평양(홍콩)의 SNS 주제를 구독하게 하려면 AWS Lambda 함수 정책의 보안 주체를 `sns.ap-east-1.amazonaws.com`으로 변경합니다. 옵트인 리전에는 2019년 3월 20일 이후에 시작된 모든 리전이 포함됩니다. 여기에는 아시아 태평양(홍콩), 중동(바레인), EU(밀라노) 및 아프리카(케이프타운)가 포함됩니다. 2019년 3월 20일 이전에 시작된 리전은 기본적으로 활성화되어 있습니다.

> **Note**
>
> AWS는 기본적으로 활성화되어 있는 리전에서 옵트인 리전으로의 Lambda의 리전 간 전송을 지원하지 않습니다. 또한 옵트인 리전에서 다른 옵트인 리전으로의 SNS 메시지의 리전 간 전달은 지원되지 않습니다.

함수에서 주제 구독

1. Amazon SNS 콘솔에 로그인합니다.
2. 탐색 창에서 주제를 선택합니다.
3. 주제 페이지에서 주제를 선택합니다.
4. 구독 섹션에서 구독 생성을 선택합니다.
5. 구독 생성 페이지의 세부 정보 섹션에서 다음을 수행합니다.

    a. 선택한 주제 ARN을 확인합니다.
    b. 프로토콜에서 AWS Lambda를 선택합니다.
    c. 엔드포인트에 함수의 ARN을 입력합니다.
    d. 구독 생성을 선택합니다.

Lambda 함수가 구독하는 SNS 주제에 메시지가 게시되면 게시된 메시지의 페이로드와 함께 Lambda 함수가 호출됩니다. 자습서를 포함하여 Amazon SNS로 AWS Lambda를 사용하는 방법에 대한 자세한 내용은 Amazon SNS로 AWS Lambda 사용을 참조하세요.

Amazon SQS 대기열로 팬아웃

Amazon SNS는 Amazon Simple Queue Service(Amazon SQS)와 긴밀하게 작동합니다. 이러한 서비스는 개발자에게 다양한 이점을 제공합니다. Amazon SNS를 사용하면 애플리케이션이 "푸시" 메커니즘을 통해 시간이 중요한 메시지를 여러 구독자에게 보낼 수 있으므로 업데이트를 주기적으로 확인하거나 "폴링"할 필요가 없습니다. Amazon SQS는 분산 애플리케이션이 폴링 모델을 통해 메시지를 교환하는 데 사용하는 메시지 대기열 서비스이며, 각 구성 요소를 동시에 사용할 필요 없이 전송 및 수신 구성 요소를 분리하는 데 사용할 수 있습니다. Amazon SNS와 Amazon SQS를 함께 사용하면 즉각적인 이벤트 알림을 필요로 하는 애플리케이션에 메시지를 전송할 수 있고, 다른 애플리케이션에서 나중에 처리할 수 있도록 메시지를 Amazon SNS 대기열에 계속 보관할 수도 있습니다.

Amazon SQS 대기열에서 Amazon SNS 주제를 구독하면 해당 주제에 메시지를 게시할 수 있으며 Amazon SNS는 구독한 대기열에 Amazon SQS 메시지를 보냅니다. Amazon SQS 메시지는 JSON 문서의 메시지에 대한 메타데이터와 함께 주제에 게시된 제목 및 메시지를 포함합니다. Amazon SQS 메시지는 다음의 JSON 문서와 유사합니다.

```
{
   "Type" : "Notification",
   "MessageId" : "63a3f6b6-d533-4a47-aef9-fcf5cf758c76",
```

```
   "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
   "Subject" : "Testing publish to subscribed queues",
   "Message" : "Hello world!",
   "Timestamp" : "2012-03-29T05:12:16.901Z",
   "SignatureVersion" : "1",
   "Signature" : "EXAMPLEnTrFPa3...",
   "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem",
   "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-west-2:123456789012:MyTopic:c7fe3a54-
ab0e-4ec2-88e0-db410a0f2bee"
}
```

Amazon SQS 대기열에서 Amazon SNS 주제 구독

Amazon SNS 주제가 Amazon SQS 대기열에 메시지를 보내도록 하려면 다음 중 하나를 수행하세요.

- 프로세스를 간소화하는 Amazon SQS 콘솔을 사용하세요. 자세한 정보는 Amazon Simple Queue Service 개발자 안내서의 Amazon SQS 대기열에서 Amazon SNS 주제 구독을 참조하세요.
- 다음 단계를 따릅니다.
  1. 메시지를 전송하고자 하는 대기열 및 대기열을 구독하고자 하는 주제의 Amazon Resource Name(ARN)을 파악합니다.
  2. Amazon SNS 주제에 대한 `sqs:SendMessage` 권한을 부여하여 대기열로 메시지를 전송할 수 있습니다.
  3. 대기열에서 Amazon SNS 주제를 구독합니다.
  4. Amazon SNS 주제에 게시하고 Amazon SQS 대기열에서 메시지를 읽을 수 있도록 IAM 사용자 또는 AWS 계정에 적절한 권한을 부여합니다.
  5. 주제에 대한 메시지를 게시하고 대기열에서 메시지를 읽어 테스트합니다.

다른 AWS 계정의 대기열에 메시지를 보내기 위해 주제를 생성하는 방법은 다른 계정의 Amazon SQS 대기열로 Amazon SNS 메시지 전송에서 확인하세요.

두 개의 대기열에 메시지를 전송하는 주제를 생성하는 AWS CloudFormation 템플릿을 보려면 AWS CloudFormation 템플릿을 사용하여 Amazon SQS 대기열에 메시지를 전송하는 주제 생성에서 확인하세요.

1단계: 대기열 및 주제의 ARN 획득

주제에 대기열을 구독할 때 사용자는 해당 대기열에 대한 ARN 사본이 필요합니다. 이와 마찬가지로 대기열에 메시지를 전송하도록 주제에 권한을 부여할 때 사용자는 주제에 대한 ARN 사본이 필요합니다.

대기열 ARN을 획득하려면 Amazon SQS 콘솔 또는 GetQueueAttributes API 작업을 사용할 수 있습니다.

Amazon SQS 콘솔로부터 대기열 ARN을 획득하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/sqs/에서 Amazon SQS 콘솔을 엽니다.
2. 획득하고자 하는 ARN 대기열 박스를 선택합니다.
3. 세부 정보 섹션에서 ARN 값을 복사하여 Amazon SNS 주제를 구독하는데 사용할 수 있도록 합니다.

주제 ARN을 가져오려면 Amazon SNS 콘솔, `sns-get-topic-attributes` 명령 또는 `GetQueueAttributes` API 작업을 사용할 수 있습니다.

Amazon SNS 콘솔로부터 주제 ARN을 획득하려면

1. Amazon SNS 콘솔에 로그인합니다.
2. 탐색 창에서 ARN을 가져오려는 주제를 선택합니다.
3. 세부 정보 섹션에서 ARN 값을 복사하여 대기열에 메시지를 전송하는 Amazon SNS 주제에 대한 권한을 부여하는데 사용합니다.

2단계: Amazon SQS 대기열에 메시지를 전송하도록 Amazon SNS 주제에 권한 부여

Amazon SNS 주제가 대기열에 메시지를 전송할 수 있으려면 Amazon SNS 주제가 `sqs:SendMessage` 작업을 수행하도록 허용하는 정책을 대기열에 설정해야 합니다.

대기열에서 주제를 구독하기 전에 주제 및 대기열이 필요합니다. 주제 또는 대기열을 아직 생성하지 않은 경우, 지금 생성합니다. 자세한 정보는 주제 생성을 참조하고 Amazon Simple Queue Service 개발자 안내서의 대기열 생성을 참조하세요.

대기열에 정책을 설정하기 위해 Amazon SQS 콘솔 또는 SetQueueAttributes API 작업을 사용할 수 있습니다. 시작하기 전에 대기열로 메시지를 전송하도록 허용하고자 하는 해당 주제의 ARN을 보유

하고 있는지 확인해야 합니다. 대기열에서 여러 주제를 구독하는 경우 정책은 각 주제에 대해 하나의 `Statement` 요소를 포함해야 합니다.

Amazon SQS 콘솔을 사용해 대기열에 SendMessage 정책 설정

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/sqs/에서 Amazon SQS 콘솔을 엽니다.
2. 정책을 설정하려는 대기열의 상자를 선택하고 액세스 정책 탭을 선택한 다음 편집을 선택합니다.
3. 액세스 정책 섹션에서 대기열에 액세스할 수 있는 사람을 정의합니다.

   - 주제의 작업을 허용하는 조건을 추가합니다.
   - 아래 예제와 같이 `Principal`을 Amazon SNS 서비스로 설정합니다
   - 혼동된 대리자 시나리오를 방지하기 위해 `aws:SourceArn` 또는 `aws:SourceAccount` 전역 조건 키를 사용합니다. 이러한 조건 키를 사용하려면 값을 주제의 ARN으로 설정합니다. 대기열에서 여러 주제를 구독하는 경우 `aws:SourceAccount`를 대신 사용할 수 있습니다.

   예를 들어, 다음 정책은 MyTopic이 MyQueue에 메시지를 전송하도록 허용합니다.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.amazonaws.com"
      },
      "Action": "sqs:SendMessage",
      "Resource": "arn:aws:sqs:us-east-2:123456789012:MyQueue",
      "Condition": {
        "ArnEquals": {
          "aws:SourceArn": "arn:aws:sns:us-east-2:123456789012:MyTopic"
        }
      }
    }
  ]
}
```

3단계: 대기열에서 Amazon SNS 주제 구독

주제를 통해 메시지를 전송하려면 대기열에서 Amazon SNS 주제를 구독해야 합니다. 사용자는 자체 ARN에 의해 해당 대기열을 지정합니다. 주제를 구독하려면 Amazon SNS 콘솔, `sns-subscribe` CLI 명령 또는 `Subscribe` API 작업을 사용할 수 있습니다. 시작하기 전에 구독하고자 하는 대기열의 ARN이 있어야 합니다.

1. Amazon SNS 콘솔에 로그인합니다.
2. 탐색 창에서 주제(Topics)를 선택합니다.
3. 주제 페이지에서 주제를 선택합니다.
4. **# ##** 페이지의 구독 페이지에서 구독 생성을 선택합니다.
5. 구독 생성 페이지의 세부 정보 섹션에서 다음을 수행합니다.

    a. 주제 ARN을 확인합니다.

    b. 프로토콜에서 Amazon SQS를 선택합니다.

    c. 엔드포인트에서 Amazon SQS 대기열의 ARN을 입력합니다.

    d. 구독 생성을 선택합니다.

    구독이 확인되면 새 구독의 구독 ID는 구독 ID를 표시합니다. 대기열 소유자가 구독을 생성하면 구독은 자동으로 확인되며 거의 즉시 활성화됩니다.

    일반적으로 사용자는 소유한 주제에 대해 사용자 계정의 자체 대기열을 구독합니다. 하지만 다른 계정의 대기열에서 주제를 구독할 수도 있습니다. 구독을 생성한 사용자가 대기열의 소유자가 아닐 경우(예. 계정 A의 사용자가 계정 A의 주제에 대해 계정 B의 대기열을 구독할 경우) 구독이 확인되어야 합니다. 다른 계정에서 대기열 구독 및 구독 확인에 대한 자세한 정보는 다른 계정의 Amazon SQS 대기열로 Amazon SNS 메시지 전송에서 확인하세요.

4단계: 사용자에게 적절한 주제 및 대기열 작업에 대한 권한 부여

사용자는 AWS Identity and Access Management(IAM)를 사용해 적절한 사용자만 Amazon SNS 주제를 게시하고 Amazon SQS 대기열에서 메시지를 읽기/삭제하도록 허용해야 합니다. IAM 사용자의 주제 및 대기열에 대한 작업 제어에 대한 자세한 정보는 Amazon SNS로 자격 증명 기반 정책 사용 및 Amazon Simple Queue Service 개발자 안내서의 Amazon SQS의 Identity and Access Management를 참조하세요.

주제 또는 대기열에 대한 액세스를 제어하는 다음의 두 가지 방법이 있습니다.

- IAM 사용자 또는 그룹에 정책을 추가합니다. 사용자에게 주제 또는 대기열에 대한 권한을 부여하는 가장 간단한 방법은 그룹을 생성하고 그룹에 적절한 정책을 추가한 후 사용자를 추가하는 것입니다. 그룹에서 사용자를 추가하거나 제거하는 것이 각각의 사용자에 대해 설정한 정책을 추적하는 것보다 훨씬 쉽습니다.
- 주제 또는 대기열에 대한 정책을 추가합니다. 또 다른 AWS 계정에 주제 또는 대기열에 대한 권한을 부여하고자 할 경우 사용할 수 있는 유일한 방법은 권한을 부여하고자 하는 AWS 계정가 주체인 정책을 추가하는 것입니다.

사용자는 대부분의 경우 첫 번째 방법을 사용해야 합니다(그룹에 정책을 적용하여 해당 그룹에 적절한 사용자를 추가 및 제거함으로써 사용자에 대한 권한을 관리). 또 다른 계정의 사용자에게 권한을 부여해야할 경우에는 두 번째 방법을 사용해야 합니다.

IAM 사용자 또는 그룹에 정책 추가

IAM 사용자 또는 그룹에 다음의 정책을 추가할 경우 해당 사용자 또는 그룹 구성원에게 MyTopic에 대한 `sns:Publish` 작업을 수행할 권한을 부여할 것입니다.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic"
    }
  ]
}
```

IAM 사용자 또는 그룹에 다음의 정책을 추가할 경우 해당 사용자 또는 그룹 구성원에게 대기열 MyQueue1 및 MyQueue2에 대한 `sqs:ReceiveMessage` 및 `sqs:DeleteMessage` 작업을 수행할 권한을 부여할 것입니다.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "sqs:ReceiveMessage",
        "sqs:DeleteMessage"
      ],
      "Resource": [
```

```
        "arn:aws:sqs:us-east-2:123456789012:MyQueue1",
        "arn:aws:sqs:us-east-2:123456789012:MyQueue2"
      ]
    }
  ]
}
```

주제 또는 대기열에 대한 정책 추가

다음의 정책 예제는 또 다른 계정에 주제 및 대기열에 대한 권한을 부여하는 방법을 보여줍니다.

> **Note**
>
> 사용자 계정에 있는 리소스에 대한 액세스 권한을 또 다른 AWS 계정에 부여할 때 해당 리소스에 대해 관리자 레벨의 액세스(와일드카드 액세스) 권한을 보유한 IAM 사용자에게도 권한을 부여합니다. 다른 계정의 다른 모든 IAM 사용자는 자동으로 사용자 리소스에 대한 액세스가 거부됩니다. 해당 AWS 계정의 특정 IAM 사용자에게 사용자의 리소스에 대한 액세스 권한을 부여하고자 할 경우 관리자 레벨의 액세스 권한을 보유한 계정 또는 IAM 사용자는 이러한 IAM 사용자들에게 리소스에 대한 권한을 위임해야 한다. 교차 계정 위임에 대한 자세한 정보는 IAM 사용 가이드의 교차 계정 액세스 권한 사용을 참조하세요.

계정 123456789012의 주제 MyTopic에 대해 다음의 정책을 추가했을 경우 사용자는 계정 111122223333에 해당 주제에 대한 `sns:Publish` 작업을 수행할 권한을 부여했을 것입니다.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111122223333"
      },
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic"
    }
  ]
}
```

계정 123456789012의 대기열 MyQueue에 대해 다음의 정책을 추가했을 경우 사용자는 계정 111122223333에 해당 대기열에 대한 `sqs:ReceiveMessage` 및 `sqs:DeleteMessage` 작업을 수행할 권한을 부여했을 것입니다.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111122223333"
      },
      "Action": [
        "sqs:DeleteMessage",
        "sqs:ReceiveMessage"
      ],
      "Resource": [
        "arn:aws:sqs:us-east-2:123456789012:MyQueue"
      ]
    }
  ]
}
```

5단계: 주제의 대기열 구독 테스트

주제를 게시하고 주제가 대기열에 전송한 메시지를 확인함으로써 주제의 대기열 구독을 테스트할 수 있습니다.

Amazon SNS 콘솔을 사용하여 주제를 게시하려면

1. AWS 계정 는 주제에 게시할 권한을 가진 AWS 계정 또는 IAM 사용자의 자격 증명을 통해 AWS Management Console에 로그인하고 https://console.aws.amazon.com/sns/에서 Amazon SNS 콘솔을 엽니다.
2. 탐색 창에서 주제를 선택하고 주제 게시를 선택합니다.
3. 제목 상자에 제목을 입력하고(예: **`Testing publish to queue`**), 메시지 상자에 텍스트를 입력한 후(예: **`Hello world!`**), 메시지 게시를 선택합니다. 다음의 메시지가 나타납니다. Your message has been successfully published.

Amazon SQS 콘솔을 사용한 주제에서의 메시지 확인

1. 대기열의 메시지를 볼 수 있는 권한이 있는 AWS 계정 또는 IAM 사용자의 자격 증명을 사용하여 AWS Management Console에 로그인하고 https://console.aws.amazon.com/sqs/에서 Amazon SQS 콘솔을 엽니다.
2. 주제를 구독하는 대기열을 선택합니다.

3. Send and receive messages(메시지 보내기 및 받기)를 선택한 다음 Poll for messages(메시지 폴링)를 선택합니다. 알림 유형으로 된 메시지가 표시됩니다.
4. 본문 열에서 추가 정보를 선택합니다. Message Details 상자는 주제에 게시한 제목 및 메시지로 된 JSON 문서를 담고 있습니다. 메시지는 다음의 JSON 문서와 유사합니다.

```
{
  "Type" : "Notification",
  "MessageId" : "63a3f6b6-d533-4a47-aef9-fcf5cf758c76",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Subject" : "Testing publish to subscribed queues",
  "Message" : "Hello world!",
  "Timestamp" : "2012-03-29T05:12:16.901Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEnTrFPa3...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/
SimpleNotificationService-f3ecfb7224c7233fe7bb5f59f96de52f.pem",
  "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
west-2:123456789012:MyTopic:c7fe3a54-ab0e-4ec2-88e0-db410a0f2bee"
}
```

5. 닫기를 선택합니다. 대기열에 알림 메시지를 전송하는 주제를 성공적으로 게시했습니다.

AWS CloudFormation 템플릿을 사용하여 Amazon SQS 대기열에 메시지를 전송하는 주제 생성

AWS CloudFormation을 통해 템플릿 파일을 사용하여 AWS 리소스 모음을 단일 유닛으로 생성 및 구성할 수 있습니다. 이 섹션은 대기열을 게시하는 주제를 쉽게 배포하는 예제 템플릿을 보여줍니다. 템플릿은 사용자가 두 개의 대기열을 생성하고, 대기열에 대한 주제를 생성하며, 해당 대기열에 정책을 추가하여 주제가 대기열에 메시지를 전송할 수 있도록 하고, 이러한 리소스에 대한 액세스를 제어하는 IAM 사용자 및 그룹을 생성하는 등의 설정 단계에서 사용됩니다.

AWS CloudFormation 템플릿을 사용하여 AWS 리소스를 배포하는 방법에 대한 자세한 정보는 AWS CloudFormation 사용 설명서의 시작하기를 참조하세요.

AWS 계정 계정에서 주제 및 대기열 설정을 위한 AWS CloudFormation 템플릿 사용

이 템플릿 예제에서는 주제에 게시할 수 있는 한 IAM 그룹 멤버의 적절한 권한과 대기열에서 메시지를 읽을 수 있는 다른 그룹 멤버의 권한을 사용하여 두 Amazon SQS 대기열에 메시지를 전송할 수 있는 Amazon SNS 주제를 생성합니다. 템플릿은 각 그룹에 추가될 IAM 사용자도 생성합니다.

템플릿 콘텐츠를 파일에 복사합니다. AWS CloudFormation 템플릿 페이지에서 템플릿을 다운로드할 수도 있습니다. 템플릿 페이지에서 AWS 서비스별로 샘플 템플릿 찾아보기를 선택한 다음 Amazon Simple Queue Service를 선택합니다.

MySNSTopic은 두 개의 Amazon SQS 대기열인 두 개의 구독된 엔드포인트를 게시하기 위해 설치합니다(MyQueue1 및 MyQueue2). MyPublishTopicGroup은 IAM 그룹으로 구성원은 Publish API 작업 또는 sns-publish 명령을 사용해 MySNSTopic을 게시할 권한을 갖고 있습니다. 템플릿은 IAM 사용자 MyPublishUser 및 MyQueueUser를 생성하고 이들에게 로그인 프로필 및 액세스 키를 부여합니다. 이 템플릿으로 스택을 생성한 사용자는 로그인 프로필에 입력 파라미터로 암호를 지정합니다. 템플릿은 MyPublishUserKey 및 MyQueueUserKey를 가진 두 IAM 사용자에 대한 액세스 키를 생성합니다. AddUserToMyPublishTopicGroup이 MyPublishTopicGroup에 MyPublishUser를 추가하여 사용자는 그룹에 할당된 권한을 갖게 됩니다.

MyRDMessageQueueGroup은 멤버가 ReceiveMessage 및 DeleteMessage API 작업을 사용하여 두 Amazon SQS 대기열에서 메시지를 읽고 삭제할 수 있는 권한이 있는 IAM 그룹입니다. AddUserToMyQueueGroup이 MyRDMessageQueueGroup에 MyQueueUser를 추가하여 사용자는 그룹에 할당된 권한을 갖게 됩니다. MyQueuePolicy는 두 대기열에 알림을 게시하도록 MySNSTopic에 대한 권한을 할당합니다.

다음 목록은 AWS CloudFormation 템플릿 콘텐츠를 보여줍니다.

```
{
  "AWSTemplateFormatVersion" : "2010-09-09",

  "Description" : "AWS CloudFormation Sample Template SNSToSQS: This Template creates
 an SNS topic that can send messages to
  two SQS queues with appropriate permissions for one IAM user to publish to the topic
 and another to read messages from the queues.
  MySNSTopic is set up to publish to two subscribed endpoints, which are two SQS queues
 (MyQueue1 and MyQueue2). MyPublishUser is an IAM user
  that can publish to MySNSTopic using the Publish API. MyTopicPolicy assigns that
 permission to MyPublishUser. MyQueueUser is an IAM user
  that can read messages from the two SQS queues. MyQueuePolicy assigns those
 permissions to MyQueueUser. It also assigns permission for
  MySNSTopic to publish its notifications to the two queues. The template creates
 access keys for the two IAM users with MyPublishUserKey
  and MyQueueUserKey. ***Warning*** you will be billed for the AWS resources used if
 you create a stack from this template.",

  "Parameters": {
    "MyPublishUserPassword": {
```

```
      "NoEcho": "true",
      "Type": "String",
      "Description": "Password for the IAM user MyPublishUser",
      "MinLength": "1",
      "MaxLength": "41",
      "AllowedPattern": "[a-zA-Z0-9]*",
      "ConstraintDescription": "must contain only alphanumeric characters."
    },
    "MyQueueUserPassword": {
      "NoEcho": "true",
      "Type": "String",
      "Description": "Password for the IAM user MyQueueUser",
      "MinLength": "1",
      "MaxLength": "41",
      "AllowedPattern": "[a-zA-Z0-9]*",
      "ConstraintDescription": "must contain only alphanumeric characters."
    }
  },


  "Resources": {
    "MySNSTopic": {
      "Type": "AWS::SNS::Topic",
      "Properties": {
        "Subscription": [{
            "Endpoint": {
              "Fn::GetAtt": ["MyQueue1", "Arn"]
            },
            "Protocol": "sqs"
          },
          {
            "Endpoint": {
              "Fn::GetAtt": ["MyQueue2", "Arn"]
            },
            "Protocol": "sqs"
          }
        ]
      }
    },
    "MyQueue1": {
      "Type": "AWS::SQS::Queue"
    },
    "MyQueue2": {
      "Type": "AWS::SQS::Queue"
```

```
    },
    "MyPublishUser": {
      "Type": "AWS::IAM::User",
      "Properties": {
        "LoginProfile": {
          "Password": {
            "Ref": "MyPublishUserPassword"
          }
        }
      }
    },
    "MyPublishUserKey": {
      "Type": "AWS::IAM::AccessKey",
      "Properties": {
        "UserName": {
          "Ref": "MyPublishUser"
        }
      }
    },
    "MyPublishTopicGroup": {
      "Type": "AWS::IAM::Group",
      "Properties": {
        "Policies": [{
          "PolicyName": "MyTopicGroupPolicy",
          "PolicyDocument": {
            "Statement": [{
              "Effect": "Allow",
              "Action": [
                "sns:Publish"
              ],
              "Resource": {
                "Ref": "MySNSTopic"
              }
            }]
          }
        }]
      }
    },
    "AddUserToMyPublishTopicGroup": {
      "Type": "AWS::IAM::UserToGroupAddition",
      "Properties": {
        "GroupName": {
          "Ref": "MyPublishTopicGroup"
        },
```

```
        "Users": [{
          "Ref": "MyPublishUser"
        }]
      }
    },
    "MyQueueUser": {
      "Type": "AWS::IAM::User",
      "Properties": {
        "LoginProfile": {
          "Password": {
            "Ref": "MyQueueUserPassword"
          }
        }
      }
    },
    "MyQueueUserKey": {
      "Type": "AWS::IAM::AccessKey",
      "Properties": {
        "UserName": {
          "Ref": "MyQueueUser"
        }
      }
    },
    "MyRDMessageQueueGroup": {
      "Type": "AWS::IAM::Group",
      "Properties": {
        "Policies": [{
          "PolicyName": "MyQueueGroupPolicy",
          "PolicyDocument": {
            "Statement": [{
              "Effect": "Allow",
              "Action": [
                "sqs:DeleteMessage",
                "sqs:ReceiveMessage"
              ],
              "Resource": [{
                  "Fn::GetAtt": ["MyQueue1", "Arn"]
                },
                {
                  "Fn::GetAtt": ["MyQueue2", "Arn"]
                }
              ]
            }]
          }
```

```
        }]
      }
    },
    "AddUserToMyQueueGroup": {
      "Type": "AWS::IAM::UserToGroupAddition",
      "Properties": {
        "GroupName": {
          "Ref": "MyRDMessageQueueGroup"
        },
        "Users": [{
          "Ref": "MyQueueUser"
        }]
      }
    },
    "MyQueuePolicy": {
      "Type": "AWS::SQS::QueuePolicy",
      "Properties": {
        "PolicyDocument": {
          "Statement": [{
            "Effect": "Allow",
            "Principal": {
              "Service": "sns.amazonaws.com"
            },
            "Action": ["sqs:SendMessage"],
            "Resource": "*",
            "Condition": {
              "ArnEquals": {
                "aws:SourceArn": {
                  "Ref": "MySNSTopic"
                }
              }
            }
          }]
        },
        "Queues": [{
          "Ref": "MyQueue1"
        }, {
          "Ref": "MyQueue2"
        }]
      }
    }
  },
  "Outputs": {
    "MySNSTopicTopicARN": {
```

```
      "Value": {
        "Ref": "MySNSTopic"
      }
    },
    "MyQueue1Info": {
      "Value": {
        "Fn::Join": [
          " ",
          [
            "ARN:",
            {
              "Fn::GetAtt": ["MyQueue1", "Arn"]
            },
            "URL:",
            {
              "Ref": "MyQueue1"
            }
          ]
        ]
      }
    },
    "MyQueue2Info": {
      "Value": {
        "Fn::Join": [
          " ",
          [
            "ARN:",
            {
              "Fn::GetAtt": ["MyQueue2", "Arn"]
            },
            "URL:",
            {
              "Ref": "MyQueue2"
            }
          ]
        ]
      }
    },
    "MyPublishUserInfo": {
      "Value": {
        "Fn::Join": [
          " ",
          [
            "ARN:",
```

```
            {
              "Fn::GetAtt": ["MyPublishUser", "Arn"]
            },
            "Access Key:",
            {
              "Ref": "MyPublishUserKey"
            },
            "Secret Key:",
            {
              "Fn::GetAtt": ["MyPublishUserKey", "SecretAccessKey"]
            }
          ]
        ]
      }
    },
    "MyQueueUserInfo": {
      "Value": {
        "Fn::Join": [
          " ",
          [
            "ARN:",
            {
              "Fn::GetAtt": ["MyQueueUser", "Arn"]
            },
            "Access Key:",
            {
              "Ref": "MyQueueUserKey"
            },
            "Secret Key:",
            {
              "Fn::GetAtt": ["MyQueueUserKey", "SecretAccessKey"]
            }
          ]
        ]
      }
    }
  }
}
```

HTTP(S) 엔드포인트로 팬아웃

Amazon SNS를 사용하여 하나 이상의 HTTP 또는 HTTPS 엔드포인트에 알림 메시지를 전송할 수 있습니다. 주제에 대한 엔드포인트를 구독할 때 사용자는 주제에 대한 알림을 게시할 수 있으며 Amazon

SNS는 HTTP POST 요청을 전송하는 한편 구독된 엔드포인트에 알림 콘텐츠를 전송합니다. 사용자는 엔드포인트 구독 시 Amazon SNS가 엔드포인트에 POST 요청을 전송하기 위해 HTTP를 사용하는지 또는 HTTPS를 사용하는지 여부를 선택합니다. HTTPS를 사용할 경우 Amazon SNS에서 다음에 대한 지원을 이용할 수 있습니다.

- 서버 이름 표시(SNI) - 이를 통해 Amazon SNS는 이 기능을 사용하여 여러 도메인을 호스팅하기 위해 여러 인증서가 필요한 서버와 같이 SNI가 필요한 HTTPS 엔드포인트를 지원합니다. SNI에 대한 자세한 정보는 서버 이름 표시를 참조하세요.

- 기본 및 다이제스트 액세스 인증 - HTTP POST 요청에 대해 HTTPS URL에 사용자 이름과 암호를 지정할 수 있습니다(예: `https://user:password@domain.com` 또는 `https://user@domain.com`). 사용자 이름과 암호는 HTTPS를 사용할 때 설정되는 SSL 연결을 통해 암호화됩니다. 도메인 이름만 일반 텍스트로 전송됩니다. 기본 및 다이제스트 액세스 인증에 대한 자세한 정보는 RFC-2617을 참조하세요.

> **Important**
>
> Amazon SNS는 현재 프라이빗 HTTP(S) 엔드포인트를 지원하지 않습니다. HTTPS URL은 API 액세스 권한을 부여한 보안 주체에 대한 Amazon SNS `GetSubscriptionAttributes` API 작업에서만 검색할 수 있습니다.

> **Note**
>
> 클라이언트 서비스는 `HTTP/1.1 401 Unauthorized` 헤더 응답을 지원할 수 있어야 합니다.

요청은 JSON 문서의 알림에 대한 메타데이터와 함께 주제에 게시된 제목 및 메시지를 포함합니다. 요청은 다음의 HTTP POST 요청과 유사합니다. 요청 본문의 HTTP 헤더 및 JSON 형식에 대한 세부 정보는 HTTP/HTTPS 헤더 및 HTTP/HTTPS 알림 JSON 형식에서 확인하세요.

```
POST / HTTP/1.1
    x-amz-sns-message-type: Notification
    x-amz-sns-message-id: da41e39f-ea4d-435a-b922-c6aae3915ebe
    x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
    x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55
```

```
    Content-Length: 761
    Content-Type: text/plain; charset=UTF-8
    Host: ec2-50-17-44-49.compute-1.amazonaws.com
    Connection: Keep-Alive
    User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "Notification",
  "MessageId" : "da41e39f-ea4d-435a-b922-c6aae3915ebe",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Subject" : "test",
  "Message" : "test message",
  "Timestamp" : "2012-04-25T21:49:25.719Z",
  "SignatureVersion" : "1",
  "Signature" :
 "EXAMPLElDMXvB8r9R83tGoNn0ecwd5UjllzsvSvbItzfaMpN2nk5HVSw7XnOn/49IkxDKz8YrlH2qJXj2iZB0Zo2O71c4
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem",
   "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55"
}
```

주제

- HTTP/S 엔드포인트에서 주제 구독
- Amazon SNS 메시지의 서명 확인
- 메시지 형식 구문 분석

HTTP/S 엔드포인트에서 주제 구독

이 섹션의 페이지에서는 HTTP/S 엔드포인트에서 Amazon SNS 주제를 구독하는 방법에 대해 설명합니다.

주제

- 1단계: Amazon SNS 메시지를 처리하도록 엔드포인트를 준비합니다.
- 2단계: Amazon SNS 주제에 대한 HTTP/HTTPS 엔드포인트 구독
- 3단계: 구독 확인
- 4단계: 구독의 전송 정책 설정(선택 사항)
- 5단계: 사용자에게 주제를 게시할 권한 부여(선택 사항)

- 6단계: HTTP/HTTPS 엔드포인트에 메시지 전송

1단계: Amazon SNS 메시지를 처리하도록 엔드포인트를 준비합니다.

사용자는 주제에 대한 HTTP 엔드포인트 또는 HTTPS 엔드포인트를 구독하기 전에 HTTP 엔드포인트 또는 HTTPS 엔드포인트가 구독 확인 및 알림 메시지를 전송하기 위해 Amazon SNS가 사용하는 HTTP POST 요청을 처리할 수 있는 능력이 있는지 확인해야 합니다. 일반적으로 이는 Amazon SNS에서 HTTP 요청을 처리하는 웹 애플리케이션의 생성 및 배포(예. endpoint host가 Linux, Apache 및 Tomcat을 실행할 경우 Java servlet)를 의미합니다. HTTP 엔드포인트 구독 시 Amazon SNS는 이를 구독 확인 요청에 전송합니다. 사용자가 구독을 생성할 때 Amazon SNS는 이 요청을 전송하므로 엔드포인트는 이 요청을 수신하고 처리하도록 준비되어야 합니다. Amazon SNS는 구독을 확인할 때까지 엔드포인트에 알림을 보내지 않습니다. 구독을 확인하고 나면 Amazon SNS는 구독하는 주제에 대해 게시 작업이 수행될 때 엔드포인트에 알림을 전송합니다.

구독 확인 및 알림 메시지 처리를 위한 엔드포인트 설정

1. 사용자의 코드는 Amazon SNS가 엔드포인트에 전송한 HTTP POST 요청의 HTTP 헤더를 읽어야 합니다. 사용자의 코드는 Amazon SNS가 사용자에게 전송한 메시지 유형을 나타내는 헤더 필드 `x-amz-sns-message-type`을 찾아야 합니다. 헤더를 확인함으로써 사용자는 HTTP 요청의 본문을 분석하지 않고도 메시지 유형을 결정할 수 있습니다. 처리해야 할 두 가지 유형 `SubscriptionConfirmation`와 `Notification`이 있습니다. `UnsubscribeConfirmation` 메시지는 주제에서 구독이 삭제된 때에만 사용됩니다.

   HTTP 헤더에 대한 세부 정보는 HTTP/HTTPS 헤더에서 확인하세요. 다음의 HTTP POST 요청은 구독 확인 메시지의 예입니다.

```
POST / HTTP/1.1
    x-amz-sns-message-type: SubscriptionConfirmation
    x-amz-sns-message-id: 165545c9-2a5c-472c-8df2-7ff2be2b3b1b
    x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
    Content-Length: 1336
    Content-Type: text/plain; charset=UTF-8
    Host: example.com
    Connection: Keep-Alive
    User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "SubscriptionConfirmation",
  "MessageId" : "165545c9-2a5c-472c-8df2-7ff2be2b3b1b",
  "Token" : "2336412f37f...",
```

```
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Message" : "You have chosen to subscribe to the topic arn:aws:sns:us-
west-2:123456789012:MyTopic.\nTo confirm the subscription, visit the SubscribeURL
 included in this message.",
  "SubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-
west-2:123456789012:MyTopic&Token=2336412f37...",
  "Timestamp" : "2012-04-26T20:45:04.751Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEpH+...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/
SimpleNotificationService-f3ecfb7224c7233fe7bb5f59f96de52f.pem"
}
```

2. 사용자의 코드는 HTTP POST 요청 및 content-type text/plain 본문의 JSON 문서를 분석하여 Amazon SNS 메시지를 구성하는 이름-값 쌍을 읽어야 합니다. JSON 분석을 사용해 제어 문자의 escaped 상태를 ASCII 문자 값으로 변환 처리합니다(예. \n을 줄바꿈 문자로 변환). Jackson JSON Processor 등의 기존 JSON 구문 분석기를 사용하거나 자체적으로 쓸 수 있습니다. 제목 및 메시지 필드의 텍스트를 유효한 JSON으로 보내려면 Amazon SNS에서 일부 제어 문자를 JSON 문서에 포함될 수 있는 이스케이프된 표현으로 변환해야 합니다. 엔드포인트에 전송된 POST 요청의 본문에 있는 JSON 문서를 수신하면 사용자는 주제에 게시된 원래 제목과 메시지의 정확한 표현을 원하는 경우 이스케이프된 문자를 다시 원본 문자 값으로 변환해야 합니다. 서명은 원본 형식의 메시지 및 제목을 서명할 문자열의 일부로 사용하기 때문에 이는 알림의 서명을 확인하고자 할 경우 아주 중요합니다.

3. 사용자의 코드는 Amazon SNS가 보낸 알림의 신뢰성, 구독 확인 또는 구독 해지 확인 메시지를 확인해야 합니다. 엔드포인트는 Amazon SNS 메시지에 포함된 정보를 사용해 서명을 재생성할 수 있으며 사용자는 Amazon SNS가 메시지로 보낸 서명을 자신의 서명과 비교함으로써 메시지의 콘텐츠를 확인할 수 있습니다. 메시지의 서명 확인에 대한 자세한 정보는 Amazon SNS 메시지의 서명 확인에서 확인하세요.

4. 사용자의 코드는 헤더 필드 `x-amz-sns-message-type`가 지정하는 유형에 기초하여 HTTP 요청의 본문에 담긴 JSON 문서를 읽고 메시지를 처리해야 합니다. 다음은 두 가지 주요 메시지 유형 처리 지침입니다.

   SubscriptionConfirmation

   `SubscribeURL`의 값을 읽고 해당 URL을 방문합니다. 구독을 확인하고 엔드포인트에서 알림 수신을 시작하려면 `SubscribeURL` URL을 방문해야 합니다(예. URL에 HTTP GET 요청을 전송). `SubscribeURL`를 확인하려면 이전 단계의 HTTP 요청 예제를 확인합니다. `SubscriptionConfirmation` 메시지의 형식에 대한 자세한 정보는 HTTP/HTTPS 구독 확

인 JSON 형식에서 확인하세요. URL을 방문하면 다음의 XML 문서와 같은 응답을 받게 됩니다. 문서는 ConfirmSubscriptionResult 요소 내 엔드포인트에 대한 구독 ARN을 되돌립니다.

```
<ConfirmSubscriptionResponse xmlns="http://sns.amazonaws.com/doc/2010-03-31/">
   <ConfirmSubscriptionResult>
      <SubscriptionArn>arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55</
SubscriptionArn>
   </ConfirmSubscriptionResult>
   <ResponseMetadata>
      <RequestId>075ecce8-8dac-11e1-bf80-f781d96e9307</RequestId>
   </ResponseMetadata>
</ConfirmSubscriptionResponse>
```

사용자는 SubscribeURL 방문 외에도 ConfirmSubscription 작업과 함께 Token(SubscriptionConfirmation 메시지의 해당 값으로 설정된)을 사용해 구독을 확인할 수 있습니다. 주제 소유자 및 구독 소유자만 엔드포인트를 구독 해지할 수 있도록 허용하고자 하는 경우 AWS 서명을 사용해 ConfirmSubscription 작업을 호출합니다.

알림

Subject 및 Message에 대한 값을 읽어 주제에 게시된 알림 정보를 획득합니다.

Notification 메시지 형식에 대한 세부 정보는 HTTP/HTTPS 헤더에서 확인하세요. 다음의 HTTP POST 요청은 endpoint example.com에 전송된 알림 메시지의 예입니다.

```
POST / HTTP/1.1
    x-amz-sns-message-type: Notification
    x-amz-sns-message-id: 22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324
    x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
    x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:c9135db0-26c4-47ec-8998-413945fb5a96
    Content-Length: 773
    Content-Type: text/plain; charset=UTF-8
    Host: example.com
    Connection: Keep-Alive
    User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "Notification",
  "MessageId" : "22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324",
```

```
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Subject" : "My First Message",
  "Message" : "Hello world!",
  "Timestamp" : "2012-05-02T00:54:06.655Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEw6JRN...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/
SimpleNotificationService-f3ecfb7224c7233fe7bb5f59f96de52f.pem",
  "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
west-2:123456789012:MyTopic:c9135db0-26c4-47ec-8998-413945fb5a96"
}
```

5. 엔드포인트가 적절한 상태 코드로 Amazon SNS의 HTTP POST 메시지에 응답하도록 확인해야 합니다. 연결은 15초 내에 끊어집니다. 엔드포인트가 연결 시간이 끊기기 전까지 응답하지 않거나 엔드포인트가 상태 코드를 200–4xx 범위 외로 되돌릴 경우 Amazon SNS는 메시지 전송이 실패한 것으로 간주합니다.
6. Amazon SNS로부터의 메시지 전송 재시도를 처리할 수 있도록 코드를 확인해야 합니다. Amazon SNS는 엔드포인트에서 성공적인 응답을 수신하지 않을 경우 메시지를 다시 전송하는 시도를 합니다. 이는 구독 확인 메시지를 포함한 모든 메시지에 적용됩니다. 기본 설정에서 메시지의 초기 전송이 실패할 경우 Amazon SNS는 최대 3회 재시도를 실시하며 실패한 시도 사이에 설정된 지연 시간은 20초입니다.

   > **Note**
   >
   > 15초가 경과하면 메시지 요청 시간이 초과됩니다. 따라서 시간이 초과되어 메시지 전송이 실패하는 경우 Amazon SNS는 이전 전송 시도의 약 35초 후에 재시도를 합니다. 엔드포인트에 대해 다른 전송 정책을 설정할 수 있습니다.

   Amazon SNS는 `x-amz-sns-message-id` 헤더 필드를 사용하여 Amazon SNS 주제에 게시된 각 메시지를 고유하게 식별합니다. 사용자는 수신 메시지와 처리한 메시지의 ID를 비교함으로써 메시지가 재전송을 시도하는지 여부를 판단할 수 있습니다.
7. HTTPS 엔드포인트를 구독할 경우 엔드포인트에 신뢰할 수 있는 인증 기관(CA)의 서버 인증서가 있는지 확인해야 합니다. Amazon SNS는 Amazon SNS가 신뢰하는 CA의 서명이 있는 서버 인증서를 보유한 HTTPS 엔드포인트에만 메시지를 보냅니다.
8. Amazon SNS 메시지를 수신하기 위해 생성한 코드를 배포합니다. 엔드포인트를 구독할 때 엔드포인트는 적어도 구독 확인 메시지를 받을 준비가 되어야 합니다.

2단계: Amazon SNS 주제에 대한 HTTP/HTTPS 엔드포인트 구독

주제를 통해 메시지를 HTTP 엔드포인트 또는 HTTPS 엔드포인트에 전송하려면 엔드포인트가 Amazon SNS 주제를 구독하게 등록해야 합니다. 사용자는 해당 URL을 사용하여 엔드포인트를 지정합니다. 주제를 구독하려면 Amazon SNS 콘솔, sns-subscribe 명령 또는 구독 API 작업을 사용할 수 있습니다. 시작하기 전에 사용자는 구독하고자 하는 엔드포인트의 URL을 보유하는지 확인해야 하며 단계 1에서 설명한 대로 엔드포인트가 확인 및 알림 메시지를 수신할 수 있도록 준비되어야 합니다.

Amazon SNS 콘솔을 이용하여 HTTP 엔드포인트 또는 HTTPS 엔드포인트에서 주제 구독

1. Amazon SNS 콘솔에 로그인합니다.
2. 탐색 창에서 주제(Topics)를 선택합니다.
3. 구독 생성(Create subscription)을 선택합니다.
4. 프로토콜(Protocol) 드롭다운 목록에서 HTTP 또는 HTTPS를 선택합니다.
5. 엔드포인트(Endpoint) 상자에서 주제가 메시지를 전송하려는 엔드포인트의 URL을 붙여 넣은 다음 구독 생성(Create subscription)을 선택합니다.
6. 확인 메시지가 표시됩니다. 닫기를 선택합니다.

   새 구독의 구독(Subscription ID)는 PendingConfirmation에 표시됩니다. 구독 확인 시 구독(Subscription ID)는 구독 ID(Subscription ID)에 표시됩니다.

3단계: 구독 확인

엔드포인트에서 구독을 설정한 후에는 Amazon SNS가 엔드포인트에 구독 확인 메시지를 전송합니다. 사용자는 이미 1단계(Step 1)에서 설명한 작업을 수행하는 엔드포인트에 배포될 코드를 보유해야 합니다. 구체적으로, 엔드포인트의 코드는 구독 확인 메시지로부터 `SubscribeURL` 값을 검색하여 `SubscribeURL` 자체에 의해 지정된 위치를 방문하거나 사용자에게 제공하여 사용자가 수동으로 `SubscribeURL`을 방문할 수 있도록 해야 합니다(예: 웹 브라우저 사용). Amazon SNS는 구독이 확인되기 전에는 엔드포인트에 메시지를 보내지 않습니다. `SubscribeURL` 방문 시, 응답은 구독에 대한 ARN을 지정하는 요소 `SubscriptionArn`를 담고 있는 XML 문서를 포함합니다. 사용자는 Amazon SNS 콘솔을 사용하여 구독 확인을 검증할 수도 있습니다. 구독 ID는 구독을 처음 추가했을 때 확인한 `PendingConfirmation` 값 대신 구독에 대한 ARN을 표시합니다.

4단계: 구독의 전송 정책 설정(선택 사항)

기본 설정에서 메시지의 초기 전송이 실패할 경우 Amazon SNS는 최대 3회 재시도를 실시하며 실패한 시도 사이에 설정된 지연 시간은 20초입니다. 1단계에서 논의되었듯 엔드포인트는 재시도되

는 메시지를 처리할 수 있는 코드를 보유해야 합니다. 사용자는 주제 또는 구독에 대한 전송 정책을 설정함으로써 Amazon SNS가 전송 실패 메시지를 재전송하는 빈도 및 간격을 제어할 수 있습니다. `DeliveryPolicy`에서 HTTP/S 알림의 콘텐츠 유형을 지정할 수도 있습니다. 자세한 내용은 HTTP/S 전송 정책 생성 섹션을 참조하세요.

5단계: 사용자에게 주제를 게시할 권한 부여(선택 사항)

기본 설정에서 주제 소유자는 주제를 게시할 권한을 갖습니다. 다른 사용자 및 애플리케이션을 활성화하여 주제를 게시하려면 AWS Identity and Access Management(IAM)를 사용하여 주제에 대한 게시 권한을 부여해야 합니다. IAM 사용자에게 Amazon SNS 작업에 대한 권한을 부여하는 방법에 대한 자세한 정보는 Amazon SNS로 자격 증명 기반 정책 사용에서 확인하세요.

주제에 대한 액세스를 제어하는 다음의 두 가지 방법이 있습니다.

- IAM 사용자 또는 그룹에 정책을 추가합니다. 사용자에게 주제에 대한 권한을 부여하는 가장 간단한 방법은 그룹을 생성하고 그룹에 적절한 정책을 추가한 후 사용자를 추가하는 것입니다. 그룹에서 사용자를 추가하거나 제거하는 것이 각각의 사용자에 대해 설정한 정책을 추적하는 것보다 훨씬 쉽습니다.
- 주제에 정책을 추가합니다. 또 다른 AWS 계정에 주제에 대한 권한을 부여하고자 할 경우 사용할 수 있는 유일한 방법은 권한을 부여하고자 하는 AWS 계정이 주체인 정책을 추가하는 것입니다.

사용자는 대부분의 경우 첫 번째 방법을 사용해야 합니다(그룹에 정책을 적용하여 해당 그룹에 적절한 사용자를 추가 및 제거함으로써 사용자에 대한 권한을 관리). 또 다른 계정의 사용자에게 권한을 부여해야할 경우에는 두 번째 방법을 사용합니다.

IAM 사용자 또는 그룹에 다음의 정책을 추가할 경우 해당 사용자 또는 그룹 구성원에게 MyTopic에 대한 `sns:Publish` 작업을 수행할 권한을 부여할 것입니다.

```
{
  "Statement":[{
    "Sid":"AllowPublishToMyTopic",
    "Effect":"Allow",
    "Action":"sns:Publish",
    "Resource":"arn:aws:sns:us-east-2:123456789012:MyTopic"
  }]
}
```

다음의 정책 예제는 또 다른 계정에 주제에 대한 권한을 부여하는 방법을 나타냅니다.

> **Note**
>
> 사용자 계정에 있는 리소스에 대한 액세스 권한을 또 다른 AWS 계정에 부여할 때 해당 리소스에 대해 관리자 레벨의 액세스(와일드카드 액세스) 권한을 보유한 IAM 사용자에게도 권한을 부여합니다. 다른 계정의 다른 모든 IAM 사용자는 자동으로 사용자 리소스에 대한 액세스가 거부됩니다. 해당 AWS 계정의 특정 IAM 사용자에게 사용자의 리소스에 대한 액세스 권한을 부여하고자 할 경우 관리자 레벨의 액세스 권한을 보유한 계정 또는 IAM 사용자는 이러한 IAM 사용자들에게 리소스에 대한 권한을 위임해야 한다. 교차 계정 위임에 대한 자세한 정보는 IAM 사용 가이드의 교차 계정 액세스 권한 사용을 참조하세요.

계정 123456789012의 주제 MyTopic에 대해 다음의 정책을 추가했을 경우 사용자는 계정 111122223333에 해당 주제에 대한 `sns:Publish` 작업을 수행할 권한을 부여했을 것입니다.

```
{
  "Statement":[{
    "Sid":"Allow-publish-to-topic",
    "Effect":"Allow",
      "Principal":{
        "AWS":"111122223333"
      },
    "Action":"sns:Publish",
    "Resource":"arn:aws:sns:us-east-2:123456789012:MyTopic"
  }]
}
```

6단계: HTTP/HTTPS 엔드포인트에 메시지 전송

주제를 게시함으로써 주제의 구독에 메시지를 전송할 수 있습니다. 주제에 게시하려면 Amazon SNS 콘솔, `sns-publish` CLI 명령 또는 `Publish` API 작업을 사용할 수 있습니다.

1단계를 따랐다면 엔드포인트에 배포한 코드는 알림을 처리해야 합니다.

Amazon SNS 콘솔을 사용하여 주제를 게시하려면

1. AWS 계정 는 주제에 게시할 권한을 가진 AWS 계정 또는 IAM 사용자의 자격 증명을 통해 AWS Management Console에 로그인하고 https://console.aws.amazon.com/sns/에서 Amazon SNS 콘솔을 엽니다.
2. 탐색 창에서 주제(Topics)를 선택한 후 주제를 선택합니다.

3. 메시지 게시(Publish message) 버튼을 선택합니다.
4. 제목(Subject) 상자에 제목을 입력합니다(예: **`Testing publish to my endpoint`**).
5. 메시지(Message) 상자에 텍스트를 입력하고(예: **`Hello world!`**), 메시지 게시(Publish message)를 선택합니다.

   다음의 메시지가 나타납니다. Your message has been successfully published.

Amazon SNS 메시지의 서명 확인

Amazon SNS에서 HTTP 엔드포인트로 보낸 메시지의 신뢰성을 확인하기 위해 메시지 서명을 확인할 수 있습니다. 메시지의 진위 여부를 확인해야 하는 두 가지 경우가 있습니다. 첫째, Amazon SNS가 주제를 구독했다는 메시지를 HTTP 엔드포인트로 보낼 때입니다. 둘째, `Subscribe` 또는 `Unsubscribe` API 작업을 실행할 때 Amazon SNS가 HTTP 엔드포인트로 확인 메시지를 보낼 때입니다.

Amazon SNS가 보낸 메시지를 확인할 때 다음을 수행해야 합니다.

- Amazon SNS에서 인증서를 받을 때는 항상 HTTPS를 사용합니다.
- 인증서의 신뢰성을 확인합니다.
- 인증서가 Amazon SNS에서 수신되었는지 확인합니다.
- 가능한 경우, 지원되는 Amazon SNS용 AWS SDK 중 하나를 사용하여 메시지의 유효성을 검사하고 확인합니다.
- Amazon SNS 메시지가 원하는 `TopicArn`에서 수신되었는지 확인합니다.

Amazon SNS는 두 가지 메시지 서명 버전을 지원합니다.

- `SignatureVersion1`: Amazon SNS는 메시지의 SHA1 해시를 기반으로 서명을 생성합니다.
- `SignatureVersion1`: Amazon SNS는 메시지의 SHA256 해시를 기반으로 서명을 생성합니다.

Amazon SNS 주제의 메시지 서명 버전 구성

기본적으로 Amazon SNS 주제는 `SignatureVersion` 1을 사용합니다. Amazon SNS 주제에 대한 해싱 알고리즘(`SignatureVersion` 1(SHA1) 또는 `SignatureVersion` 2(SHA256))을 선택하려면 `SetTopicAttributes` API 작업을 사용할 수 있습니다.

다음 코드 예제는 AWS CLI를 사용하여 주제 속성 `SignatureVersion`을 설정하는 방법을 보여줍니다.

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-east-2:123456789012:MyTopic \
    --attribute-name SignatureVersion \
    --attribute-value 2
```

HTTP 쿼리 기반 요청을 사용할 때 Amazon SNS 메시지의 서명을 확인하려면

1. Amazon SNS가 엔드포인트에 전송한 HTTP POST 요청 본문의 JSON 문서에서 이름/값 쌍을 추출합니다. 일부 이름/값 쌍의 값을 사용하여 서명할 문자열을 생성합니다. Amazon SNS 메시지의 서명을 확인할 때 이스케이프된 컨트롤 문자를 `Message` 및 `Subject` 값의 원본 문자로 변환하는 것은 무척 중요합니다. 이러한 값은 서명할 문자열의 일부로 사용 시 원본 형식이어야 합니다. JSON 문서를 분석하는 방법에 대한 자세한 내용은 1단계: Amazon SNS 메시지를 처리하도록 엔드포인트를 준비합니다.에서 확인하세요.

   `SignatureVersion`은 Amazon SNS에서 메시지 서명을 생성하는 데 사용하는 서명 버전을 알려줍니다. 사용자는 서명 버전에서 서명을 생성하는 방법에 따르는 요구 사항을 결정할 수 있습니다. 알림의 경우 Amazon SNS는 현재 서명 버전 1 및 2를 지원합니다. 이 섹션에서는 이러한 서명 버전을 사용하여 서명을 확인하는 단계를 제공합니다.
2. Amazon SNS가 메시지에 서명하는데 사용한 X509 인증서를 획득합니다. `SigningCertURL` 값은 메시지에 대한 전자 서명을 생성하는데 사용한 X509 인증서의 위치를 나타냅니다. 이 위치에서 인증서를 검색합니다.
3. 인증서로부터 공개 키를 추출합니다. `SigningCertURL`에 의해 지정된 인증서로부터의 공개 키는 메시지의 신뢰성 및 무결성을 확인하는데 사용합니다.
4. 메시지 유형을 결정합니다. 서명할 문자열 형식은 `Type` 값에 의해 지정된 메시지 유형에 따릅니다.
5. 서명할 문자열을 생성합니다. 서명할 문자열은 메시지에서 지정된 이름/값 쌍의 줄바꿈 문자로 구분된 목록입니다. 각각의 이름/값 쌍은 이름, 줄바꿈 문자, 값, 줄바꿈 문자 순으로 표기됩니다. 이름/값 쌍은 바이트, sort 순으로 나열됩니다.

   메시지 유형에 따라 서명할 문자열은 다음의 이름/값 쌍을 보유해야 합니다.

   알림

   알림 메시지는 다음의 이름/값 쌍을 포함해야 합니다.

   ```
   Message
   MessageId
   Subject (if included in the message)
   ```

```
Timestamp
TopicArn
Type
```

다음의 예는 Notification에 대해 서명할 문자열입니다.

```
Message
My Test Message
MessageId
4d4dc071-ddbf-465d-bba8-08f81c89da64
Subject
My subject
Timestamp
2019-01-31T04:37:04.321Z
TopicArn
arn:aws:sns:us-east-2:123456789012:s4-MySNSTopic-1G1WEFCOXTC0P
Type
Notification
```

SubscriptionConfirmation 및 UnsubscribeConfirmation

SubscriptionConfirmation 및 UnsubscribeConfirmation 메시지는 다음의 이름/값 쌍을 포함해야 합니다.

```
Message
MessageId
SubscribeURL
Timestamp
Token
TopicArn
Type
```

다음의 예는 SubscriptionConfirmation에 대해 서명할 문자열입니다.

```
Message
My Test Message
MessageId
3d891288-136d-417f-bc05-901c108273ee
SubscribeURL
```

```
https://sns.us-east-2.amazonaws.com/?
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-east-2:123456789012:s4-
MySNSTopic-1G1WEFCOXTC0P&Token=233...
Timestamp
2019-01-31T19:25:13.719Z
Token
233...
TopicArn
arn:aws:sns:us-east-2:123456789012:s4-MySNSTopic-1G1WEFCOXTC0P
Type
SubscriptionConfirmation
```

6. Base64 형식으로부터의 `Signature` 값을 디코딩합니다. 메시지는 Base64로 인코딩된 `Signature` 값의 서명을 전송합니다. 서명 값을 계산한 서명과 비교하기 전에 Base64 형식으로부터의 `Signature` 값을 디코딩하여 동일한 형식을 사용하여 값을 비교해야 합니다.
7. Amazon SNS 메시지의 파생 해시 값을 생성합니다. Amazon SNS 메시지를 서명을 생성하는데 사용한 동일한 해시 알고리즘에 정규 형식으로 제출합니다.
   a. `SignatureVersion`이 1이면 SHA1을 해시 알고리즘으로 사용합니다.
   b. `SignatureVersion`이 2이면 SHA256을 해시 알고리즘으로 사용합니다.
8. Amazon SNS 메시지의 확인 해시 값을 생성합니다. 확인 해시 값은 공개 키 값(단계 3의)을 사용하여 Amazon SNS 메시지로 전송된 서명 암호를 푼 결과입니다.
9. Amazon SNS 메시지의 신뢰성과 무결성을 확인합니다. 파생 해시 값(단계 7의)과 확인 해시 값(단계 8의)을 비교합니다. 값이 동일하다면 수신자는 메시지가 전송하는 동안 수정되지 않았으며 Amazon SNS에서 기원했음을 확신합니다. 값이 동일하지 않다면 수신자는 이를 신뢰하지 말아야 합니다.

메시지 형식 구문 분석

Amazon SNS는 다음의 형식을 사용합니다.

주제

- HTTP/HTTPS 헤더
- HTTP/HTTPS 구독 확인 JSON 형식
- HTTP/HTTPS 알림 JSON 형식
- HTTP/HTTPS 구독 해지 확인 JSON 형식
- SetSubscriptionAttributes 전송 정책 JSON 형식

- SetTopicAttributes 전송 정책 JSON 형식

HTTP/HTTPS 헤더

Amazon SNS가 구독 확인, 알림 또는 구독 해지 확인 메시지를 HTTP/HTTPS 엔드포인트에 전송할 때 Amazon SNS에 특정한 여러 헤더 값과 함께 POST 메시지를 전송합니다. 이러한 작업의 헤더 값을 사용하면 Type 값을 읽기 위해 JSON 메시지 본문을 구문 분석할 필요 없이 메시지 유형을 파악할 수 있습니다. 기본적으로 Amazon SNS는 `text/plain; charset=UTF-8`로 설정된 `Content-Type`로 모든 알림을 HTTP/S 엔드포인트로 전송합니다. 텍스트/일반(기본값) 외 `Content-Type`을 선택하려면 HTTP/S 전송 정책 생성의 `headerContentType`를 참조하세요.

`x-amz-sns-message-type`

메시지의 유형입니다. 가능한 값은 `SubscriptionConfirmation`, `Notification` 및 `UnsubscribeConfirmation` 입니다.

`x-amz-sns-message-id`

범용 고유 식별자(UUID)로 게시되는 각 메시지마다 고유합니다. 재시도 중에 Amazon SNS가 재전송하는 알림의 경우 원본 메시지의 메시지 ID가 사용됩니다.

`x-amz-sns-topic-arn`

이 메시지가 게시된 주제에 대한 Amazon Resource Name(ARN)입니다.

`x-amz-sns-subscription-arn`

이 엔드포인트에 대한 구독의 ARN입니다.

다음의 HTTP POST 헤더는 HTTP 엔드포인트에 대한 `Notification` 메시지 헤더의 예입니다.

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: 165545c9-2a5c-472c-8df2-7ff2be2b3b1b
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55
Content-Length: 1336
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent
```

HTTP/HTTPS 구독 확인 JSON 형식

사용자가 HTTP/HTTPS 엔드포인트를 구독한 후에 Amazon SNS는 구독 확인 메시지를 HTTP/HTTPS 엔드포인트에 전송합니다. 이 메시지는 구독을 확인하기 위해 방문해야 하는 SubscribeURL 값을 담고 있습니다(또는, Token 값을 ConfirmSubscription과 함께 사용할 수 있습니다).

> **Note**
>
> Amazon SNS는 구독이 확인되기 전에는 알림을 이 엔드포인트로 전송하지 않습니다.

구독 확인 메시지는 다음의 이름/값 쌍을 갖는 JSON 문서를 포함하는 메시지 본문으로 된 POST 메시지입니다.

Type

메시지의 유형입니다. 구독 확인의 경우 유형은 SubscriptionConfirmation입니다.

MessageId

범용 고유 식별자(UUID)로 게시되는 각 메시지마다 고유합니다. 재시도 중에 Amazon SNS가 재전송하는 메시지의 경우 원본 메시지의 메시지 ID가 사용됩니다.

Token

구독 확인을 위해 ConfirmSubscription 작업에 사용할 수 있는 값입니다. 또는, 간단히 SubscribeURL을 방문하면 됩니다.

TopicArn

이 메시지가 구독된 주제에 대한 Amazon Resource Name(ARN)입니다.

Message

메시지를 설명하는 문자열입니다. 구독 확인의 경우 이 문자열은 다음과 같습니다.

```
You have chosen to subscribe to the topic arn:aws:sns:us-
east-2:123456789012:MyTopic.\nTo confirm the subscription, visit the SubscribeURL
 included in this message.
```

SubscribeURL

구독 확인을 위해 방문해야 하는 URL입니다. 또는 Token을 ConfirmSubscription 작업으로 사용하여 구독을 확인하면 됩니다.

Timestamp

구독 확인이 전송된 시간(GMT)입니다.

SignatureVersion

사용한 Amazon SNS 서명의 버전입니다.

- SignatureVersion가 1인 경우, Signature는, Message, MessageId, Type, Timestamp, 및 TopicArn 값의 Base64로 인코딩된 SHA1withRSA 서명입니다.
- SignatureVersion가 2인 경우, Signature는, Message, MessageId, Type, Timestamp, 및 TopicArn 값의 Base64로 인코딩된 SHA256withRSA 서명입니다.

Signature

Message, MessageId, Type, Timestamp, 및 TopicArn 값의 Base64로 인코딩된 SHA1withRSA 또는 SHA256withRSA 서명입니다.

SigningCertURL

메시지에 서명하기 위해 사용된 인증서의 URL입니다.

다음의 HTTP POST 메시지는 HTTP 엔드포인트에 대한 SubscriptionConfirmation 메시지의 예입니다.

```
POST / HTTP/1.1
x-amz-sns-message-type: SubscriptionConfirmation
x-amz-sns-message-id: 165545c9-2a5c-472c-8df2-7ff2be2b3b1b
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
Content-Length: 1336
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "SubscriptionConfirmation",
  "MessageId" : "165545c9-2a5c-472c-8df2-7ff2be2b3b1b",
  "Token" : "2336412f37...",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Message" : "You have chosen to subscribe to the topic arn:aws:sns:us-
west-2:123456789012:MyTopic.\nTo confirm the subscription, visit the SubscribeURL
 included in this message.",
```

```
  "SubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-
west-2:123456789012:MyTopic&Token=2336412f37...",
  "Timestamp" : "2012-04-26T20:45:04.751Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEpH
+DcEwjAPg8O9mY8dReBSwksfg2S7WKQcikcNKWLQjwu6A4VbeS0QHVCkhRS7fUQvi2egU3N858fiTDN6bkkOxYDVrY0Ad8L
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem"
}
```

HTTP/HTTPS 알림 JSON 형식

Amazon SNS가 구독된 HTTP 또는 HTTPS 엔드포인트에 알림을 전송할 때 엔드포인트에 전송된 POST 메시지는 다음의 이름/값 쌍으로 된 JSON 문서를 구성하는 메시지 본문을 보유합니다.

`Type`

메시지의 유형입니다. 알림의 경우 유형은 `Notification`입니다.

`MessageId`

범용 고유 식별자(UUID)로 게시되는 각 메시지마다 고유합니다. 재시도 중에 Amazon SNS가 재전송하는 알림의 경우 원본 메시지의 메시지 ID가 사용됩니다.

`TopicArn`

이 메시지가 게시된 주제에 대한 Amazon Resource Name(ARN)입니다.

`Subject`

알림이 주제에 게시되었을 때 `Subject` 매개 변수입니다.

> **Note**
>
> 이는 선택 가능한 파라미터입니다. `Subject`이 지정되지 않을 경우에는 이 이름-값 쌍은 본 JSON 문서에 표시되지 않습니다.

`Message`

알림이 주제에 게시되었을 때 지정되는 `Message` 값입니다.

Timestamp

알림이 게시된 시간(GMT)입니다.

SignatureVersion

사용한 Amazon SNS 서명의 버전입니다.

- SignatureVersion가 1인 경우, Signature는 Message, MessageId, Subject(있는 경우), Type, Timestamp, TopicArn 값의 Base64로 인코딩된 SHA1withRSA 서명입니다.
- SignatureVersion가 2인 경우, Signature는 Message, MessageId, Subject(있는 경우), Type, Timestamp, TopicArn 값의 Base64로 인코딩된 SHA256withRSA 서명입니다.

Signature

Message, MessageId, Subject(있는 경우), Type, Timestamp, TopicArn 값의 Base64로 인코딩된 SHA1withRSA 또는 SHA256withRSA 서명입니다.

SigningCertURL

메시지에 서명하기 위해 사용된 인증서의 URL입니다.

UnsubscribeURL

이 주제에서 엔드포인트를 구독 해지하는데 사용하는 URL입니다. 이 URL을 방문하면 Amazon SNS는 엔드포인트를 구독 해지하고 이 엔드포인트로 전송하는 알림을 중지합니다.

다음의 HTTP POST 메시지는 HTTP 엔드포인트에 대한 Notification 메시지의 예입니다.

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: 22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:c9135db0-26c4-47ec-8998-413945fb5a96
Content-Length: 773
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent

{
  "Type" : "Notification",
  "MessageId" : "22b80b92-fdea-4c2c-8f9d-bdfb0c7bf324",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
```

```
  "Subject" : "My First Message",
  "Message" : "Hello world!",
  "Timestamp" : "2012-05-02T00:54:06.655Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEw6JRN...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem",
  "UnsubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=Unsubscribe&SubscriptionArn=arn:aws:sns:us-
west-2:123456789012:MyTopic:c9135db0-26c4-47ec-8998-413945fb5a96"
  }
}
```

HTTP/HTTPS 구독 해지 확인 JSON 형식

주제에서 HTTP/HTTPS 엔드포인트가 구독 해지된 후 Amazon SNS는 엔드포인트에 구독 해지 확인 메시지를 전송합니다.

구독 해지 확인 메시지는 다음의 이름/값 쌍을 갖는 JSON 문서를 포함하는 메시지 본문으로 된 POST 메시지입니다.

`Type`

메시지의 유형입니다. 구독 해지 확인의 경우 유형은 `UnsubscribeConfirmation`입니다.

`MessageId`

범용 고유 식별자(UUID)로 게시되는 각 메시지마다 고유합니다. 재시도 중에 Amazon SNS가 재전송하는 메시지의 경우 원본 메시지의 메시지 ID가 사용됩니다.

`Token`

구독 재확인을 위해 `ConfirmSubscription` 작업에 사용할 수 있는 값입니다. 또는, 간단히 `SubscribeURL`을 방문하면 됩니다.

`TopicArn`

이 엔드포인트가 구독 해지된 주제에 대한 Amazon Resource Name(ARN)입니다.

`Message`

메시지를 설명하는 문자열입니다. 구독 해지 확인의 경우 이 문자열은 다음과 같습니다.

```
You have chosen to deactivate subscription arn:aws:sns:us-
east-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55.\nTo cancel this
```

```
operation and restore the subscription, visit the SubscribeURL included in this
message.
```

SubscribeURL

구독 재확인을 위해 방문해야 하는 URL입니다. 또는, 그 대신 Token을 ConfirmSubscription 작업으로 사용하여 구독을 재확인하면 됩니다.

Timestamp

구독 해지 확인이 전송된 시간(GMT)입니다.

SignatureVersion

사용한 Amazon SNS 서명의 버전입니다.

- SignatureVersion가 1인 경우, Signature는, Message, MessageId, Type, Timestamp, 및 TopicArn 값의 Base64로 인코딩된 SHA1withRSA 서명입니다.
- SignatureVersion가 2인 경우, Signature는, Message, MessageId, Type, Timestamp, 및 TopicArn 값의 Base64로 인코딩된 SHA256withRSA 서명입니다.

Signature

Message, MessageId, Type, Timestamp, 및 TopicArn 값의 Base64로 인코딩된 SHA1withRSA 또는 SHA256withRSA 서명입니다.

SigningCertURL

메시지에 서명하기 위해 사용된 인증서의 URL입니다.

다음의 HTTP POST 메시지는 HTTP 엔드포인트에 대한 UnsubscribeConfirmation 메시지의 예입니다.

```
POST / HTTP/1.1
x-amz-sns-message-type: UnsubscribeConfirmation
x-amz-sns-message-id: 47138184-6831-46b8-8f7c-afc488602d7d
x-amz-sns-topic-arn: arn:aws:sns:us-west-2:123456789012:MyTopic
x-amz-sns-subscription-arn: arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55
Content-Length: 1399
Content-Type: text/plain; charset=UTF-8
Host: myhost.example.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent
```

```
{
  "Type" : "UnsubscribeConfirmation",
  "MessageId" : "47138184-6831-46b8-8f7c-afc488602d7d",
  "Token" : "2336412f37...",
  "TopicArn" : "arn:aws:sns:us-west-2:123456789012:MyTopic",
  "Message" : "You have chosen to deactivate subscription arn:aws:sns:us-
west-2:123456789012:MyTopic:2bcfbf39-05c3-41de-beaa-fcfcc21c8f55.\nTo cancel this
 operation and restore the subscription, visit the SubscribeURL included in this
 message.",
  "SubscribeURL" : "https://sns.us-west-2.amazonaws.com/?
Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-
west-2:123456789012:MyTopic&Token=2336412f37fb6...",
  "Timestamp" : "2012-04-26T20:06:41.581Z",
  "SignatureVersion" : "1",
  "Signature" : "EXAMPLEHXgJm...",
  "SigningCertURL" : "https://sns.us-west-2.amazonaws.com/SimpleNotificationService-
f3ecfb7224c7233fe7bb5f59f96de52f.pem"
}
```

SetSubscriptionAttributes 전송 정책 JSON 형식

`SetSubscriptionAttributes` 작업에 요청을 전송하고 `DeliveryPolicy`의 값에 대해 `AttributeName` 매개 변수를 설정할 경우, `AttributeValue` 매개 변수의 값은 유효한 JSON 객체이어야 합니다. 예를 들어, 다음의 예는 최대 5회 재시도하도록 전송 정책을 설정합니다.

```
http://sns.us-east-2.amazonaws.com/
?Action=SetSubscriptionAttributes
&SubscriptionArn=arn%3Aaws%3Asns%3Aus-east-2%3A123456789012%3AMy-Topic
%3A80289ba6-0fd4-4079-afb4-ce8c8260f0ca
&AttributeName=DeliveryPolicy
&AttributeValue={"healthyRetryPolicy":{"numRetries":5}}
...
```

`AttributeValue` 매개 변수의 값에 다음의 JSON 형식을 사용합니다.

```
{
    "healthyRetryPolicy" : {
        "minDelayTarget" :  int,
        "maxDelayTarget" : int,
        "numRetries" : int,
        "numMaxDelayRetries" : int,
        "backoffFunction" : "linear|arithmetic|geometric|exponential"
```

```
    },
    "throttlePolicy" : {
        "maxReceivesPerSecond" : int
    },
    "requestPolicy" : {
        "headerContentType" : "text/plain | application/json | application/xml"
    }
}
```

SetSubscriptionAttribute 작업에 대한 자세한 내용을 확인하려면 Amazon Simple Notification Service API 참조의 SetTopicAttributes로 이동하세요. 지원되는 HTTP 콘텐츠 유형 헤더에 대한 자세한 내용은 HTTP/S 전송 정책 생성을 참조하세요.

SetTopicAttributes 전송 정책 JSON 형식

SetTopicAttributes 작업에 요청을 전송하고 DeliveryPolicy의 값에 대해 AttributeName 매개 변수를 설정할 경우, AttributeValue 매개 변수의 값은 유효한 JSON 객체이어야 합니다. 예를 들어, 다음의 예는 최대 5회 재시도하도록 전송 정책을 설정합니다.

```
http://sns.us-east-2.amazonaws.com/
?Action=SetTopicAttributes
&TopicArn=arn%3Aaws%3Asns%3Aus-east-2%3A123456789012%3AMy-Topic
&AttributeName=DeliveryPolicy
&AttributeValue={"http":{"defaultHealthyRetryPolicy":{"numRetries":5}}}
...
```

AttributeValue 매개 변수의 값에 다음의 JSON 형식을 사용합니다.

```
{
    "http" : {
        "defaultHealthyRetryPolicy" : {
            "minDelayTarget":  int,
            "maxDelayTarget": int,
            "numRetries": int,
            "numMaxDelayRetries": int,
            "backoffFunction": "linear|arithmetic|geometric|exponential"
        },
        "disableSubscriptionOverrides" : Boolean,
        "defaultThrottlePolicy" : {
            "maxReceivesPerSecond" : int
        },
        "defaultRequestPolicy" : {
```

```
            "headerContentType" : "text/plain | application/json | application/xml"
        }
    }
}
```

`SetTopicAttribute` 작업에 대한 자세한 내용을 확인하려면 Amazon Simple Notification Service API 참조의 SetTopicAttributes로 이동하세요. 지원되는 HTTP 콘텐츠 유형 헤더에 대한 자세한 내용은 HTTP/S 전송 정책 생성을 참조하세요.

AWS Event Fork Pipelines로 팬아웃

이제 Amazon SNS는 이벤트 보관 및 분석을 위해 Amazon Data Firehose와의 네이티브 통합 사용을 권장합니다. Firehose 전송 스트림을 SNS 주제로 구독하면 Amazon Simple Storage Service (Amazon S3) 버킷, Amazon Redshift 테이블, OpenSearch Amazon 서비스 (서비스) 등과 같은 보관 및 분석 엔드포인트에 알림을 보낼 수 있습니다. OpenSearch Firehose 전송 스트림과 함께 Amazon SNS를 사용하면 함수를 사용할 필요가 없는 완전 관리형 코드 없는 솔루션입니다. AWS Lambda 자세한 설명은 팬아웃에서 Firehose로의 전송 스트림 섹션을 참조하세요.

Amazon SNS를 사용하여 게시자 서비스에 의해 트리거되는 이벤트에 응답하여 자동으로 작업을 수행하는 구독자 서비스를 사용하는 이벤트 기반 애플리케이션을 빌드할 수 있습니다. 이 아키텍처 패턴은 서비스에서 재사용성, 상호 작용성 및 확장성을 향상할 수 있습니다. 하지만 이벤트 저장, 백업, 검색, 분석, 다시 보기 등 일반적인 이벤트 처리 요구 사항을 해결하는 파이프라인으로 이벤트 처리를 분기하는 것은 공수가 많이 들어갈 수 있습니다.

이벤트 기반 애플리케이션 개발을 가속화하기 위해 AWS Event Fork Pipelines에 의해 제공되는 이벤트 처리 파이프라인에서 Amazon SNS 주제를 구독할 수 있습니다. AWS Event Fork Pipelines는 AWS Serverless Application Model(AWS SAM)을 기반으로 하는 오픈 소스 중첩 애플리케이션 제품군으로, AWS Event Fork Pipelines 제품군(사용자 지정 IAM 역할 또는 리소스 정책을 생성하는 앱 표시 선택)에서 AWS 계정으로 직접 배포할 수 있습니다.

AWS Event Fork Pipelines 사용 사례는 AWS Event Fork Pipelines 샘플 애플리케이션 배포 및 테스트에서 확인하세요.

주제

- AWS Event Fork Pipelines 작동 방식
- AWS Event Fork Pipelines 배포
- AWS Event Fork Pipelines 샘플 애플리케이션 배포 및 테스트

- AWS Event Fork Pipelines에서 Amazon SNS 주제 구독

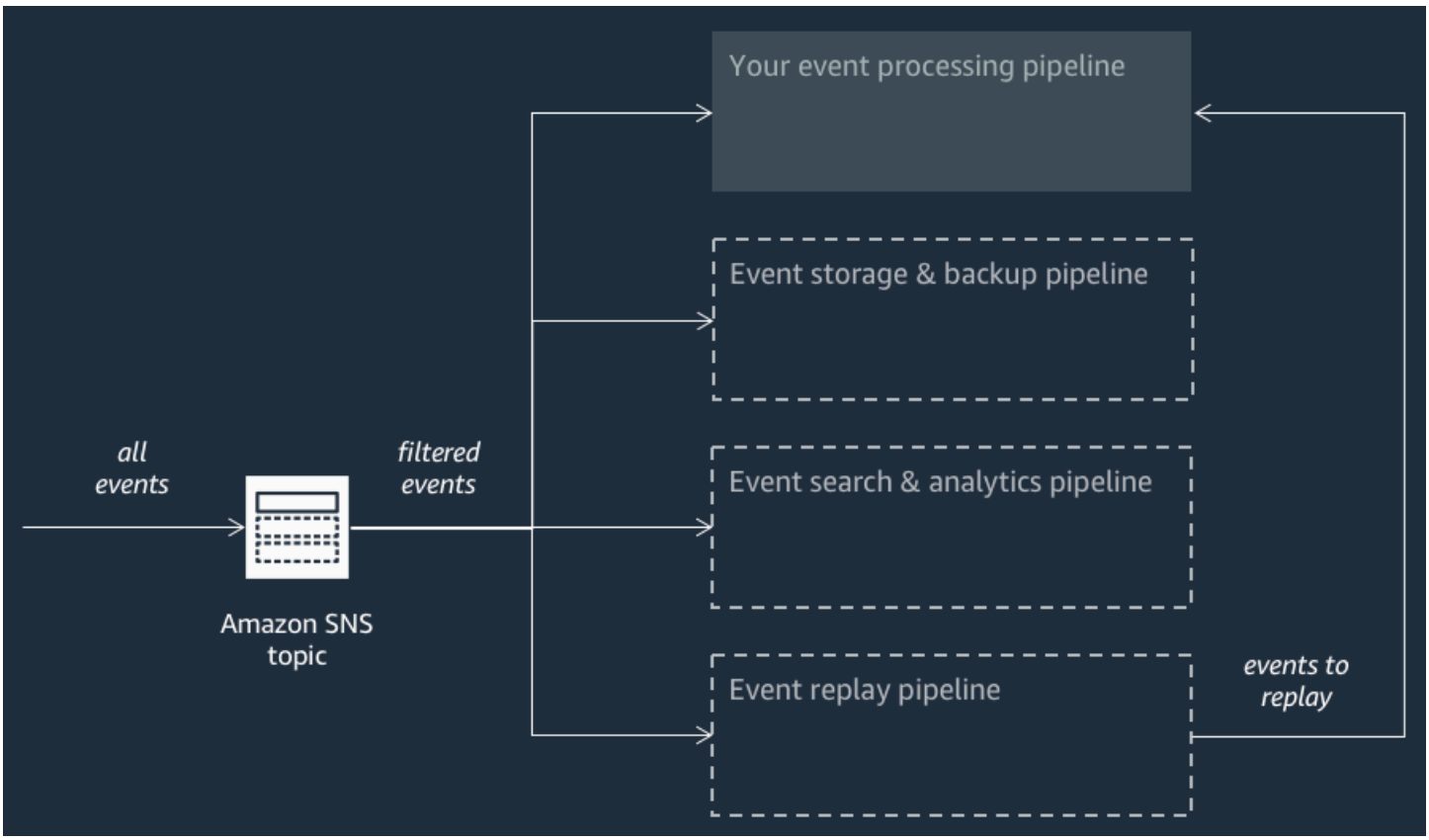
AWS Event Fork Pipelines 작동 방식

AWS Event Fork Pipelines는 서버리스 설계 패턴입니다. 하지만 AWS SAM을 기반으로 하는 중첩 서버리스 애플리케이션의 모음이기도 합니다(이벤트 기반 플랫폼을 보강하기 위해 AWS Serverless Application Repository(AWS SAR)에서 직접 AWS 계정으로 배포할 수 있음). 아키텍처에서 필요할 경우 이러한 중첩 애플리케이션을 개별적으로 배포할 수 있습니다.

주제

- 이벤트 저장 및 백업 파이프라인
- 이벤트 검색 및 분석 파이프라인
- 이벤트 다시 보기 파이프라인

다음 다이어그램은 3개의 중첩 애플리케이션에 의해 보완된 AWS Event Fork Pipelines 애플리케이션입니다. 아키텍처에서 필요할 경우 AWS SAR의 AWS Event Fork Pipelines 제품군에 포함된 어떤 파이프라인도 개별적으로 배포할 수 있습니다.



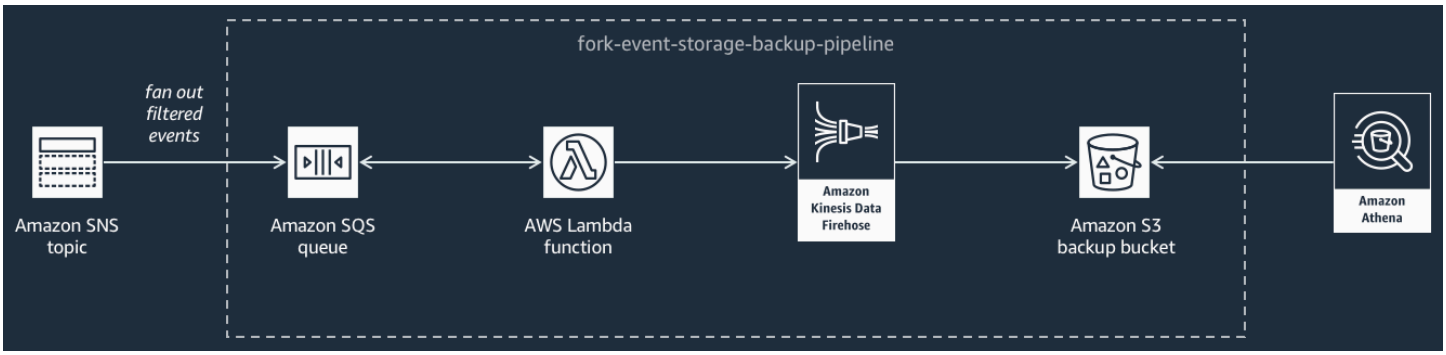
각 파이프라인은 동일한 Amazon SNS 주제를 구독하여 이러한 이벤트가 주제에 게시될 경우 동시에 이벤트를 처리할 수 있습니다. 각 파이프라인은 독립적이며 각각 구독 필터 정책을 설정할 수 있습니다. 그러므로 특정 파이프라인이 주제에 게시되는 모든 이벤트가 아니라 이벤트 하위 집합만 처리할 수 있습니다.

> **Note**
>
> 정규 이벤트 처리 파이프라인(아마도 이미 Amazon SNS 주제를 구독)과 함께 3개의 AWS Event Fork Pipelines를 배치하므로 기존 워크로드에서 AWS Event Fork Pipelines를 활용하기 위해 현재 메시지 게시자의 어떤 부분도 변경할 필요가 없습니다.

이벤트 저장 및 백업 파이프라인

다음 다이어그램은 이벤트 저장 및 백업 파이프라인입니다. 이 파이프라인에서 Amazon SNS 주제를 구독하여 시스템을 통과하는 이벤트를 자동으로 백업할 수 있습니다.

이 파이프라인은 Amazon SNS 주제에 의해 전송된 이벤트를 버퍼링하는 Amazon SQS 대기열, AWS Lambda 대기열에서 이러한 이벤트를 자동으로 폴링하여 Amazon Data Firehose 스트림으로 푸시하는 함수, 스트림에서 로드된 이벤트를 안정적으로 백업하는 Amazon S3 버킷으로 구성됩니다.

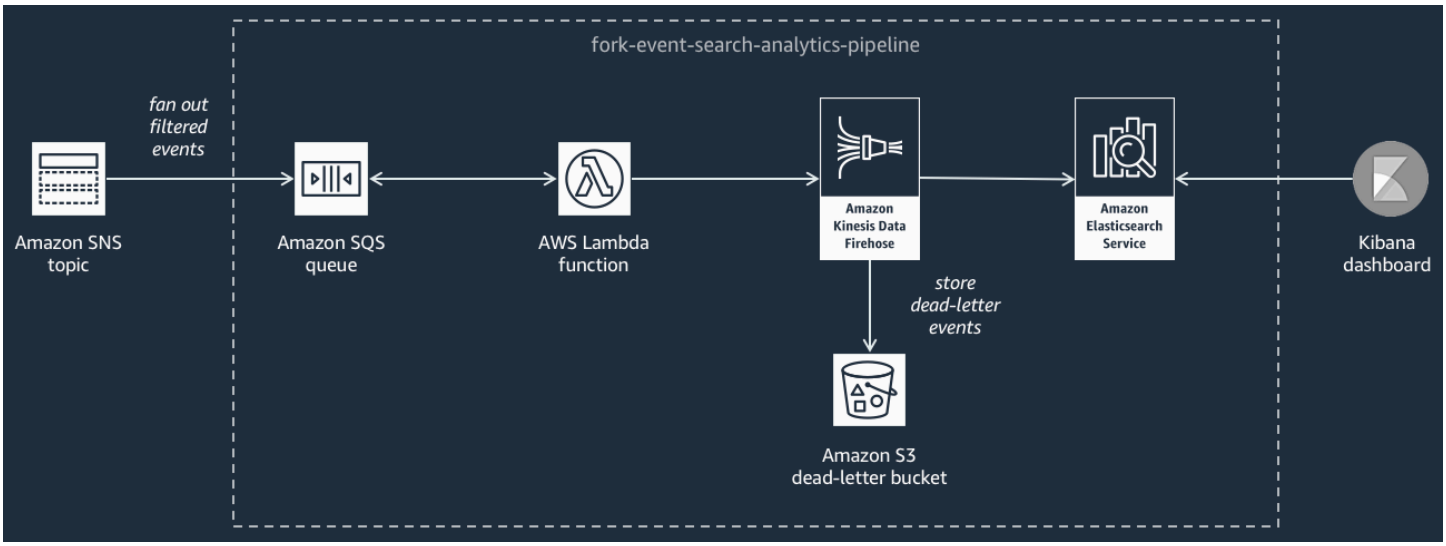


Firehose 스트림의 동작을 미세 조정하려면 버킷으로 로드하기 전에 이벤트를 버퍼링, 변환 및 압축하도록 구성할 수 있습니다. 이벤트가 로드될 때 Amazon Athena를 사용하여 표준 SQL 쿼리로 버킷을 쿼리할 수 있습니다. 또한 기존 Amazon S3 버킷을 재사용하거나 새 버킷을 생성하도록 파이프라인을 구성할 수 있습니다.

이벤트 검색 및 분석 파이프라인

다음 다이어그램은 이벤트 검색 및 분석 파이프라인입니다. 이 파이프라인에서 Amazon SNS 주제를 구독하여 검색 도메인에서 시스템을 통과하는 이벤트를 인덱싱한 후 이들 이벤트에 대해 분석을 실행할 수 있습니다.

이 파이프라인은 Amazon SNS 주제에 의해 전송된 이벤트를 버퍼링하는 Amazon SQS 대기열, AWS Lambda 대기열에서 이벤트를 폴링하여 Amazon Data Firehose 스트림으로 푸시하는 함수, Firehose 스트림에서 로드된 이벤트를 인덱싱하는 Amazon 서비스 도메인, OpenSearch 도메인 검색에서 인덱싱할 수 없는 데드레터 이벤트를 저장하는 Amazon S3 버킷으로 구성됩니다..



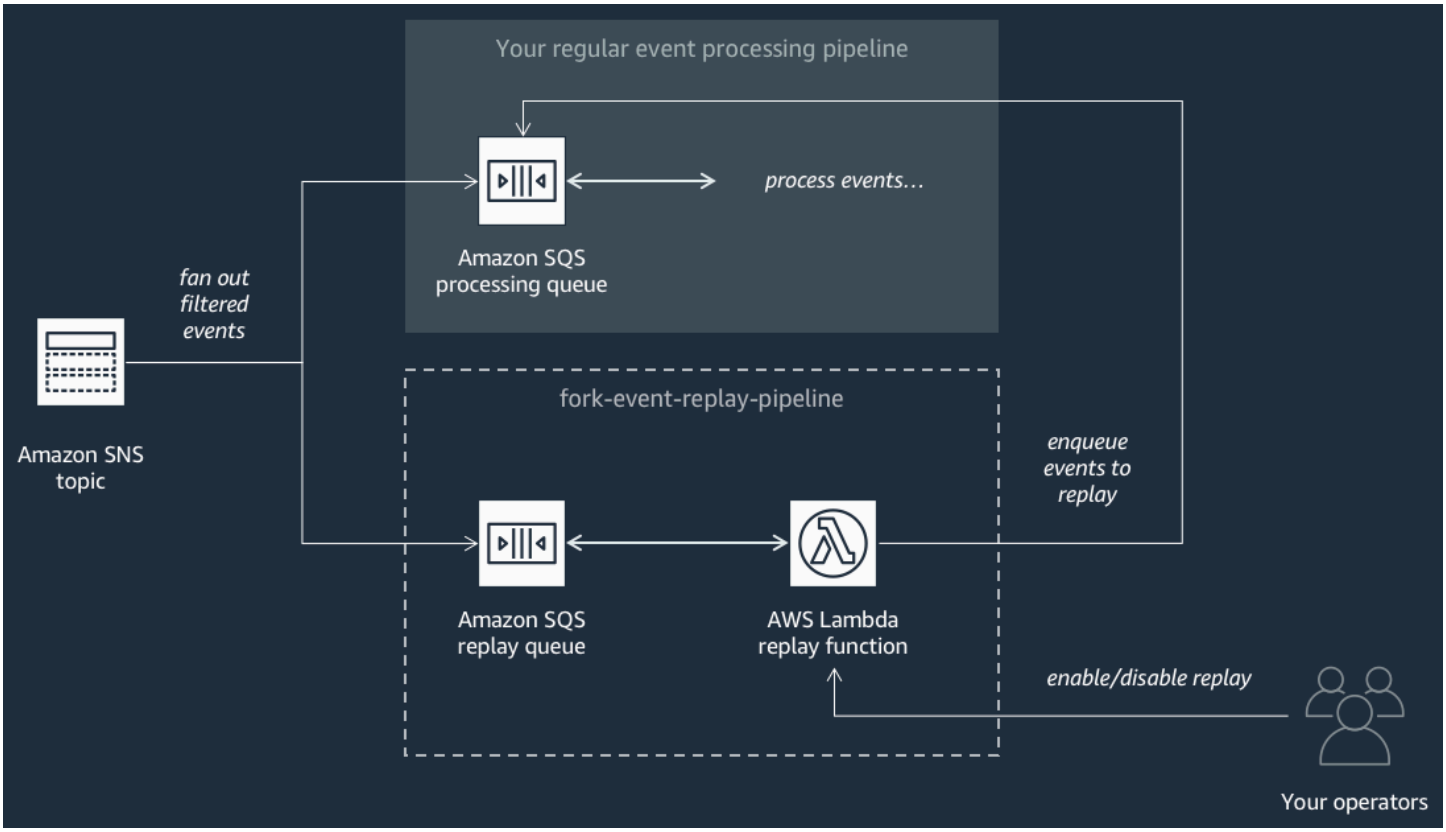
이벤트 버퍼링, 변환 및 압축에서 Firehose 스트림을 미세 조정하기 위해 이 파이프라인을 구성할 수 있습니다.

파이프라인에서 기존 OpenSearch 도메인을 재사용할지 AWS 계정 아니면 새 도메인을 생성할지 구성할 수도 있습니다. 검색 도메인에서 이벤트가 인덱싱될 때 Kibana를 사용하여 이벤트에 대한 분석을 실행하고 시각적 대시보드를 실시간으로 업데이트할 수 있습니다.

이벤트 다시 보기 파이프라인

다음 다이어그램은 이벤트 다시 보기 파이프라인입니다. 시스템에서 지난 14일간 처리한 이벤트를 기록하려면 이 파이프라인에서 Amazon SNS 주제를 구독한 후 이벤트를 다시 처리할 수 있습니다(예를 들어 플랫폼이 장애로부터 복구해야 할 경우).

이 파이프라인은 Amazon SNS 주제로부터 전달된 이벤트를 버퍼링하는 Amazon SQS 대기열과 대기열에서 이벤트를 폴링하여 같은 주제를 구독하는 정규 이벤트 처리 파이프라인으로 리드라이브하는 AWS Lambda 함수로 구성됩니다.



> **Note**
>
> 기본적으로 다시 보기 함수는 비활성화되어 이벤트를 리드라이브하지 않습니다. 이벤트를 다시 처리해야 할 경우 AWS Lambda 다시 보기 함수용 이벤트 소스로 Amazon SQS 다시 보기 대기열을 활성화해야 합니다.

AWS Event Fork Pipelines 배포

AWS Event Fork Pipelines 제품군(S사용자 지정 IAM 역할 또는 리소스 정책을 생성하는 앱 표시를 선택)은 AWS Serverless Application Repository에서 퍼블릭 애플리케이션 그룹으로 사용할 수 있으며, 여기에서 AWS Lambda 콘솔을 사용하여 수동으로 배포 및 테스트할 수 있습니다. AWS Lambda 콘솔을 사용하여 파이프라인을 배포하는 자세한 내용은 AWS Event Fork Pipelines에서 Amazon SNS 주제 구독에서 확인하세요.

프로덕션 시나리오에서는 전체 애플리케이션의 AWS SAM 템플릿에 AWS Event Fork Pipelines를 포함하는 것이 좋습니다. 중첩 애플리케이션 기능을 사용하면 리소스 `AWS::Serverless::Application`을 AWS SAM 템플릿에 추가하고 중첩 애플리케이션의 AWS SAR `ApplicationId` 및 `SemanticVersion`을 참조하여 이렇게 할 수 있습니다.

예를 들어 다음 YAML 코드 조각을 AWS SAM 템플릿의 Resources 섹션에 추가하여 이벤트 저장 및 백업 파이프라인을 중첩 애플리케이션으로 사용할 수 있습니다.

```
Backup:
  Type: AWS::Serverless::Application
  Properties:
    Location:
      ApplicationId: arn:aws:serverlessrepo:us-east-2:123456789012:applications/fork-
event-storage-backup-pipeline
      SemanticVersion: 1.0.0
    Parameters:
      #The ARN of the Amazon SNS topic whose messages should be backed up to the Amazon
 S3 bucket.
      TopicArn: !Ref MySNSTopic
```

파라미터 값을 지정할 때 AWS CloudFormation 내장 함수를 사용하여 템플릿의 다른 리소스를 참조할 수 있습니다. 예를 들어, 위 YAML 코드 조각에서 TopicArn 파라미터는 AWS SAM 템플릿의 다른 위치에서 정의된 AWS::SNS::Topic 리소스 MySNSTopic을 참조합니다. 자세한 정보는 AWS CloudFormation 사용 설명서의 내장 함수 참조를 확인하세요.

> **Note**
>
> AWS SAR 애플리케이션의 AWS Lambda 콘솔 페이지에는 Copy as SAM Resource(SAM 리소스로 복사) 버튼이 포함되어 있습니다. 이 버튼은 AWS SAR 앱을 클립보드로 중첩하는 데 필요한 YAML을 복사합니다.

AWS Event Fork Pipelines 샘플 애플리케이션 배포 및 테스트

이벤트 기반 애플리케이션 개발을 가속화하기 위해 AWS Event Fork Pipelines에 의해 제공되는 이벤트 처리 파이프라인에서 Amazon SNS 주제를 구독할 수 있습니다. AWS Event Fork Pipelines는 AWS Serverless Application Model(AWS SAM)을 기반으로 하는 오픈 소스 중첩 애플리케이션 제품군으로, AWS Event Fork Pipelines 제품군(사용자 지정 IAM 역할 또는 리소스 정책을 생성하는 앱 표시 선택)에서 AWS 계정으로 직접 배포할 수 있습니다. 자세한 설명은 AWS Event Fork Pipelines 작동 방식 섹션을 참조하세요.

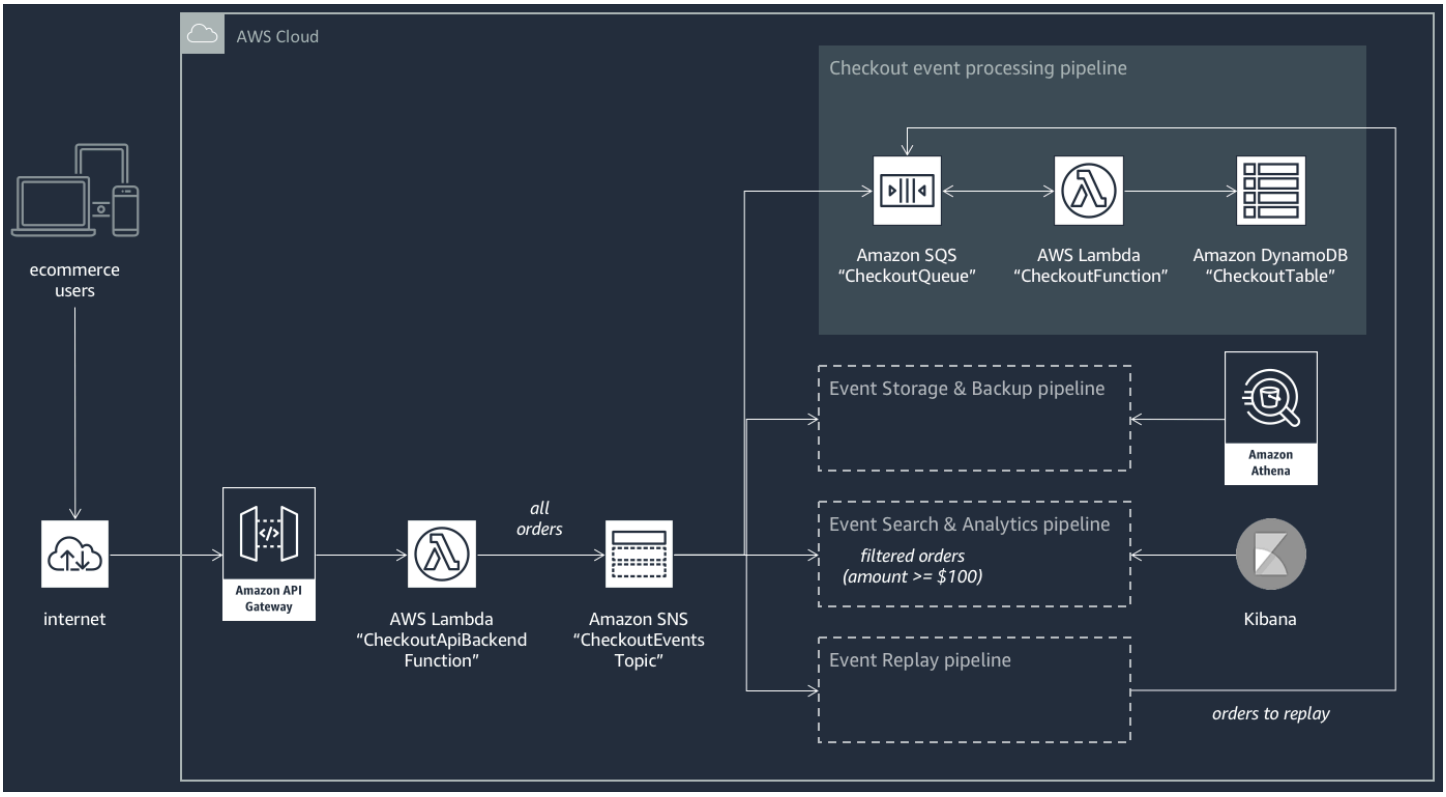
이 페이지에서는 AWS Management Console을 사용하여 AWS Event Fork Pipelines 샘플 애플리케이션을 배포하고 테스트하는 방법을 보여줍니다.

> **Important**
>
> AWS Event Fork Pipelines 샘플 애플리케이션을 배포한 후 불필요한 비용이 발생하는 것을 방지하려면 해당 AWS CloudFormation 스택을 삭제하세요. 자세한 정보는 AWS CloudFormation 사용 설명서의 AWS CloudFormation 콘솔에서 스택 삭제를 참조하세요.

주제

예제 AWS Event Fork Pipelines 사용 사례

다음 시나리오에서는 AWS Event Fork Pipelines를 사용하는 이벤트 기반 서버리스 전자 상거래 애플리케이션을 설명합니다. 이 예제 전자 상거래 애플리케이션을 에서 사용한 다음 콘솔을 AWS 계정 사용하여 배포할 수 있습니다. AWS Lambda 콘솔에서 테스트하고 소스 코드를 검토할 수 GitHub 있습니다. AWS Serverless Application Repository



이 전자 상거래 애플리케이션은 API Gateway에 의해 호스팅되고 AWS Lambda 함수 `CheckoutApiBackendFunction`에 의해 백업되는 RESTful API를 통해 구매자로부터 주문을 수신합니다. 이 함수는 모든 수신된 주문을 `CheckoutEventsTopic`이라는 Amazon SNS 주제 에 게시하고, 이 주제는 주문을 4가지 파이프라인으로 분산합니다.

첫 번째 파이프라인은 전자 상거래 애플리케이션의 소유자가 설계 및 구현한 정규 체크아웃 처리 파이프라인입니다. 이 파이프라인에는 모든 수신된 주문을 버퍼링하는 Amazon SQS 대기열 `CheckoutQueue`, 대기열을 폴링하여 이들 주문을 처리하는 `CheckoutFunction`이라는 이름의 AWS Lambda 함수 , 모든 접수된 주문을 안전하게 저장하는 DynamoDB 테이블 `CheckoutTable`이 있습니다.

AWS Event Fork Pipelines 적용

전자 상거래 애플리케이션의 구성 요소가 핵심 비즈니스 로직을 처리합니다. 그러나 전자 상거래 애플리케이션 소유자도 다음을 처리해야 합니다.

- 규정 준수—안전한 압축 백업(저장 시 암호화) 및 민감한 정보의 폐기
- 복원성—이행 프로세스 중단 시 최근 주문 다시 보기
- 검색 가능성—접수된 주문에 대한 분석 실행 및 지표 생성

이 이벤트 처리 로직을 구현하는 대신, 애플리케이션 소유자는 AWS Event Fork Pipelines에서 `CheckoutEventsTopic` Amazon SNS 주제를 구독할 수 있습니다.

- 이벤트 저장 및 백업 파이프라인은 데이터를 변환하여 신용카드 세부 정보를 제거하고, 데이터를 60초간 버퍼링하고, GZIP을 사용하여 데이터를 압축하고, Amazon S3용 기본 고객 관리형 키를 사용하여 암호화하도록 구성되어 있습니다. 이 키는 AWS에 의해 관리되고 AWS Key Management Service(AWS KMS)에 의해 구동됩니다.

  자세한 내용은 Amazon Data Firehose 개발자 안내서의 Amazon S3 선택, Amazon Data Firehose 데이터 변환 및 설정 구성을 참조하십시오.
- 이벤트 검색 및 분석 파이프라인은(는) 인덱스 재시도 지속시간 30초, 검색 도메인에서 인덱싱되지 않은 주문을 저장할 버킷, 인덱싱할 주문을 제한하는 필터 정책으로 구성되어 있습니다.

  자세한 내용은 Amazon Data Firehose 개발자 안내서의 대상 OpenSearch 서비스 선택을 참조하십시오.
- 이벤트 다시 보기 파이프라인은(는) 전자 상거래 애플리케이션 소유자가 설계 및 구현한 일반 체크아웃 처리 파이프라인의 Amazon SQS 대기열 부분으로 구성되어 있습니다.

  자세한 정보는 Amazon Simple Queue Service 개발자 안내서의 대기열 이름 및 URL을 참조하세요.

다음 JSON filter 필터 정책은 이벤트 검색 및 분석 파이프라인의 구성에 설정되었습니다. 수신 주문 중 전체 금액이 $100 이상인 주문만 일치됩니다. 자세한 설명은 Amazon SNS 메시지 필터링 섹션을 참조하세요.

```
{
   "amount": [{ "numeric": [ ">=", 100 ] }]
}
```

전자 상거래 애플리케이션 소유자는 AWS Event Fork Pipelines 패턴을 사용하여 흔히 이벤트 처리를 위한 일반적인 로직을 코딩하는 데 따르는 개발 오버헤드를 방지할 수 있습니다. 대신, AWS Event Fork Pipelines를 AWS Serverless Application Repository에서 AWS 계정로 직접 배포할 수 있습니다.

1단계: 샘플 애플리케이션 배포

1. AWS Lambda 콘솔에 로그인합니다.
2. 탐색 창에서 함수를 선택한 후 함수 생성을 선택합니다.
3. 함수 생성 페이지에서 다음을 수행합니다.

a. 서버리스 앱 리포지토리 찾아보기, 퍼블릭 애플리케이션, 사용자 지정 IAM 역할 또는 리소스 정책을 생성하는 앱 표시를 선택합니다.

b. `fork-example-ecommerce-checkout-api`을 검색하여 이 애플리케이션을 선택합니다.

4. fork-example-ecommerce-checkout-api 페이지에서 다음을 수행하십시오.

a. Application settings(애플리케이션 설정) 섹션에서 애플리케이션 이름을 입력합니다(예: `fork-example-ecommerce-my-app`).

> **Note**
>
> - 나중에 손쉽게 리소스를 찾을 수 있도록 접두사 `fork-example-ecommerce`를 유지하세요.
> - 각 배포에서 애플리케이션 이름이 고유해야 합니다. 애플리케이션 이름을 재사용할 경우 배포가 이전에 배포된 AWS CloudFormation 스택만 업데이트합니다(새 스택을 생성하지 않음).

b. (선택 사항) 애플리케이션의 Lambda 함수 실행을 위한 다음 LogLevel설정 중 하나를 입력합니다.

- `DEBUG`
- `ERROR`
- `INFO`(기본값)
- `WARNING`

5. I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications.(본인은 이 앱이 사용자 지정 IAM 역할, 리소스 정책을 생성하고 중첩 애플리케이션을 배포함을 확인합니다.)를 선택하고 페이지 맨 아래에서 배포를 선택합니다.

fork-example-ecommerce-***my-app*** 배포 상태 페이지에서 Lambda는 애플리케이션 배포 중 상태를 표시합니다.

리소스 섹션에서 AWS CloudFormation이 스택을 생성하기 시작하고 각 리소스에 대해 CREATE_IN_PROGRESS 상태를 표시합니다. 이 과정이 완료되면 AWS CloudFormation에는 CREATE_COMPLETE 상태가 표시됩니다.

> **Note**
>
> 모든 리소스를 배포하는 데 20~30분 정도 걸릴 수 있습니다.

배포가 완료되면 Lambda가 Your application has been deployed(애플리케이션이 배포됨) 상태를 표시합니다.

2단계: 샘플 애플리케이션 실행

1. AWS Lambda 콘솔의 탐색 창에서 애플리케이션을 선택합니다.
2. 애플리케이션 페이지의 검색란에서 `serverlessrepo-fork-example-ecommerce-`*`my-app`*을 검색하여 이 애플리케이션을 선택합니다.
3. 리소스 섹션에서 다음을 수행합니다.
   a. 유형이 다음과 ApiGatewayRestApi같은 리소스를 찾으려면 예를 `ServerlessRestApi` 들어 유형별로 리소스를 정렬한 다음 리소스를 확장하십시오.
   b. ApiGateway디플로이먼트와 ApiGateway 스테이지 유형의 두 개의 중첩된 리소스가 표시됩니다.
   c. Prod API endpoint(Prod API 엔드포인트) 링크를 복사하고 여기에 `/checkout`를 추가합니다. 예:

   ```
   https://abcdefghij.execute-api.us-east-2.amazonaws.com/Prod/checkout
   ```

4. 다음 JSON을 `test_event.json` 파일에 복사합니다.

```
{
   "id": 15311,
   "date": "2019-03-25T23:41:11-08:00",
   "status": "confirmed",
   "customer": {
      "id": 65144,
      "name": "John Doe",
      "email": "john.doe@example.com"
   },
   "payment": {
      "id": 2509,
      "amount": 450.00,
      "currency": "usd",
```

```
      "method": "credit",
      "card-network": "visa",
      "card-number": "1234 5678 9012 3456",
      "card-expiry": "10/2022",
      "card-owner": "John Doe",
      "card-cvv": "123"
   },
   "shipping": {
      "id": 7600,
      "time": 2,
      "unit": "days",
      "method": "courier"
   },
   "items": [{
      "id": 6512,
      "product": 8711,
      "name": "Hockey Jersey - Large",
      "quantity": 1,
      "price": 400.00,
      "subtotal": 400.00
   }, {
      "id": 9954,
      "product": 7600,
      "name": "Hockey Puck",
      "quantity": 2,
      "price": 25.00,
      "subtotal": 50.00
   }]
}
```

5. HTTPS 요청을 API 엔드포인트로 전송하려면 `curl` 명령을 실행하여 샘플 이벤트 페이로드를 입력으로 전달합니다. 예:

```
curl -d "$(cat test_event.json)" https://abcdefghij.execute-api.us-
east-2.amazonaws.com/Prod/checkout
```

API가 다음과 같이 실행 성공을 나타내는 빈 응답을 반환합니다.

```
{ }
```

3단계: 샘플 애플리케이션 및 파이프라인의 실행을 확인

1단계: 샘플 체크아웃 애플리케이션의 실행을 확인

1. Amazon DynamoDB 콘솔에 로그인합니다.
2. 탐색 창에서 테이블을 선택합니다.
3. `serverlessrepo-fork-example`을 검색하고 `CheckoutTable`을 선택합니다.
4. 테이블 세부 정보에서 항목을 선택한 다음 생성된 항목을 선택합니다.

   저장된 속성이 표시됩니다.

2단계: 이벤트 저장 및 백업 파이프라인의 실행을 확인

1. Amazon S3 콘솔에 로그인합니다.
2. 탐색 창에서 버킷을 선택합니다.
3. `serverlessrepo-fork-example`을 검색하고 `CheckoutBucket`을 선택합니다.
4. 확장명이 `.gz`인 파일을 찾을 때까지 디렉터리 계층을 탐색합니다.
5. 파일을 다운로드하려면 작업, 열기를 선택합니다.
6. 이 파이프라인은 규정 준수를 위해 신용카드 정보를 폐기하는 Lambda 함수로 구성되어 있습니다.

   저장된 JSON 페이로드가 신용카드 정보를 포함하지 않는지 확인하려면 파일을 압축합니다.

3단계: 이벤트 검색 및 분석 파이프라인의 실행을 확인

1. OpenSearch 서비스 콘솔에 로그인합니다.
2. 탐색 창의 My domains(내 도메인)에서 접두사가 `serverl-analyt`인 도메인을 선택합니다.
3. 이 파이프라인은 숫자 일치 조건을 설정하는 Amazon SNS 구독 필터 정책으로 구성되어 있습니다.

   이벤트가 금액이 USD $100를 초과하는 주문을 참조하기 때문에 인덱싱되는지 확인하려면, serverl-analyt-***abcdefgh1ijk*** 페이지에서 Indices(인덱스), checkout_events(체크아웃 이벤트)를 선택합니다.

4단계: 이벤트 다시 보기 애플리케이션의 실행을 확인

1. Amazon SQS 콘솔에 로그인합니다.

2. 대기열 목록에서 `serverlessrepo-fork-example`을 검색하고 `ReplayQueue`를 선택합니다.
3. [메시지 전송 및 수신(Send and receive messages)]을 선택합니다.
4. fork-example-ecommerce- ***my-app***... ReplayP- ReplayQueue - ***123ABCD4E5F6*** 에서 메시지 보내기 및 받기 대화 상자에서 메시지 설문 조사를 선택합니다.
5. 이벤트가 대기 상태로 전환되는지 확인하려면 대기열에 표시되는 메시지 옆의 추가 정보를 선택합니다.

4단계: 문제를 시뮬레이션하고 복구를 위해 이벤트를 다시 보기

1단계: 시뮬레이션된 문제를 활성화하고 두 번째 API 요청을 전송

1. AWS Lambda 콘솔에 로그인합니다.
2. 탐색 창에서 함수를 선택합니다.
3. `serverlessrepo-fork-example`을 검색하고 `CheckoutFunction`을 선택합니다.
4. fork-example-ecommerce- ***##*** 앱 - - ABCDEF에서*... CheckoutFunction* 페이지의 환경 변수 섹션에서 BUG_ENABLED 변수를 true로 설정한 다음 저장을 선택합니다.
5. 다음 JSON을 `test_event_2.json` 파일에 복사합니다.

```
{
    "id": 9917,
    "date": "2019-03-26T21:11:10-08:00",
    "status": "confirmed",
    "customer": {
       "id": 56999,
       "name": "Marcia Oliveira",
       "email": "marcia.oliveira@example.com"
    },
    "payment": {
       "id": 3311,
       "amount": 75.00,
       "currency": "usd",
       "method": "credit",
       "card-network": "mastercard",
       "card-number": "1234 5678 9012 3456",
       "card-expiry": "12/2025",
       "card-owner": "Marcia Oliveira",
       "card-cvv": "321"
    },
```

```
    "shipping": {
       "id": 9900,
       "time": 20,
       "unit": "days",
       "method": "plane"
    },
    "items": [{
       "id": 9993,
       "product": 3120,
       "name": "Hockey Stick",
       "quantity": 1,
       "price": 75.00,
       "subtotal": 75.00
    }]
 }
```

6. HTTPS 요청을 API 엔드포인트로 전송하려면 `curl` 명령을 실행하여 샘플 이벤트 페이로드를 입력으로 전달합니다. 예:

   ```
   curl -d "$(cat test_event_2.json)" https://abcdefghij.execute-api.us-
   east-2.amazonaws.com/Prod/checkout
   ```

   API가 다음과 같이 실행 성공을 나타내는 빈 응답을 반환합니다.

   ```
   { }
   ```

2단계: 시뮬레이션된 데이터 손상을 확인

1. Amazon DynamoDB 콘솔에 로그인합니다.
2. 탐색 창에서 테이블을 선택합니다.
3. `serverlessrepo-fork-example`을 검색하고 `CheckoutTable`을 선택합니다.
4. 테이블 세부 정보에서 항목을 선택한 다음 생성된 항목을 선택합니다.

   저장된 속성이 표시되고 일부가 CORRUPTED!(손상됨!)으로 표시됩니다.

3단계: 시뮬레이션된 문제를 비활성화

1. AWS Lambda 콘솔에 로그인합니다.
2. 탐색 창에서 함수를 선택합니다.

3. `serverlessrepo-fork-example`을 검색하고 `CheckoutFunction`을 선택합니다.
4. fork-example-ecommerce-***## # - - ABCDEF##...*** *`CheckoutFunction`* 페이지의 환경 변수 섹션에서 BUG_ENABLED 변수를 false로 설정한 다음 저장을 선택합니다.

4단계: 문제로부터 복구하기 위해 다시 보기를 활성화

1. AWS Lambda 콘솔의 탐색 창에서 함수를 선택합니다.
2. `serverlessrepo-fork-example`을 검색하고 `ReplayFunction`을 선택합니다.
3. Designer(디자이너) 섹션을 확장하고, SQS 타일을 선택한 다음, SQS 섹션에서 활성을 선택합니다.

> **Note**
>
> Amazon SQS 이벤트 소스 트리거가 활성화되려면 약 1분 정도 걸립니다.

4. 저장을 선택합니다.
5. 복구된 속성을 보려면 Amazon DynamoDB 콘솔로 돌아갑니다.
6. 다시 보기를 비활성화하려면 AWS Lambda 콘솔로 돌아가 `ReplayFunction`에 대한 Amazon SQS 이벤트 소스 트리거를 비활성화합니다.

AWS Event Fork Pipelines에서 Amazon SNS 주제 구독

이벤트 기반 애플리케이션 개발을 가속화하기 위해 AWS Event Fork Pipelines에 의해 제공되는 이벤트 처리 파이프라인에서 Amazon SNS 주제를 구독할 수 있습니다. AWS Event Fork Pipelines는 AWS Serverless Application Model(AWS SAM)을 기반으로 하는 오픈 소스 중첩 애플리케이션 제품군으로, AWS Event Fork Pipelines 제품군(사용자 지정 IAM 역할 또는 리소스 정책을 생성하는 앱 표시 선택)에서 AWS 계정으로 직접 배포할 수 있습니다. 자세한 설명은 AWS Event Fork Pipelines 작동 방식 섹션을 참조하세요.

이 섹션에서는 AWS Management Console을 사용하여 파이프라인을 배포한 다음 AWS Event Fork Pipelines에서 Amazon SNS 주제를 구독하는 방법을 보여줍니다. 시작하기 전에 Amazon SNS 주제를 생성합니다.

파이프라인을 구성하는 리소스를 삭제하려면 AWS Lambda 콘솔의 애플리케이션 페이지에서 파이프라인을 찾아 SAM 템플릿 섹션을 확장하고 스택을 선택한 다음 기타 작업, CloudFormation스택 삭제를 선택합니다.

주제

이벤트 저장 및 백업 파이프라인을 배포하고 구독 설정하려면

이제 Amazon SNS는 이벤트 보관 및 분석을 위해 Amazon Data Firehose와의 네이티브 통합 사용을 권장합니다. Firehose 전송 스트림을 SNS 주제로 구독하면 Amazon Simple Storage Service (Amazon S3) 버킷, Amazon Redshift 테이블, OpenSearch Amazon 서비스 (서비스) 등과 같은 보관 및 분석 엔드포인트에 알림을 보낼 수 있습니다. OpenSearch Firehose 전송 스트림과 함께 Amazon SNS를 사용하면 함수를 사용할 필요가 없는 완전 관리형 코드 없는 솔루션입니다. AWS Lambda 자세한 설명은 팬아웃에서 Firehose로의 전송 스트림 섹션을 참조하세요.

이 페이지에서는 이벤트 저장 및 백업 파이프라인을 배포하여 Amazon SNS 주제를 구독하는 방법을 보여줍니다. 이 프로세스는 자동으로 파이프라인과 연결된 AWS SAM 템플릿을 AWS CloudFormation 스택으로 변환한 후 이 스택을 AWS 계정로 배포합니다. 또한 이 프로세스는 이벤트 저장 및 백업 파이프라인을 구성하는 다음과 같은 리소스 세트를 생성하고 구성합니다.

- Amazon SQS 대기열
- Lambda 함수
- Firehose 전송 스트림
- Amazon S3 백업 버킷

S3 버킷을 대상으로 스트림을 구성하는 방법에 대한 자세한 내용은 Amazon Data Firehose API 참조를 참조하십시오`S3DestinationConfiguration`.

이벤트 변환과 이벤트 버퍼링, 이벤트 압축 및 이벤트 암호화를 구성하는 방법에 대한 자세한 내용은 Amazon Data Firehose 개발자 안내서의 Amazon Data Firehose 전송 스트림 생성을 참조하십시오.

이벤트 필터링에 대한 자세한 정보는 이 설명서의 Amazon SNS 구독 필터 정책에서 확인하세요.

1. AWS Lambda 콘솔에 로그인합니다.
2. 탐색 창에서 함수를 선택한 후 함수 생성을 선택합니다.
3. 함수 생성 페이지에서 다음을 수행합니다.

a. 서버리스 앱 리포지토리 찾아보기, 퍼블릭 애플리케이션, 사용자 지정 IAM 역할 또는 리소스 정책을 생성하는 앱 표시를 선택합니다.

b. `fork-event-storage-backup-pipeline`을 검색하여 이 애플리케이션을 선택합니다.

4. fork-event-storage-backup-pipeline 페이지에서 다음을 수행하십시오.

a. Application settings(애플리케이션 설정) 섹션에서 애플리케이션 이름을 입력합니다(예: `my-app-backup`).

> **Note**
>
> - 각 배포에서 애플리케이션 이름이 고유해야 합니다. 애플리케이션 이름을 재사용할 경우 배포가 이전에 배포된 AWS CloudFormation 스택만 업데이트합니다(새 스택을 생성하지 않음).

b. (선택 사항) 에 BucketArn수신 이벤트가 로드되는 S3 버킷의 ARN을 입력합니다. 값을 입력하지 않을 경우 AWS 계정에 새로운 S3 버킷이 생성됩니다.

c. (선택 사항) 에 DataTransformationFunctionArn수신 이벤트를 변환하는 데 사용되는 Lambda 함수의 ARN을 입력합니다. 값을 입력하지 않을 경우 데이터 변환이 비활성화됩니다.

d. (선택 사항) 애플리케이션의 Lambda 함수 실행을 위한 다음 LogLevel설정 중 하나를 입력합니다.

- `DEBUG`
- `ERROR`
- `INFO`(기본값)
- `WARNING`

e. 의 TopicArn경우 이 포크 파이프라인 인스턴스를 구독할 Amazon SNS 주제의 ARN을 입력합니다.

f. (선택 사항) StreamBufferingIntervalInSecondsFor 및 StreamBufferingSizeInMB에 수신 이벤트의 버퍼링을 구성하기 위한 값을 입력합니다. 값을 입력하지 않을 경우 300초 및 5MB가 사용됩니다.

g. (선택 사항) 수신 이벤트를 압축하기 위한 다음 StreamCompressionFormat설정 중 하나를 입력합니다.

- `GZIP`
- `SNAPPY`

- UNCOMPRESSED(기본값)
- ZIP

h. (선택 사항) 의 경우 S3 백업 버킷에 저장된 파일의 이름을 지정하기 위한 StreamPrefix문자열 접두사를 입력합니다. 값을 입력하지 않을 경우 접두사가 사용되지 않습니다.

i. (선택 사항 SubscriptionFilterPolicy) 수신 이벤트를 필터링하는 데 사용할 Amazon SNS 구독 필터 정책을 JSON 형식으로 입력합니다. 필터 정책은 OpenSearch 서비스 인덱스에 인덱싱되는 이벤트를 결정합니다. 값을 입력하지 않을 경우 필터링이 사용되지 않습니다(모든 이벤트가 인덱싱됨).

j. (선택 사항) 에 SubscriptionFilterPolicyScope문자열을 입력하거나 페이로드 기반 `MessageBody` 또는 `MessageAttributes` 속성 기반 메시지 필터링을 활성화합니다.

k. I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications.(본인은 이 앱이 사용자 지정 IAM 역할, 리소스 정책을 생성하고 중첩 애플리케이션을 배포함을 확인합니다.)를 선택하고 배포를 선택합니다.

Deployment status for ***my-app***(my-app 배포 상태) 페이지에서 Lambda가 Your application is being deployed(애플리케이션을 배포 중) 상태를 표시합니다.

리소스 섹션에서 AWS CloudFormation이 스택을 생성하기 시작하고 각 리소스에 대해 CREATE_IN_PROGRESS 상태를 표시합니다. 이 과정이 완료되면 AWS CloudFormation에는 CREATE_COMPLETE 상태가 표시됩니다.

배포가 완료되면 Lambda가 Your application has been deployed(애플리케이션이 배포됨) 상태를 표시합니다.

Amazon SNS 주제에 게시되는 메시지는 이벤트 저장 및 백업 파이프라인에 의해 프로비저닝되는 S3 백업 버킷에 자동으로 저장됩니다.

이벤트 검색 및 분석 파이프라인을 배포하고 구독 설정하려면

이제 Amazon SNS는 이벤트 보관 및 분석을 위해 Amazon Data Firehose와의 네이티브 통합 사용을 권장합니다. Firehose 전송 스트림을 SNS 주제로 구독하면 Amazon Simple Storage Service (Amazon S3) 버킷, Amazon Redshift 테이블, OpenSearch Amazon 서비스 (서비스) 등과 같은 보관 및 분석 엔드포인트에 알림을 보낼 수 있습니다. OpenSearch Firehose 전송 스트림과 함께 Amazon SNS를 사용하면 함수를 사용할 필요가 없는 완전 관리형 코드 없는 솔루션입니다. AWS Lambda 자세한 설명은 팬아웃에서 Firehose로의 전송 스트림 섹션을 참조하세요.

이 페이지에서는 이벤트 검색 및 분석 파이프라인을 배포하여 Amazon SNS 주제를 구독 설정하는 방법을 보여줍니다. 이 프로세스는 자동으로 파이프라인과 연결된 AWS SAM 템플릿을 AWS CloudFormation 스택으로 변환한 후 이 스택을 AWS 계정로 배포합니다. 또한 이 프로세스는 이벤트 검색 및 분석 파이프라인을 구성하는 다음과 같은 리소스 세트를 생성하고 구성합니다.

- Amazon SQS 대기열
- Lambda 함수
- Firehose 전송 스트림
- 아마존 OpenSearch 서비스 도메인
- Amazon S3 배달 못한 편지 버킷

인덱스를 대상으로 사용하여 스트림을 구성하는 방법에 대한 자세한 내용은 Amazon Data Firehose API 참조를 참조하십시오`ElasticsearchDestinationConfiguration`.

이벤트 변환과 이벤트 버퍼링, 이벤트 압축 및 이벤트 암호화를 구성하는 방법에 대한 자세한 내용은 Amazon Data Firehose 개발자 안내서의 Amazon Data Firehose 전송 스트림 생성을 참조하십시오.

이벤트 필터링에 대한 자세한 정보는 이 설명서의 Amazon SNS 구독 필터 정책에서 확인하세요.

1. AWS Lambda 콘솔에 로그인합니다.
2. 탐색 창에서 함수를 선택한 후 함수 생성을 선택합니다.
3. 함수 생성 페이지에서 다음을 수행합니다.
   a. 서버리스 앱 리포지토리 찾아보기, 퍼블릭 애플리케이션, 사용자 지정 IAM 역할 또는 리소스 정책을 생성하는 앱 표시를 선택합니다.
   b. `fork-event-search-analytics-pipeline`을 검색하여 이 애플리케이션을 선택합니다.
4. fork-event-search-analytics-pipeline 페이지에서 다음을 수행하십시오.
   a. Application settings(애플리케이션 설정) 섹션에서 애플리케이션 이름을 입력합니다(예: `my-app-search`).

> **Note**
>
> 각 배포에서 애플리케이션 이름이 고유해야 합니다. 애플리케이션 이름을 재사용할 경우 배포가 이전에 배포된 AWS CloudFormation 스택만 업데이트합니다(새 스택을 생성하지 않음).

b. (선택 사항) 수신 DataTransformationFunctionArn이벤트를 변환하는 데 사용되는 Lambda 함수의 ARN을 입력합니다. 값을 입력하지 않을 경우 데이터 변환이 비활성화됩니다.

c. (선택 사항) 애플리케이션의 Lambda 함수 실행을 위한 다음 LogLevel설정 중 하나를 입력합니다.

- `DEBUG`
- `ERROR`
- `INFO`(기본값)
- `WARNING`

d. (선택 사항 SearchDomainArn) 필요한 컴퓨팅 및 스토리지 기능을 구성하는 클러스터인 OpenSearch 서비스 도메인의 ARN을 입력합니다. 값을 입력하지 않을 경우 기본 구성으로 새 도메인이 생성됩니다.

e. 의 TopicArn경우 이 포크 파이프라인 인스턴스를 구독할 Amazon SNS 주제의 ARN을 입력합니다.

f. 에 SearchIndexName이벤트 검색 및 분석을 위한 OpenSearch 서비스 인덱스의 이름을 입력합니다.

> **Note**
>
> 다음 할당량이 인덱스 이름에 적용됩니다.
>
> - 대문자를 포함할 수 없음
> - 다음 문자를 포함할 수 없음: `\ / * ? " < > | ` , #`
> - 다음 문자로 시작할 수 없음: `- + _`
> - 다음과 같을 수 없음: `. ..`
> - 80자를 초과할 수 없음
> - 255바이트를 초과할 수 없음
> - 콜론을 포함할 수 없음 (OpenSearch 서비스 7.0부터)

g. (선택 사항) OpenSearch 서비스 인덱스의 순환 기간에 대한 다음 SearchIndexRotationPeriod설정 중 하나를 입력합니다.

- `NoRotation`(기본값)
- `OneDay`
- `OneHour`

- `OneMonth`
- `OneWeek`

인덱스 교체는 오래된 데이터를 만료시킬 수 있도록 인덱스 이름에 타임스탬프를 추가합니다.

h. 에 이벤트를 인덱스로 구성하기 위한 OpenSearch 서비스 유형의 이름을 입력합니다. SearchTypeName

> **Note**
>
> - OpenSearch 서비스 유형 이름은 null 바이트를 제외한 모든 문자를 포함할 수 있지만 로 시작할 수는 없습니다. _
> - OpenSearch 서비스 6.x의 경우 인덱스당 유형이 하나만 있을 수 있습니다. 이미 다른 유형이 있는 기존 색인에 새 유형을 지정하는 경우 Firehose는 런타임 오류를 반환합니다.

i. (선택 사항) StreamBufferingIntervalInSecondsFor 및 StreamBufferingSizeInMB에 수신 이벤트의 버퍼링을 구성하기 위한 값을 입력합니다. 값을 입력하지 않을 경우 300초 및 5MB가 사용됩니다.

j. (선택 사항) 수신 이벤트를 압축하기 위한 다음 StreamCompressionFormat설정 중 하나를 입력합니다.

- `GZIP`
- `SNAPPY`
- `UNCOMPRESSED`(기본값)
- `ZIP`

k. (선택 사항 StreamPrefix) 의 경우 S3 데드레터 버킷에 저장된 파일의 이름을 지정하는 문자열 접두사를 입력합니다. 값을 입력하지 않을 경우 접두사가 사용되지 않습니다.

l. (선택 사항 StreamRetryDurationInSecons) Firehose가 서비스 색인의 이벤트를 인덱싱할 수 없는 경우의 OpenSearch 재시도 기간을 입력합니다. 값을 입력하지 않을 경우 300초가 사용됩니다.

m. (선택 사항 SubscriptionFilterPolicy) 수신 이벤트를 필터링하는 데 사용할 Amazon SNS 구독 필터 정책을 JSON 형식으로 입력합니다. 필터 정책은 OpenSearch 서비스 인덱스에 인덱싱

되는 이벤트를 결정합니다. 값을 입력하지 않을 경우 필터링이 사용되지 않습니다(모든 이벤트가 인덱싱됨).

n. I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications.(본인은 이 앱이 사용자 지정 IAM 역할, 리소스 정책을 생성하고 중첩 애플리케이션을 배포함을 확인합니다.)를 선택하고 배포를 선택합니다.

배포 상태 ***my-app-search***페이지에 Lambda는 애플리케이션 배포 중 상태를 표시합니다.

리소스 섹션에서 AWS CloudFormation이 스택을 생성하기 시작하고 각 리소스에 대해 CREATE_IN_PROGRESS 상태를 표시합니다. 이 과정이 완료되면 AWS CloudFormation에는 CREATE_COMPLETE 상태가 표시됩니다.

배포가 완료되면 Lambda가 Your application has been deployed(애플리케이션이 배포됨) 상태를 표시합니다.

Amazon SNS 주제에 게시된 메시지는 이벤트 검색 및 분석 파이프라인에서 제공하는 OpenSearch 서비스 인덱스에서 자동으로 인덱싱됩니다. 파이프라인이 이벤트를 인덱싱할 수 없을 경우 해당 이벤트가 S3 배달 못한 편지 버킷에 저장됩니다.

이벤트 다시 보기 파이프라인을 배포하고 구독 설정하려면

이 페이지에서는 이벤트 다시 보기 파이프라인을 배포하여 Amazon SNS 주제를 구독 설정하는 방법을 보여줍니다. 이 프로세스는 자동으로 파이프라인과 연결된 AWS SAM 템플릿을 AWS CloudFormation 스택으로 변환한 후 이 스택을 AWS 계정로 배포합니다. 또한 이 프로세스는 Amazon SQS 대기열 및 Lambda 함수 등 이벤트 다시 보기 파이프라인을 구성하는 리소스 세트를 생성하고 구성합니다.

이벤트 필터링에 대한 자세한 정보는 이 설명서의 Amazon SNS 구독 필터 정책에서 확인하세요.

1. AWS Lambda 콘솔에 로그인합니다.
2. 탐색 창에서 함수를 선택한 후 함수 생성을 선택합니다.
3. 함수 생성 페이지에서 다음을 수행합니다.

   a. 서버리스 앱 리포지토리 찾아보기, 퍼블릭 애플리케이션, 사용자 지정 IAM 역할 또는 리소스 정책을 생성하는 앱 표시를 선택합니다.

   b. `fork-event-replay-pipeline`을 검색하여 이 애플리케이션을 선택합니다.
4. [fork-event-replay-pipeline (세부 정보 구성)] 페이지에서 다음을 수행합니다.

a. Application settings(애플리케이션 설정) 섹션에서 애플리케이션 이름을 입력합니다(예: `my-app-replay`).

> **Note**
>
> 각 배포에서 애플리케이션 이름이 고유해야 합니다. 애플리케이션 이름을 재사용할 경우 배포가 이전에 배포된 AWS CloudFormation 스택만 업데이트합니다(새 스택을 생성하지 않음).

b. (선택 사항) 애플리케이션의 Lambda 함수 실행을 위한 다음 LogLevel설정 중 하나를 입력합니다.

- `DEBUG`
- `ERROR`
- `INFO`(기본값)
- `WARNING`

c. (선택 사항) 에 ReplayQueueRetentionPeriodInSecondsAmazon SQS 재생 대기열에 메시지가 보관되는 시간을 초 단위로 입력합니다. 값을 입력하지 않을 경우 1,209,600초(14일)가 사용됩니다.

d. 의 TopicArn경우 이 포크 파이프라인 인스턴스를 구독할 Amazon SNS 주제의 ARN을 입력합니다.

e. 에 DestinationQueueNameLambda 재생 함수가 메시지를 전달하는 Amazon SQS 대기열의 이름을 입력합니다.

f. (선택 사항 SubscriptionFilterPolicy) 수신 이벤트를 필터링하는 데 사용할 Amazon SNS 구독 필터 정책을 JSON 형식으로 입력합니다. 이 필터 정책은 다시 보기를 위해 버퍼링할 이벤트를 결정합니다. 값을 입력하지 않을 경우 필터링이 사용되지 않습니다(모든 이벤트가 버퍼링됨).

g. I acknowledge that this app creates custom IAM roles, resource policies and deploys nested applications.(본인은 이 앱이 사용자 지정 IAM 역할, 리소스 정책을 생성하고 중첩 애플리케이션을 배포함을 확인합니다.)를 선택하고 배포를 선택합니다.

배포 상태 ***my-app-replay***페이지에 Lambda는 애플리케이션 배포 중 상태를 표시합니다.

리소스 섹션에서 AWS CloudFormation이 스택을 생성하기 시작하고 각 리소스에 대해 CREATE_IN_PROGRESS 상태를 표시합니다. 이 과정이 완료되면 AWS CloudFormation에는 CREATE_COMPLETE 상태가 표시됩니다.

배포가 완료되면 Lambda가 Your application has been deployed(애플리케이션이 배포됨) 상태를 표시합니다.

Amazon SNS 주제에 게시되는 메시지는 이벤트 다시 보기 파이프라인에 의해 프로비저닝되는 Amazon SQS 대기열에서 다시 보기를 위해 자동으로 버퍼링됩니다.

> **Note**
>
> 기본적으로 다시 보기는 비활성화됩니다. 다시 보기를 활성화하려면 Lambda 콘솔에서 해당 함수의 페이지로 이동하여 Designer(디자이너) 섹션을 확장하고, SQS 타일을 선택한 다음, SQS 섹션에서 활성을 선택합니다.

Amazon SNS와 함께 Amazon EventBridge 스케줄러 사용

Amazon EventBridge 스케줄러는 하나의 중앙 관리형 서비스에서 작업을 생성, 실행 및 관리할 수 있는 서버리스 스케줄러입니다. EventBridge 스케줄러를 사용하면 반복 패턴에 대해 cron 및 rate 표현식을 사용하여 일정을 만들거나 일회성 간접 호출을 구성할 수 있습니다. 전송을 위한 유연한 기간을 설정하고, 재시도 제한을 정의하고, 실패한 API 간접 호출의 최대 보존 시간을 설정할 수 있습니다.

이 페이지에서는 EventBridge 스케줄러를 사용하여 일정에 따라 Amazon SNS 주제의 메시지를 게시하는 방법을 설명합니다.

주제

실행 역할 설정

새 일정을 생성할 때 EventBridge 스케줄러에 사용자를 대신하여 대상 API 작업을 간접적으로 호출할 수 있는 권한이 있어야 합니다. 실행 역할을 사용하여 EventBridge 스케줄러에 이러한 권한을

부여합니다. 일정의 실행 역할에 연결하는 권한 정책은 필요한 권한을 정의합니다. 이러한 권한은 EventBridge 스케줄러가 간접적으로 호출하려는 대상 API에 따라 달라집니다.

다음 절차와 같이 EventBridge 스케줄러 콘솔을 사용하여 일정을 생성하면 EventBridge 스케줄러가 선택한 대상을 기준으로 실행 역할을 자동으로 설정합니다. EventBridge 스케줄러 SDK, AWS CLI 또는 AWS CloudFormation 중 하나를 사용하여 일정을 생성하려면 EventBridge 스케줄러가 대상을 간접적으로 호출하는 데 필요한 권한을 부여하는 기존 실행 역할이 있어야 합니다. 일정에 대한 실행 역할을 수동으로 설정하는 방법에 대한 자세한 내용은 EventBridge 스케줄러 사용 설명서의 Setting up an execution role을 참조하세요.

일정 생성

콘솔을 사용하여 일정 생성

1. https://console.aws.amazon.com/scheduler/home에서 Amazon EventBridge 스케줄러 콘솔을 엽니다.
2. 일정 페이지에서 일정 생성을 선택합니다.
3. 일정 세부 정보 지정 페이지의 일정 이름 및 설명 섹션에서 다음을 수행합니다.

   a. 일정 이름에 일정의 이름을 입력합니다. 예: **`MyTestSchedule`**.

   b. (선택 사항) 설명에 일정에 대한 설명을 입력합니다. 예: **`My first schedule`**.

   c. 일정 그룹 드롭다운 목록에서 일정 그룹을 선택합니다. 그룹이 없는 경우 기본값을 선택합니다. 일정 그룹을 생성하려면 자체 일정 생성을 선택합니다.

   일정 그룹을 사용하여 일정 그룹에 태그를 추가합니다.

4. • 일정 옵션을 선택합니다.

발생	수행할 작업	
일회성 일정 일회성 일정은 사용자가 지정하는 날짜와 시간에 한 번만 대상을 간접적으로 호출합니다.	날짜 및 시간에 대해 다음을 수행합니다. • YYYY/MM/DD 형식으로 유효한 날짜를 입력합니다.	

발생	수행할 작업	
	• 24시간 hh:mm 형식으로 타임스탬프를 입력합니다. • 시간대에서 시간대를 선택하세요.	

발생	수행할 작업	
반복되는 일정 반복 일정은 cron 표현식 또는 rate 표현식을 사용하여 지정한 속도로 대상을 간접적으로 호출합니다.	a. Schedule type(일정 유형)에서 다음 중 하나를 수행합니다. • cron 식을 사용하여 일정을 정의하려면 Cron-based schedule(Cron 기반 일정)을 선택하고 cron 식을 입력합니다. • rate 식을 사용하여 일정을 정의하려면 Rate-based schedule(Rate 기반 일정)을 선택하고 rate 식을 입력합니다. cron 및 rate 표현식에 대한 자세한 내용은 EventBridge Scheduler 사용 설명서의 EventBridge 스케줄러의 스케줄 유형을 참조하세요. b. 유연한 기간에서 끄기를 선택하여 옵션을 끄거나 미리 정의된 기간 중 하나를 선택합니다. 예를 들어, 15분을 선택하고 1시간에 한 번씩 대상을 간접적으로 호출하도록 반복 일정을 설정하면 일정은 매시간 시작 후 15분 이내에 실행됩니다.	

5. (선택 사항) 이전 단계에서 반복 일정을 선택한 경우 기간 섹션에서 다음을 수행합니다.

   a. 시간대에서 시간대를 선택합니다.

b. 시작 날짜 및 시간에 YYYY/MM/DD 형식으로 유효한 날짜를 입력한 다음 24시간 hh:mm 형식으로 타임스탬프를 지정합니다.

c. 종료 날짜 및 시간에 YYYY/MM/DD 형식으로 유효한 날짜를 입력한 다음 24시간 hh:mm 형식으로 타임스탬프를 지정합니다.

6. 다음(Next)을 선택합니다.

7. 대상 선택 페이지에서 EventBridge 스케줄러가 간접적으로 호출하는 AWS API 작업을 선택합니다.

a. Amazon SNS 게시를 선택합니다.

b. 게시 섹션에서 SNS 주제를 선택하거나 새 SNS 주제 생성을 선택합니다.

c. (선택 사항) JSON 페이로드를 입력합니다. 페이로드를 입력하지 않으면 EventBridge 스케줄러는 빈 이벤트를 사용하여 함수를 간접적으로 호출합니다.

8. 다음(Next)을 선택합니다.

9. 설정 페이지에서 다음 작업을 수행하십시오.

a. 일정을 켜려면 일정 상태에서 일정 활성화를 토글합니다.

b. 일정에 대한 재시도 정책을 구성하려면 재시도 정책 및 데드-레터 큐(DLQ)에서 다음을 수행합니다.

- 재시도를 토글합니다.
- 최대 이벤트 수명에 EventBridge 스케줄러가 처리되지 않은 이벤트를 보관해야 하는 최대 시간과 분을 입력합니다.
- 최대 시간은 24시간입니다.
- 최대 재시도 횟수에는 대상이 오류를 반환할 경우 EventBridge 스케줄러가 일정을 재시도하는 최대 횟수를 입력합니다.

  최댓값은 185회입니다.

재시도 정책을 사용하면 일정이 대상을 간접적으로 호출하지 못할 경우 EventBridge 스케줄러가 일정을 다시 실행합니다. 구성된 경우 일정에 대한 최대 보존 기간과 재시도 횟수를 설정해야 합니다.

c. EventBridge 스케줄러가 전송되지 않은 이벤트를 저장하는 위치를 선택합니다.

DLQ(Dead Letter Queue) 옵션	수행할 작업	
저장 안 함	[None]을 선택합니다.	
일정을 생성하는 위치와 같은 AWS 계정에 이벤트 저장	a. 내 AWS 계정의 Amazon SQS 대기열을 DLQ로 선택을 선택합니다. b. Amazon SQS 대기열의 Amazon 리소스 이름(ARN)을 선택합니다.	
일정을 생성하는 위치와 다른 AWS 계정에 이벤트 저장	a. 다른 AWS 계정의 Amazon SQS 대기열을 DLQ로 지정을 선택합니다. b. Amazon SQS 대기열의 Amazon 리소스 이름(ARN)을 입력합니다.	

d. 고객 관리형 키를 사용하여 대상 입력을 암호화하려면 암호화에서 암호화 설정 사용자 지정(고급)을 선택합니다.

   이 옵션을 선택하는 경우 기존 KMS 키 ARN을 입력하거나 AWS KMS key 생성을 선택하여 AWS KMS 콘솔로 이동합니다. EventBridge 스케줄러가 저장 데이터를 암호화하는 방법에 대한 자세한 내용은 Amazon EventBridge 스케줄러 사용 설명서의 Encryption at rest를 참조하세요.

e. EventBridge 스케줄러가 새 실행 역할을 생성하도록 하려면 이 일정에 대한 새 역할 생성을 선택합니다. 그런 다음 역할 이름을 입력합니다. 이 옵션을 선택하면 EventBridge 스케줄러가 템플릿 대상에 필요한 필수 권한을 역할에 연결합니다.

10. 다음(Next)을 선택합니다.
11. 일정 검토 및 생성 페이지에서 일정의 세부 정보를 검토합니다. 각 섹션에서 편집을 선택하여 해당 단계로 돌아가서 세부 정보를 편집합니다.
12. 일정 생성을 선택합니다.

일정 페이지에서 새 일정과 기존 일정 목록을 볼 수 있습니다. 상태 열에서 새 일정이 활성화됨 상태인지 확인합니다.

관련 리소스

EventBridge 스케줄러에 대한 자세한 내용은 다음을 참조하세요.

- EventBridge 스케줄러 사용 설명서
- EventBridge 스케줄러 API 참조
- EventBridge 스케줄러 요금

애플리케이션 대 개인(A2P) 메시징에 Amazon SNS 사용

이 섹션에서는 모바일 애플리케이션, 휴대폰 번호 및 이메일 주소 등의 구독자를 통해 사용자 알림에 Amazon SNS를 사용하는 방법에 대한 정보를 제공합니다.

주제

- 모바일 문자 메시지(SMS)
- 모바일 푸시 알림
- 이메일 알림

모바일 문자 메시지(SMS)

사용자는 Amazon SNS를 사용하여 SMS 수신 가능한 디바이스에 문자 메시지 또는 SMS 메시지를 전송할 수 있습니다. 전화번호로 메시지를 직접 전송할 수 있으며, 전화번호에서 주제를 구독하고 메시지를 주제로 전송하여 메시지를 여러 전화번호로 한 번에 전송할 수 있습니다.

AWS 계정에 대한 SMS 기본 설정을 지정하여 사용 사례 및 예산에 맞게 SMS 전송을 조정할 수 있습니다. 예를 들어, 메시지가 비용에 맞게 최적화되는지 또는 안정성 있는 전송을 위해 최적화되는지를 선택할 수 있습니다. 또한 개별 메시지 전송에 대한 지출 할당량 및 AWS 계정에 대한 월 지출 할당량을 지정할 수 있습니다.

현지 법률 및 규정에서 요구하는 경우(예: 미국 및 캐나다), SMS 수신자는 옵트아웃을 통해 AWS 계정에서 SMS 메시지 수신을 중지하도록 선택할 수 있습니다. 수신자가 옵트아웃한 후, 메시지 전송을 다시 시작할 수 있도록 제한 사항을 적용하여 전화 번호를 다시 옵트인할 수 있습니다.

Amazon SNS는 여러 리전에서 SMS 메시징을 지원하며, 200개 이상의 국가 및 리전에 메시지를 전송할 수 있습니다. 자세한 내용은 지원되는 국가 및 리전 섹션을 참조하세요.

주제

- SMS 샌드박스
- SMS 메시지에 대한 발신 자격 증명
- Amazon SNS로 SMS 메시징 지원 요청
- SMS 메시징 기본 설정 지정
- SMS 메시지 전송

- SMS 활동 모니터링
- 전화번호 및 SMS 구독 관리
- 지원되는 국가 및 리전
- SMS 모범 사례

SMS 샌드박스

Amazon SNS를 사용하여 SMS 메시지를 보내기 시작하는 경우 AWS 계정이 SMS 샌드박스에 있습니다. SMS 샌드박스는 SMS 발신자로서의 평판을 위협하지 않고 Amazon SNS 기능을 사용해 볼 수 있는 안전한 환경을 제공합니다. 계정이 SMS 샌드박스에 있는 동안 다음 제한 사항과 함께 Amazon SNS의 모든 기능을 사용할 수 있습니다.

- 확인된 대상 전화번호로만 SMS 메시지를 보낼 수 있습니다.
- 확인된 대상 전화번호는 최대 10개까지 보유할 수 있습니다.
- 대상 전화번호는 확인 또는 마지막 확인 시도 이후 24시간 이상이 경과한 후에만 삭제할 수 있습니다.

계정이 샌드박스 외부로 이동하면 이러한 제한이 제거되고 모든 수신자에게 SMS 메시지를 보낼 수 있습니다.

주제

- SMS 샌드박스에서 전화번호 추가 및 확인
- SMS 샌드박스에서 전화번호 삭제
- SMS 샌드박스 환경에서 나가기

SMS 샌드박스에서 전화번호 추가 및 확인

AWS 계정이 SMS 샌드박스에 있는 동안 SMS 메시지 전송을 시작하려면 원본 ID를 만들고 대상 전화번호를 추가한 다음 인증하세요.

> **Note**
>
> SMS 샌드박스에 없는 계정과 마찬가지로 일부 국가 또는 리전의 수신자에게 SMS 메시지를 보내려면 먼저 발신 자격 증명이 필요합니다. 자세한 정보는 지원되는 국가 및 리전을 참조하세요.

발신 ID에는 발신자 ID와 다양한 유형의 발신 번호가 포함됩니다. 기존 발신 번호를 보려면 Amazon SNS 콘솔의 탐색 창에서 발신 번호를 선택합니다. 현재 발신자 ID는 이 목록에 표시되지 않습니다.

대상 전화번호를 추가하고 확인하려면

1. Amazon SNS 콘솔에 로그인합니다.
2. 전화번호의 발신 자격 증명을 생성합니다.
3. 콘솔 메뉴에서 SMS 메시징을 지원하는 AWS 리전을 선택합니다.
4. 탐색 창에서 문자 메시지(SMS)를 선택합니다.
5. 모바일 문자 메시징(SMS) 페이지의 샌드박스 대상 전화번호에서 전화번호 추가를 선택합니다.
6. 대상 세부 정보 아래에 국가 코드와 전화번호를 입력하고 확인 메시지에 사용할 언어를 지정한 다음 전화번호 추가를 선택합니다.

   Amazon SNS는 대상 전화번호로 일회용 암호(OTP)를 전송합니다. 대상 전화번호가 15분 이내에 OTP를 받지 못하면 확인 코드 재전송을 선택합니다. 동일한 대상 전화번호로 OTP를 24시간마다 최대 5회까지 보낼 수 있습니다.
7. 확인 코드 상자에 대상 전화번호로 전송된 OTP를 입력한 다음 전화번호 확인을 선택합니다.

   대상 전화번호와 확인 상태는 샌드박스 대상 전화번호 섹션에 나타납니다. 확인 상태가 보류 중이면 확인에 실패한 것입니다. 예를 들어, 전화번호와 함께 국가 코드를 입력하지 않은 경우 이러한 일이 발생할 수 있습니다. 확인 또는 마지막 확인 시도 이후 24시간 이상이 경과한 후에만 보류 중이거나 확인된 대상 전화번호를 삭제할 수 있습니다.
8. 이 대상 전화번호를 사용할 각 리전에서 이 단계를 반복합니다.

OTP 문자 미수신 문제 해결

전화번호가 OTP 문자를 수신하지 못할 수 있는 일반적인 문제를 해결합니다.

- Amazon SNS SMS 지출 한도: SMS 메시지 전송에 대한 지출 한도를 초과한 경우 AWS 계정 한도가 증가하거나 청구 문제가 해결될 때까지 OTP 문자를 포함한 추가 메시지가 전송되지 않을 수 있습니다.
- SMS 알림 수신이 거부된 전화번호: 일부 국가 또는 지역에서는 수신자가 OTP 문자에 일반적으로 사용되는 단축 번호로 SMS 메시지를 수신하도록 선택해야 합니다. 수신자의 전화번호가 옵트인되지 않은 경우 OTP 문자를 받지 못합니다.

- 이동통신사 제한 또는 필터링: 일부 이동통신사에는 OTP 문자를 비롯한 특정 유형의 SMS 메시지 전송을 차단하는 제한 또는 필터링 메커니즘이 있을 수 있습니다. 이는 이동통신사에서 구현한 보안 정책 또는 스팸 방지 조치 때문일 수 있습니다.
- 잘못된 전화번호 또는 잘못된 전화번호: 수신자가 제공한 전화번호가 정확하지 않거나 유효하지 않은 경우 OTP 문자가 전달되지 않습니다.
- 네트워크 문제: 일시적인 네트워크 문제 또는 중단으로 인해 OTP 문자를 포함한 SMS 메시지가 수신자의 전화로 전송되지 않을 수 있습니다.
- 전송 지연: 경우에 따라 네트워크 혼잡 또는 기타 요인으로 인해 SMS 메시지 전송이 지연될 수 있습니다. OTP 텍스트는 결국 전달될 수 있지만 예상 시간보다 지연될 수 있습니다.
- 계정 일시 중지 또는 해지: 미결제 또는 서비스 AWS 약관 위반과 같은 문제가 발생하는 경우 OTP 문자를 포함한 Amazon SNS 메시징 기능이 일시 중지되거나 종료될 수 있습니다. AWS 계정

SMS 샌드박스에서 전화번호 삭제

SMS 샌드박스에서 보류 중이거나 확인된 대상 전화번호를 삭제할 수 있습니다.

SMS 샌드박스에서 대상 전화번호를 삭제하려면

1. 전화번호 확인 후 24시간 또는 마지막 확인 시도 후 24시간을 기다립니다.
2. Amazon SNS 콘솔에 로그인합니다.
3. 콘솔 메뉴에서 대상 전화번호를 추가한 SMS 메시징을 지원하는 AWS 리전을 선택합니다.
4. 탐색 창에서 문자 메시지(SMS)를 선택합니다.
5. 모바일 문자 메시징(SMS) 페이지의 샌드박스 대상 전화번호에서 삭제할 전화번호를 선택한 다음 전화번호 삭제를 선택합니다.
6. 전화번호를 삭제할 것인지 확인하려면 **`delete me`**을 입력한 다음, 삭제를 선택합니다.

   대상 전화번호를 확인하거나 확인을 시도한 후 24시간 이상이 경과하면 해당 전화번호가 삭제되고 Amazon SNS가 대상 전화번호 목록을 업데이트합니다.
7. 대상 전화번호를 추가하고 더 이상 사용하지 않을 각 리전에서 이 단계를 반복합니다.

SMS 샌드박스 환경에서 나가기

SMS 샌드박스 AWS 계정 밖으로 이동하려면 먼저 대상 전화번호를 추가하고, 확인하고, 테스트해야 합니다. 그런 다음 를 사용하여 케이스를 AWS Support생성해야 합니다.

AWS 계정을 SMS 샌드박스에서 제외하도록 요청하려면

1. 전화번호 확인

   a. SMS 샌드박스에 있는 동안 Amazon SNS 콘솔을 여십시오. AWS 계정

   b. 탐색 창의 모바일에서 문자 메시지 (SMS) 를 선택합니다.

   c. 샌드박스 대상 전화번호 섹션에서 하나 이상의 대상 전화번호를 추가하고 확인합니다. 이렇게 확인하면 메시지를 성공적으로 보내고 받을 수 있습니다.

2. SMS 게시 테스트

   • 하나 이상의 확인된 전화번호로 메시지를 보내고 받을 수 있는지 확인하세요. SMS 메시지를 게시하는 방법에 대한 자세한 지침은 을 참조하십시오휴대폰에 게시.

3. 샌드박스 편집 시작

   • Amazon SNS 콘솔의 모바일 문자 메시징(SMS) 페이지의 계정 정보에서 SMS 샌드박스 종료를 선택합니다. 이 작업을 수행하면 Amazon Support Center로 리디렉션되고 서비스 할당량 증가 옵션이 선택된 지원 사례가 자동으로 생성됩니다.

4. 양식을 작성해 주세요.

   • 서비스 할당량 증가의 지원 양식에서 다음을 수행하십시오.

      i. SNS 문자 메시징을 서비스로 선택하십시오.

      ii. SMS 메시지를 보내려는 웹 사이트 URL 또는 앱 이름을 입력합니다.

      iii. 보낼 메시지 유형을 일회용 암호, 홍보용 또는 트랜잭션 형식으로 지정합니다.

      iv. SMS 메시지를 보낼 형식을 선택합니다. AWS 리전

      v. SMS 메시지를 보내려는 국가 또는 지역을 나열하십시오.

      vi. 고객이 메시지 수신을 옵트인하는 방법을 설명하세요.

      vii. 사용하려는 메시지 템플릿을 모두 포함하세요.

5. 할당량 및 지역을 지정하세요.

   • 요청에서 다음과 같이 하세요.

      i. 이동할 AWS 리전위치를 선택하세요 AWS 계정.

      ii. 리소스 유형에 대한 일반 제한을 선택합니다.

      iii. 할당량 SMS 샌드박스 종료를 선택합니다.

iv. (선택 사항) 추가 증가 또는 기타 조정을 요청하려면 다른 요청 추가를 선택하고 필요한 세부 정보를 지정합니다.

v. 새 할당량 값에 요청하려는 한도를 USD로 입력합니다.

6. 추가 세부 정보

   a. 사례 설명에 요청과 관련된 추가 세부 정보를 입력하십시오.

   b. 연락처 옵션에서 원하는 연락처 언어를 선택합니다.

7. 요청을 제출하십시오.

   - 제출을 선택하여 요청을 로 보내십시오 AWS Support.

AWS Support 팀에서 24시간 이내에 요청에 대한 초기 응답을 제공합니다.

시스템이 원치 않는 콘텐츠 또는 악성 콘텐츠를 전송하지 않도록 하기 위해 각 요청을 신중하게 고려합니다. 가능한 경우 이 24시간 기간 내에 요청이 승인될 것입니다. 그러나 사용자의 추가 정보가 필요한 경우 요청을 해결하는 데 시간이 오래 걸릴 수도 있습니다.

정책에 맞지 않는 사용 사례인 경우, 요청에 대한 권한을 부여하지 않을 수도 있습니다.

SMS 메시지에 대한 발신 자격 증명

Amazon SNS를 사용하여 SMS 메시지를 보낼 때 다음 유형의 발신 자격 증명을 사용하여 수신자에게 자신을 식별할 수 있습니다.

- 발신자 ID
- 발신 번호

> **Note**
>
> Amazon SNS SMS 메시징은 현재 Amazon Pinpoint가 지원되지 않는 리전에서 사용할 수 있습니다. 유럽(스톡홀름), 중동(바레인), 유럽(파리), 남아메리리카(상파울루) 또는 미국 서부(캘리포니아 북부)에서 운영하는 경우 미국 동부(버지니아 북부) 리전의 Amazon Pinpoint 콘솔을 열고 10DLC 회사 및 캠페인을 등록하되, 10DLC 번호는 요청하지 마십시오. 대신 AWS Service Quotas 콘솔을 사용하여 해당 리전의 10DLC 번호를 요청하는 동안 서비스 한도 증가 사례를 생성합니다. 발신 자격 증명을 요청하는 방법에 대해 자세히 알아보려면 Amazon SNS로 SMS 메시징 지원 요청에서 확인하세요.

발신자 ID

발신자 ID는 SMS 메시지의 발신자로 식별되는 알파벳 이름입니다. 발신자 ID를 이용해 SMS 메시지를 보내고 수신자가 발신자 ID 인증이 지원되는 지역에 있는 경우, 수신자의 디바이스에 전화번호 대신 발신자 ID가 나타납니다. 발신자 ID는 발신자에 대하여 전화번호, 긴 코드 또는 단축 코드보다 많은 정보를 SMS 수신자에게 제공합니다.

발신자 ID는 세계 여러 국가 및 리전에서 지원됩니다. 일부 지역에서는 기업이 개별 고객에게 SMS 메시지를 발송하는 경우, 규제기관이나 산업 그룹에 사전 등록한 발신자 ID를 써야 합니다. 발신자 ID를 지원하거나 요구하는 국가 및 리전의 전체 목록은 지원되는 국가 및 리전을 확인하십시오.

발신자 ID 사용에 따르는 추가 요금은 없습니다. 그러나 발신자 ID 인증에 대한 지원 및 요구 사항은 다양합니다. 주요 시장(캐나다, 중국, 미국 포함)에서는 발신자 ID 사용을 지원하지 않습니다. 일부 리전에서는 기업이 개별 고객에게 SMS 메시지를 전송하는 경우, 규제기관이나 산업 그룹에 사전 등록한 발신자 ID를 써야 합니다.

> **Important**
>
> AWS는 발신자 ID가 다른 사람, 회사 또는 제품을 사칭하는 데 사용되는 SMS 스푸핑을 금지합니다. 귀하가 소유한 브랜드 또는 상표를 나타내는 발신자 ID만 사용하세요.

장점

발신자 ID는 메시지 발신자에 대한 더 많은 정보를 수신자에게 제공합니다. 단축 코드나 긴 코드를 사용하기보다는 발신자 ID를 사용하여 브랜드 자격 증명을 수립하는 편이 쉽습니다. 발신자 ID 사용에 따르는 추가 요금은 없습니다.

단점

발신자 ID 인증의 지원 및 요건은 모든 국가 및 리전에 일관된 것이 아닙니다. 주요 시장(캐나다, 중국, 미국 포함)에서는 발신자 ID를 지원하지 않습니다. 일부 지역에서는 발신자 ID를 사용하기 전에 규제기관에 사전 등록해야 합니다.

국가별 발신자 ID 등록

SMS 메시징에 대한 발신자 ID를 등록하려면 AWS Support에 사례를 개설해야 합니다. 지원 사례 개설 후 AWS에서는 추가 필수 문서를 공유합니다. 또한 발신자 ID를 등록하는 해당 국가에 대해 다음 정보를 제공해야 합니다.

국가 이름	메시지 유형	형식 제한 및 요구 사항	등록 요구 사항
오스트레일리아(AU)	거래 및 홍보	• 영숫자 • 최대 11자 • 공백 금지 • 특수 문자 금지 • 발신자 ID는 SMS를 보내는 회사의 브랜드 이름이어야 함	• 등록하려는 발신자 ID • 사용자는 어느 AWS 리전에서 API/서비스를 호출할지 • 회사 이름 • 회사 주소(회사 도시, 주/도, 우편번호 포함) • 회사 국가 • 회사 URL(앱 또는 회사 웹사이트로 연결되는 링크) • 예상 월별 볼륨 • 사용 사례, 메시지 목적 설명 • 명확하지 않은 경우 회사 이름과 발신자 ID 간의 연관성에 대한 설명 제공 • 회사의 공식 사업자/무역 허가증 번호 또는 VAT # • 보내려는 메시지 템플릿 • 사업자 등록증. 예를 들면 다음과 같습니다(이에 국한되지 않음). • ABN(호주 기업 번호)

국가 이름	메시지 유형	형식 제한 및 요구 사항	등록 요구 사항
			• ACN(호주 회사 번호) • ARBN(호주 등록 기관 번호) • ICN(원주민 법인 번호) • LOA(위임장)

국가 이름	메시지 유형	형식 제한 및 요구 사항	등록 요구 사항
벨로루시(BY)	트랜잭션 메시지 전용	• 영숫자 • 최대 11자 • 공백 금지 • 특수 문자 금지 • 발신자 ID는 SMS를 보내는 회사의 브랜드 이름이어야 함	• 등록하려는 발신자 ID • 사용자는 어느 AWS 리전에서 API/서비스를 호출할지 • 회사 이름 • 회사 주소(회사 도시, 주/도, 우편번호 포함) • 회사 국가 • 회사 연락처 전화번호 • 회사 URL(앱 또는 회사 웹사이트로 연결되는 링크) • 예상 월별 볼륨 • 사용 사례, 메시지 목적 설명 • 명확하지 않은 경우 회사 이름과 발신자 ID 간의 연관성에 대한 설명 제공 • 회사의 공식 사업자/무역 허가증 번호 또는 VAT # • 보내려는 메시지 템플릿

국가 이름	메시지 유형	형식 제한 및 요구 사항	등록 요구 사항
중국(CN) 중국에서는 발신자 ID 등록이 필요하지 않지만 앞에 본문 서명이 추가된 콘텐츠/템플릿 등록은 요구합니다(예: [Amazon]).	다음 키워드는 허용되지 않습니다. • 파룽궁 • SB • 천안문 광장 다음과 같은 범주의 메시지 • 신용카드 • 디지털 결제(암호화폐 포함) • 사기성 트래픽 URL(피싱 또는 스팸) • 도박 • 부적절한 콘텐츠(성인, 폭력, 약물, 알코올) • 대출 • 성형수술 • 정치 • 종교 • 주식 거래 • 가상 화폐 대괄호 안의 서명은 SMS 메시지 본문 앞에 추가해야 합니다. 중국으로 성공적으로	• 최대 11자 • 공백 금지 • 특수 문자 금지	• 회사 이름 • 회사 주소 • 주/도 • 국가 • 회사 전화번호 • 회사 웹사이트 • 예상 월별 볼륨 • 메시지 유형: 프로모션/트랜잭션 • 사용 사례 설명 • 등록하려는 템플릿(전송된 SMS는 규정 준수 및 성공적인 전송을 위해 제공된 템플릿에서 벗어나지 않아야 함). 예를 들어, [회사 이름] 귀하의 일회용 비밀번호는 {OTP}입니다. 이 코드는 10분 후에 만료됩니다. • 프로모션 콘텐츠를 트랜잭션 메시지 유형으로 전송하지 않겠다는 확인 • 메시지 서명 서명 형식 제한 및 요구 사항를 참조하세요.

국가 이름	메시지 유형	형식 제한 및 요구 사항	등록 요구 사항
	전송하려면 메시지 콘텐츠 템플릿과 함께 메시지 서명을 등록해야 합니다. 이 메시지 서명은 전송된 각 SMS의 메시지 내용 앞에 추가해야 합니다. 등록된 서명을 SMS 메시지 본문 앞에 추가하지 않으면 SMS가 차단되거나 필터링될 수 있습니다. 서명은 대괄호에 넣어 SMS 메시지 본문 앞에 추가해야 합니다.		

국가 이름	메시지 유형	형식 제한 및 요구 사항	등록 요구 사항
이집트(EG)	트랜잭션 메시지 전용	• 영숫자 • 최대 11자 • 공백 금지 • 특수 문자 금지	• 등록하려는 발신자 ID • 사용자는 어느 AWS 리전에서 API/서비스를 호출할지 • 회사 이름 • 회사 주소(회사 도시, 주/도, 우편번호 포함) • 회사 국가 • 회사 연락처 전화번호 • 회사 URL(앱 또는 회사 웹사이트로 연결되는 링크) • 예상 월별 볼륨 • 사용 사례, 메시지 목적 설명 • 명확하지 않은 경우 회사 이름과 발신자 ID 간의 연관성에 대한 설명 제공 • 이집트에 법인/사무실이 있는지 아니면 이집트에 위치하지 않은 회사가 이집트로 보내는 것인지 • 사용 사례가 이 발신자 ID에서 전송되는 모든 메시지를 포함한다는 서면 확인서

국가 이름	메시지 유형	형식 제한 및 요구 사항	등록 요구 사항
			• 명확하지 않은 경우 회사 이름과 발신자 ID 간의 연관성 설명 • 보내려는 메시지 템플릿
인도(IN)	트랜잭션 메시지 전용	• 영숫자 • 최대 6자 • 공백 금지 • 특수 문자 금지	인도 발신자 ID 등록 요구 사항를 참조하세요.

국가 이름	메시지 유형	형식 제한 및 요구 사항	등록 요구 사항
요르단(JO)	트랜잭션 메시지 전용	• 영숫자 • 최대 11자 • 공백 금지 • 특수 문자 금지	• 등록하려는 발신자 ID • 사용자는 어느 AWS 리전에서 API/서비스를 호출할지 • 회사 이름 • 회사 주소(회사 도시, 주/도, 우편번호 포함) • 회사 국가 • 회사 연락처 전화번호 • 회사 URL(앱 또는 회사 웹사이트로 연결되는 링크) • 예상 월별 볼륨 • 사용 사례, 메시지 목적 설명 • 명확하지 않은 경우 회사 이름과 발신자 ID 간의 연관성에 대한 설명 제공 • 사업자 등록 증명서 • 보내려는 메시지 유형(예: OTP, 알림) • 사용 사례가 이 발신자 ID에서 전송되는 모든 메시지를 설명한다는 서면 확인서

국가 이름	메시지 유형	형식 제한 및 요구 사항	등록 요구 사항
			• 보내려는 메시지 템플릿

국가 이름	메시지 유형	형식 제한 및 요구 사항	등록 요구 사항
쿠웨이트(KW)	트랜잭션 메시지 전용	• 영숫자 • 최대 11자 • 공백 금지 • 특수 문자 금지	• 등록하려는 발신자 ID • 사용자는 어느 AWS 리전에서 API/서비스를 호출할지 • 회사 이름 • 회사 주소(회사 도시, 주/도, 우편번호 포함) • 회사 국가 • 회사 연락처 전화번호 • 회사 URL(앱 또는 회사 웹사이트로 연결되는 링크) • 예상 월별 볼륨 • 사용 사례, 메시지 목적 설명 • 명확하지 않은 경우 회사 이름과 발신자 ID 간의 연관성에 대한 설명 제공 • 보내려는 메시지 유형(예: OTP, 알림) • 사용 사례가 이 발신자 ID에서 전송되는 모든 메시지를 설명한다는 서면 확인서 • 보내려는 메시지 템플릿

국가 이름	메시지 유형	형식 제한 및 요구 사항	등록 요구 사항
필리핀(PH)	트랜잭션 메시지 전용	• 영숫자 • 최대 11자 • 공백 금지 • 특수 문자 금지	• 등록하려는 발신자 ID • 사용자는 어느 AWS 리전에서 API/서비스를 호출할지 • 회사 이름 • 회사 주소(회사 도시, 주/도, 우편번호 포함) • 회사 국가 • 회사 연락처 전화번호 • 회사 URL(앱 또는 회사 웹사이트로 연결되는 링크) • 예상 월별 볼륨 • 사용 사례, 메시지 목적 설명 • 명확하지 않은 경우 회사 이름과 발신자 ID 간의 연관성에 대한 설명 제공 • 보내려는 메시지 유형(예: OTP, 알림) • 사용 사례가 이 발신자 ID에서 전송되는 모든 메시지를 설명한다는 서면 확인서 • 보내려는 메시지 템플릿

국가 이름	메시지 유형	형식 제한 및 요구 사항	등록 요구 사항
카타르(QA)	트랜잭션 메시지 전용	• 영숫자 • 최대 11자 • 공백 금지 • 특수 문자 금지	• 등록하려는 발신자 ID • 사용자는 어느 AWS 리전에서 API/서비스를 호출할지 • 회사 이름 • 회사 주소(회사 도시, 주/도, 우편번호 포함) • 회사 국가 • 회사 연락처 전화번호 • 회사 URL(앱 또는 회사 웹사이트로 연결되는 링크) • 예상 월별 볼륨 • 사용 사례, 메시지 목적 설명 • 명확하지 않은 경우 회사 이름과 발신자 ID 간의 연관성에 대한 설명 제공 • 전송 중인 SMS 유형 • (트랜잭션/프로모션/OTP) 규정 준수를 위해 트랜잭션/OTP 메시지만 카타르로 보내는 발신자 ID와 함께 사용할 수 있다는 점을 참고하세요.

국가 이름	메시지 유형	형식 제한 및 요구 사항	등록 요구 사항
			• 사용 사례가 이 발신자 ID에서 전송되는 모든 메시지를 설명한다는 서면 확인서 • 보내려는 메시지 템플릿

국가 이름	메시지 유형	형식 제한 및 요구 사항	등록 요구 사항
러시아(RU)	트랜잭션 메시지 전용	• 영숫자 • 최대 11자 • 공백 금지 • 특수 문자 금지	• 등록하려는 발신자 ID • 사용자는 어느 AWS 리전에서 API/서비스를 호출할지 • 회사 이름 • 회사 주소(회사 도시, 주/도, 우편번호 포함) • 회사 국가 • 회사 연락처 전화번호 • 회사 URL(앱 또는 회사 웹사이트로 연결되는 링크) • 납세 ID 또는 라이선스 번호 • 사업자 등록 증명서 • 연락처 이메일 주소 • 예상 월별 볼륨 • 사용 사례, 메시지 목적 설명 • 명확하지 않은 경우 회사 이름과 발신자 ID 간의 연관성에 대한 설명 제공 • 이 사용 사례가 이 계정에서 러시아로 전송되는 모든 메시지에 적용된다는 확인

국가 이름	메시지 유형	형식 제한 및 요구 사항	등록 요구 사항
			• 트랜잭션이 아닌 메시지는 별도의 발신자 ID를 사용하여 전송해야 한다는 것을 인지한다는 확인 • 사용료 $272/월 이 정기 월 사용료를 승인하는지 확인해 주세요. 예/아니요 • 보내려는 메시지 템플릿

국가 이름	메시지 유형	형식 제한 및 요구 사항	등록 요구 사항
사우디아라비아(SA)	각 발신자 ID는 트랜잭션용이거나 프로모션용이어야 합니다. 하나의 발신자 ID를 두 가지 유형의 트래픽에 모두 사용할 수 없습니다. OTP 또는 2FA 트래픽을 전송하는 경우 발신자 ID를 해당 용도로만 사용해야 합니다. 프로모션 발신자 ID에는 사우디아라비아의 방해 금지(DND) 목록이 적용됩니다.	• 프로모션 발신자 ID 길이: 2~8자, 발신자 ID 앞에 “'-AD'가 붙음 • 트랜잭션 발신자 ID 길이: 2~11자 • 발신자 ID는 발신자의 브랜드 아이덴티티를 나타내야 함 • 1자 이상의 문자 포함 • ASCII 특수 문자(예: #, @) 사용 금지 • 대문자와 소문자, 숫자 0~9 포함 가능	사우디아라비아의 발신자 ID 등록 지원은 해외 기업에만 해당합니다. 현재 사우디아라비아 현지 회사의 발신자 ID 등록은 지원하지 않습니다. • 등록하려는 발신자 ID • 사용자는 어느 AWS 리전에서 API/서비스를 호출할지 • 회사 이름 • 회사 주소(회사 도시, 주/도, 우편번호 포함) • 회사 국가 • 회사 연락처 전화번호 • 회사 URL(앱 또는 회사 웹사이트로 연결되는 링크) • 예상 월별 볼륨 • 사용 사례, 메시지 목적 설명 • 명확하지 않은 경우 회사 이름과 발신자 ID 간의 연관성에 대한 설명 제공 • 보내려는 메시지 템플릿.

국가 이름	메시지 유형	형식 제한 및 요구 사항	등록 요구 사항
			• 이 발신자 ID가 트랜잭션 콘텐츠에 사용되는지 아니면 프로모션 콘텐츠에 사용되는지 • 트랜잭션이 아닌 메시지는 별도의 발신자 ID를 사용하여 전송해야 한다는 것을 인지한다는 확인 • 2FA 또는 OTP 트래픽을 전송하는 경우 발신자 ID를 해당 용도로만 사용한다는 확인
싱가포르(SG)	트랜잭션 메시지 전용	• 최대 11자 • 공백 금지 • 특수 문자 금지	싱가포르 발신자 ID 등록 요구 사항를 참조하세요.

국가 이름	메시지 유형	형식 제한 및 요구 사항	등록 요구 사항
스리랑카(LK)	제한 또는 특별 요구 사항 없음	• 영숫자 • 최대 11자 • 공백 금지 • 특수 문자 금지	• 등록하려는 발신자 ID • 사용자는 어느 AWS 리전에서 API/서비스를 호출할지 • 회사 이름 • 회사 유형(현지/국외) • 회사 URL(앱 또는 회사 웹사이트로 연결되는 링크) • 사용 사례, 메시지 목적 설명 • 보내려는 메시지 템플릿 • 이 발신자 ID가 트랜잭션 콘텐츠에 사용되는지 아니면 프로모션 콘텐츠에 사용되는지 • 트랜잭션이 아닌 메시지는 별도의 발신자 ID를 사용하여 전송해야 한다는 것을 인지한다는 확인 • 2FA 또는 OTP 트래픽을 전송하는 경우 발신자 ID를 해당 용도로만 사용한다는 확인

국가 이름	메시지 유형	형식 제한 및 요구 사항	등록 요구 사항
태국(TH)	제한 또는 특별 요구 사항 없음	• 영숫자 • 최대 11자 • 공백 금지 • 특수 문자 금지	• 등록하려는 발신자 ID • 사용자는 어느 AWS 리전에서 API/서비스를 호출할지 • 회사 이름 • 회사 주소(회사 도시, 주/도, 우편번호 포함) • 회사 국가 • 회사 연락처 전화번호 • 회사 URL(앱 또는 회사 웹사이트로 연결되는 링크) • 예상 월별 볼륨 • 사용 사례, 메시지 목적 설명 • 명확하지 않은 경우 회사 이름과 발신자 ID 간의 연관성에 대한 설명 제공 • 전송되는 SMS 유형(트랜잭션/프로모션/OTP) • 보내려는 메시지 템플릿

국가 이름	메시지 유형	형식 제한 및 요구 사항	등록 요구 사항
터키(TR)	제한 또는 특별 요구 사항 없음	• 영숫자 • 최대 11자 • 공백 금지 • 특수 문자 금지	• 등록하려는 발신자 ID • 사용자는 어느 AWS 리전에서 API/서비스를 호출할지 • 회사 이름 • 회사 주소(회사 도시, 주/도, 우편번호 포함) • 회사 URL(앱 또는 회사 웹사이트로 연결되는 링크) • 회사가 터키 현지에 있는지 아니면 국외에 있는지 • 예상 월별 볼륨 • 사용 사례, 메시지 목적 설명 • 명확하지 않은 경우 회사 이름과 발신자 ID 간의 연관성에 대한 설명 제공 • 전송되는 SMS 유형(트랜잭션/프로모션/OTP) • 보내려는 메시지 템플릿

국가 이름	메시지 유형	형식 제한 및 요구 사항	등록 요구 사항
우크라이나(UA)	제한 또는 특별 요구 사항 없음	• 영숫자 • 최대 11자 • 공백 금지 • 특수 문자 금지	• 등록하려는 발신자 ID • 사용자는 어느 AWS 리전에서 API/서비스를 호출할지 • 회사 이름 • 회사 주소(회사 도시, 주/도, 우편번호 포함) • 회사 URL(앱 또는 회사 웹사이트로 연결되는 링크) • VAT 번호 • 현지 또는 국외 기업 • 예상 월별 볼륨 • 사용 사례, 메시지 목적 설명 • 명확하지 않은 경우 회사 이름과 발신자 ID 간의 연관성에 대한 설명 제공 • 전송되는 SMS 유형(트랜잭션/프로모션/OTP) • 보내려는 메시지 템플릿

국가 이름	메시지 유형	형식 제한 및 요구 사항	등록 요구 사항
아랍에미리트연합국 (UAE)	트랜잭션 메시지 전용	• 영숫자 • 포괄적인 발신자 ID는 허용되지 않으며 브랜드 나타내야 함 • 최대 11자 • 공백 금지 • 특수 문자 금지	• 등록하려는 발신자 ID • 사용자는 어느 AWS 리전에서 API/서비스를 호출할지 • 회사 이름 • 회사 주소(회사 도시, 주/도, 우편번호 포함) • 회사 국가 • 회사 연락처 전화번호 • 회사 URL(앱 또는 회사 웹사이트로 연결되는 링크) • 예상 월별 볼륨 • 명확하지 않은 경우 회사 이름과 발신자 ID 간의 연관성에 대한 설명 제공 • 사용 사례, 메시지 목적 설명 • 회사의 공식 사업자/무역 허가증 번호 또는 VAT # • 사용 사례가 이 발신자 ID에서 전송되는 모든 트래픽을 설명한다는 서면 확인서 • 보내려는 메시지 템플릿

국가 이름	메시지 유형	형식 제한 및 요구 사항	등록 요구 사항
베트남(VN)	트랜잭션 메시지 전용. 마케팅 및 프로모션 메시지는 허용되지 않습니다. 금지된 콘텐츠는 다음과 같습니다. • 성인용 콘텐츠 • 자선 서비스 • Cryptocurrency • 복권 • 모바일 도박/카지노 • 스포츠 베팅 • 투표	• 최대 11자 • 공백 금지 • 특수 문자 금지	• 등록하려는 발신자 ID • 사용자는 어느 AWS 리전에서 API/서비스를 호출할지 • 예상 월별 볼륨 • 명확하지 않은 경우 회사 이름과 발신자 ID 간의 연관성에 대한 설명 제공 • 사용 사례, 메시지 목적 설명 • 사용 사례가 이 발신자 ID에서 전송되는 모든 트래픽을 설명한다는 서면 확인서

서명 형식 제한 및 요구 사항

중국으로 성공적으로 전송하려면 메시지 콘텐츠 템플릿과 함께 메시지 서명을 등록해야 합니다. 이 메시지 서명은 전송된 각 SMS의 메시지 내용 앞에 추가해야 합니다. 등록된 서명을 SMS 메시지 본문 앞에 추가하지 않으면 SMS가 차단되거나 필터링될 수 있습니다.

- 서명은 대괄호에 넣어 SMS 메시지 본문 앞에 추가해야 함
- 표준 텍스트는 대괄호 안에 포함해야 함
- 유니코드 텍스트는 서명을 포함하려면 렌티큘러 브래킷(렌즈 모양 괄호)를 사용해야 함 - U+3010 왼쪽 렌티큘러 브래킷 및 U+3011 오른쪽 렌티큘러 브래킷 - 예: 【알림】
- 3~11자여야 함
- 중국어/영어 문자가 지원됨

프랑스 발신자 ID 요구 사항

이 가이드에서는 프랑스 이동통신사에서 프랑스로 SMS 문자 메시지를 보내는 데 요구되는 전용 발신자 ID를 만드는 데 필요한 단계와 지침을 제공합니다.

주제

- 프랑스 전용 발신자 ID 설정
- 발신자 ID 명명 지침

프랑스 전용 발신자 ID 설정

다음 방법 중 하나를 사용하여 전용 발신자 ID를 설정할 수 있습니다. Amazon SNS는 `Publish` API를 사용하여 게시된 SMS 메시지에 사용자를 대신하여 발신자 ID를 사용합니다.

- Amazon SNS 콘솔을 사용하여 게시된 모든 SMS 메시지에 사용할 기본 발신자 ID를 구성할 수 있습니다. 자세히 알아보려면 를 사용하여 SMS 메시징 기본 설정 지정 AWS Management Console 페이지를 방문하세요.
- SMS 메시지를 게시하도록 Amazon SNS에 요청할 때 `Publish` API를 사용하여 `AWS.SNS.SMS.SenderID` 메시지 속성을 포함하여 발신자 ID를 설정할 수 있습니다. 자세히 알아보려면 메시지 전송(콘솔) 페이지를 방문하세요.

발신자 ID 명명 지침

- 발신자 ID 이름은 최대 11자의 영숫자여야 합니다.
- 발신자 ID 이름에 특수 문자나 공백을 사용해서는 안 됩니다.
- 발신자 ID와 SMS 문자 메시지를 보내는 회사의 브랜드 명과 같은 이름을 사용합니다.

인도 발신자 ID 등록 요구 사항

인도에 있는 수신자에게 메시지를 보낼 때 Amazon SNS는 기본적으로 국제 장거리 교환원(ILDO) 연결을 사용하여 이러한 메시지를 전송합니다. 수신자가 ILDO 연결을 통해 보낸 메시지를 볼 때 임의의 숫자 ID에서 보낸 것으로 나타납니다(전용 단축 코드를 구매한 경우는 제외).

> **Note**
>
> 로컬 경로를 사용한 메시지 전송 요금은 Amazon SNS 전 세계 SMS 요금 페이지에 나와 있습니다. ILDO 연결을 통한 메시지 전송 요금은 로컬 경로를 통한 메시지 전송 요금보다 높습니다.

SMS 메시지에 영문자로 구성된 발신자 ID를 사용하려면 ILDO 경로가 아닌 로컬 경로를 통해 메시지를 보내야 합니다. 로컬 경로를 사용하여 메시지를 보내려면 먼저 DLT(분산 원장 기술) 포털을 통해 사용 사례 및 메시지 템플릿을 인도의 통신 규제 기관(TRAI)에 등록해야 합니다. 이러한 등록 요구 사항은 인도의 소비자가 수신하는 원치 않는 메시지 수를 줄이고 잠재적으로 유해한 메시지로부터 소비자를 보호하기 위해 고안되었습니다. 이러한 등록 절차는 Vilpower 서비스를 통해 Vodafone India에 의해 관리됩니다.

주제

- 1단계: TRAI에 등록
- 2단계: 발신자 ID 요청
- 3단계: SMS 메시지 전송
- 인도의 수신자에게 전송된 SMS 메시지 문제 해결

1단계: TRAI에 등록

인도의 수신자에게 SMS 메시지를 보내려면 먼저 조직을 TRAI(Telecom Regulatory Authority of India)에 등록해야 합니다. 등록 과정에서 다음 정보를 제공할 준비를 하세요.

- 조직의 영구 계정 번호(PAN)
- 조직의 세금 공제 계정 번호(TAN)
- 조직의 상품 및 서비스 세금 식별 번호(GSTIN)
- 조직의 기업 자격 증명 번호(CIN)
- 조직을 등록할 수 있는 권한을 부여받는 승인 문서(LOA).

다음은 TRAI에 조직을 등록(요금이 부과될 수 있음)하는 데 사용할 수 있는 몇 가지 분산 원장 기술(DLT) 등록 사이트의 샘플 목록입니다. 등록 절차는 사이트에 따라 다릅니다. 지원이 필요한 경우 해당 지원 팀에 문의하세요.

- BSNL DLT - 등록이 무료입니다.
- Jio Trueconnect - 등록 프로세스를 완료하는 데 수수료를 청구하고 있습니다.
- Smart Enterprise Solutions - 등록 절차를 완료하는 데 수수료를 부과합니다.
- Vilpower - 필요에 맞게 다운로드하고 수정할 수 있는 템플릿이 포함되어 있습니다. Vilpower는 등록 프로세스를 완료하는 데 수수료를 청구하고 있습니다.

조직을 TRAI에 등록하려면

Vilpower를 사용하여 조직을 TRAI에 등록하는 방법은 다음과 같습니다.

1. 웹 브라우저에서 Vilpower 웹 사이트(https://www.vilpower.in)로 이동합니다.
2. [Signup]을 선택하여 다른 계정을 만듭니다. 등록 프로세스에서 다음을 수행합니다.
   - 등록할 엔터티 유형으로 As Enterprise(엔터프라이즈 형태)를 선택합니다.
   - 텔레마케터 이름에는 Infobip Private Limited - ALL을 사용합니다. 메시지가 표시되면 **`Infobip`** 입력을 시작한 다음 드롭다운 목록에서 Infobip Private Limited – ALL을 선택합니다.
   - [Enter Telemarketer ID]에서 **`110200001152`**를 입력합니다.
   - 헤더 ID를 제공하라는 메시지가 표시되면 등록할 발신자 ID를 입력합니다.

     > **Note**
     >
     > 인도에서는 발신자 ID의 길이가 정확히 6자여야 합니다.

   - 콘텐츠 템플릿을 제공하라는 메시지가 표시되면 수신자에게 보낼 메시지 콘텐츠를 입력합니다. 보내려는 모든 메시지에 대해 템플릿을 포함시킵니다.

> **Note**
>
> DLT 등록 공급자 웹 사이트는 Amazon Web Services Services에 의해 관리되지 않습니다. 웹 사이트의 단계는 변경될 수 있습니다.

2단계: 발신자 ID 요청

인도에서 발신자 ID를 요청하려면 AWS Support 요청을 제출해야 합니다. 발신자 ID 요청의 단계를 수행합니다. 요청 시 다음 필수 정보를 제공합니다.

- 발신자가 SMS 메시지를 보낼 AWS 리전입니다.
- DLT 등록 프로세스 중에 사용된 회사 이름입니다.
- DLT 엔터티 등록에 성공한 후 받은 보안 주체 엔터티 ID(PEID)입니다.
- 예상 월별 볼륨.
- 사용 사례에 대한 설명.
- 최종 사용자 옵트인 흐름에 대한 설명입니다.

- 최종 사용자의 옵트인이 수집되고 등록되었다는 확인.

3단계: SMS 메시지 전송

조직을 TRAI에 등록한 후 인도에 있는 수신자에게 SMS 메시지를 보낼 수 있습니다.

1. Amazon SNS 콘솔에 로그인합니다.
2. 콘솔 메뉴에서 리전 선택기를 SMS 메시징을 지원하는 리전으로 설정합니다.
3. 탐색 창에서 문자 메시지(SMS)를 선택합니다.
4. 모바일 문자 메시징(SMS) 페이지에서 문자 메시지 게시를 선택합니다. SMS 메시지 게시 창이 열립니다.
5. [Message type]에서 다음 중 하나를 선택합니다.

   - 프로모션 – 중요하지 않은 메시지입니다(예: 마케팅 메시지).

     숫자 발신자 ID를 사용하는 경우 이 옵션을 선택합니다.
   - 트랜잭션 – 고객 트랜잭션을 지원하는 중요한 메시지입니다(예: 멀티 팩터 인증을 위한 일회용 암호).

     알파벳 또는 영숫자 발신자 ID를 사용하는 경우 이 옵션을 선택합니다.

   이 메시지 수준 설정은 [Text messaging preferences] 페이지에서 설정하는 기본 메시지 유형보다 우선적으로 적용됩니다.

   프로모션 및 트랜잭션 메시지에 대한 요금 정보는 글로벌 SMS 요금에서 확인하세요.
6. 번호에 메시지를 전송하려는 전화번호를 입력합니다.
7. 메시지에 전송할 메시지를 입력합니다.

   SMS 메시지에 내용을 추가할 때 DLT 등록 템플릿의 내용과 정확히 일치하는지 확인하세요. 통신 사업자는 메시지 내용에 추가 문자 반환, 공백, 구두점 또는 일치하지 않는 문장이 포함된 경우 SMS 메시지를 차단합니다. 템플릿의 변수는 30자 이하일 수 있습니다.
8. 발신 ID 섹션에서 발신자 ID에 3~11자로 구성된 사용자 지정 ID를 입력합니다.

   발신자 ID는 프로모션 메시지의 경우 숫자로, 트랜잭션 메시지의 경우 알파벳 또는 영숫자를 사용할 수 있습니다. 발신자 ID는 수신 디바이스에 메시지 발신자로 표시됩니다.

인도에 등록된 숫자 프로모션 발신자 ID의 경우 SMS 전송 요청에 발신자 ID를 발신 번호 파라미터로 지정합니다.

9. 국가별 속성 섹션을 확장하고 인도의 수신자에게 SMS 메시지를 보내는 데 필요한 다음 속성을 지정합니다.

   - 엔터티 ID – 인도의 수신자에게 SMS 메시지를 보내기 위해 규제 기관에서 받은 엔터티 ID 또는 보안 주체 엔터티(PE) ID입니다.

     이 ID는 TRAI에 등록한 엔터티를 고유하게 식별하는 1–50자의 사용자 지정 TRAI 제공 문자열입니다.
   - 템플릿 ID – 인도의 수신자에게 SMS 메시지를 보내기 위해 규제 기관에서 받은 템플릿 ID입니다.

     이 ID는 TRAI에 등록한 템플릿을 고유하게 식별하는 1–50자의 사용자 지정 TRAI 제공 문자열입니다. 템플릿 ID는 이전 단계에서 지정한 발신자 ID 및 메시지 내용과 연결되어야 합니다.

10. 메시지 게시를 선택합니다.

다른 국가의 수신자에게 SMS 메시지를 보내는 방법에 대한 자세한 내용은 휴대폰에 게시에서 확인하세요.

인도의 수신자에게 전송된 SMS 메시지 문제 해결

다음은 통신 사업자에서 SMS 메시지를 차단할 수 있는 몇 가지 이유입니다.

- 보낸 콘텐츠와 일치하는 템플릿을 찾을 수 없습니다.

  전송된 컨텐츠: **<#> 12345 is your OTP to verify mobile number. Your OTP is valid for 15 minutes -- ABC Pvt. Ltd.**

  일치하는 템플릿: 없음

  문제: DLT 등록 템플릿의 시작 부분에 <#> 또는 {#var#}이(가) 포함된 DLT 템플릿이 없습니다.
- 변수 값이 30자를 초과합니다.

  전송된 컨텐츠: **12345 is your OTP code for ABC (ABC Company - India Private Limited) - (ABC 123456789). Share with your agent only. - ABC Pvt. Ltd.**

  일치하는 템플릿: **{#var#} is your OTP code for {#var#} ({#var#}) - ({#var#} {#var#}). Share with your agent only. - ABC Pvt. Ltd.**

문제: 전송된 콘텐츠의 "ABC Company - India Private Limited" 값이 단일 `{#var#}`자 제한인 30자를 초과합니다.

- 메시지 문장의 대소문자가 템플릿의 대소문자와 일치하지 않습니다.

  전송된 컨텐츠: **`12345 is your OTP code for ABC (ABC Company - India Private Limited) - (ABC 123456789). Share with your agent only. - ABC Pvt. Ltd.`**

  일치하는 템플릿: **`{#var#} is your OTP code for {#var#} ({#var#}) - ({#var#} {#var#}). Share with your agent only. - ABC PVT. LTD.`**

  문제: DLT 일치 템플릿에 추가된 회사 이름은 대문자로 표시되지만 전송된 콘텐츠는 이름의 일부를 소문자로 변경했습니다(“ABC Pvt. Ltd.” 과 “ABC PVT.) LTD.”

싱가포르 발신자 ID 등록 요구 사항

Amazon SNS 고객은 싱가포르 SMS 발신자 ID 레지스트리(SSIR)를 통해 등록된 발신자 ID를 사용하여 싱가포르에서 SMS 트래픽을 보낼 수 있습니다. SSIR은 싱가포르의 정보통신 미디어 개발 기관(IMDA)이 소유한 싱가포르 네트워크 정보 센터(SGNIC)를 통해 2022년 3월에 개시되었으며, 이를 통해 조직에서 싱가포르의 휴대폰으로 SMS를 보내는 경우 발신자 ID를 등록할 수 있습니다.

등록된 싱가포르 발신자 ID를 사용하려면 고유 법인 번호(UEN)를 받은 다음 Amazon에 요청을 제출하여 발신자 ID를 사용하도록 계정을 허용 목록에 추가하고, 최종적으로 SSIR을 통해 등록 프로세스를 완료해야 합니다.

2023년 1월 30일까지 ID를 등록하지 않으면 해당 발신자 ID를 사용하여 보낸 모든 메시지의 ID가 규제 기관 규칙에 따라 LIKELY-SCAM으로 변경됩니다. 해당 날짜 이후에도 규제 기관은 재량에 따라 미등록 트래픽을 계속 필터링하거나 차단할 수 있습니다.

> **Important**
>
> Amazon Pinpoint 리전에서 발신자 ID를 요청하는 경우 Amazon Pinpoint 콘솔을 사용하여 발신자 ID를 등록합니다. Amazon Pinpoint 리전 이외의 리전에 대한 등록 절차를 수동으로 완료하려면 싱가포르 발신자 ID 등록을 사용합니다.
> 싱가포르에서 계속해서 메시지를 보낼 수 있으려면 2023년 1월 30일까지 등록을 완료해야 합니다.
> 등록 단계를 다음 순서로 완료하는 것이 매우 중요합니다. 이러한 단계를 순서대로 수행하지 않으면 서비스에서 발신자 ID가 차단되거나 발신자 ID가 모바일 디바이스에 유지되지 않을 수 있습니다.

단계 1. 싱가포르 고유 법인 번호(UEN) 등록하기
단계 2. Amazon Pinpoint 리전에서 발신자 ID를 요청하는 경우 Amazon Pinpoint 발신자 ID 등록 지침을 사용하여 발신자 ID를 등록합니다.

- 계정이 Amazon Pinpoint 리전에 속하지 않는 곳에서 발신자 ID를 등록하려면 싱가포르 발신자 ID 등록 지침을 사용하여 발신자 ID를 수동으로 등록합니다.
- 다른 회사를 대신하여 SMS 문자 메시지를 보낼 때는 회사의 승인서(LOA)가 필요합니다.
- AWS 발신자 ID 등록을 제출한 후 승인이나 상태 변경을 기다리지 말고 바로 3단계로 이동하세요.

단계 3. 싱가포르 네트워크 정보 센터(SGNIC)에 발신자 ID 등록

주제

싱가포르 고유 법인 번호(UEN) 등록하기

SSIR에 등록을 시작하려면 먼저 싱가포르 고유 법인 번호(UEN)를 받아야 합니다. UEN은 회계 및 법인 규제 기관(ACRA)에 비즈니스를 등록할 때 받는 고유한 법인 번호입니다. 자세한 내용은 Who Must Register with ACRA?(누가 ACRA에 등록해야 하나요?)를 참조하십시오. 처리 시간은 ACRA가 요청을 얼마나 쉽게 검증할 수 있는지에 따라 달라질 수 있습니다.

싱가포르 발신자 ID를 Amazon Pinpoint에 등록하기

싱가포르 고유 법인 번호(UEN)를 등록하고 나면 Amazon Pinpoint 콘솔에서 발신자 ID 등록 프로세스를 완료할 수 있습니다(Amazon Pinpoint 리전에만 해당). 발신자 ID를 등록할 때 누락된 부분이 없고 정확한 정보인지 확인하세요. 그렇지 않으면 등록이 거부될 수 있습니다.

> **Important**
>
> Amazon Pinpoint 콘솔을 통해 제출하는 정보는 등록이 완료될 수 있도록 통신 사업자 파트너에게 전달됩니다.

싱가포르 발신자 ID를 등록하는 방법

계정이 Amazon Pinpoint 리전에 있을 때 이 단계를 사용하여 발신자 ID를 등록하세요. 계정이 Amazon Pinpoint 리전에 있지 않은 경우 싱가포르 발신자 ID 등록을 완료하기 위한 수동 등록 절차의 내용을 참조하세요.

1. AWS 관리 콘솔에 로그인한 후 https://console.aws.amazon.com/pinpoint/에서 Amazon Pinpoint 콘솔을 엽니다.
2. 탐색 창의 SMS 및 음성(SMS and voice)에서 전화번호(Phone numbers)를 선택합니다.
3. Sender ID registrations(발신자 ID 등록) 탭에서 Create registration(등록 생성)을 선택합니다.
4. 목적지 국가로 Singapore(싱가포르)를 선택합니다.
5. Company Information(회사 정보) 섹션에서 다음 정보를 입력합니다.
   - Company Name(회사 이름)에는 UEN 등록에 표시된 것과 정확히 같은 회사 이름을 입력합니다.
   - Tax ID(세금 ID)에는 ACRA로부터 받은 UEN 번호를 입력합니다.
   - 회사 웹 사이트(Company Website)에는 회사 웹 사이트의 전체 URL을 입력합니다.
   - Address 1(주소 1)에는 기업 본사의 상세 주소를 입력합니다.
   - Address 2(주소 2)(선택 사항)에는 필요한 경우 기업 본사의 건물 번호를 입력합니다.
   - City(도시)에는 본사의 도시를 입력합니다.
   - State(주)에는 본사의 주를 입력합니다.
   - Zip Code(우편 번호)에는 본사의 우편 번호를 입력합니다.
   - Country(국가)에는 2자리 IDO 국가 코드를 입력합니다.
6. Contact Information(연락처 정보) 섹션에서 다음 정보를 입력합니다.
   - 이름(First Name)에는 비즈니스 연락처가 될 사람의 이름을 입력합니다.

- 성(Last Name)에는 비즈니스 연락처가 될 사람의 성을 입력합니다.
- Support 이메일(Support Email)에는 비즈니스 연락처가 될 사람의 이메일 주소를 입력합니다.
- Support 전화번호(Support Phone Number)에 비즈니스 연락처가 될 사람의 전화번호를 입력합니다.

7. Sender ID Information(발신자 ID 정보) 섹션에서 다음을 입력합니다.
   - Sender ID(발신자 ID)에는 메시지에 표시할 발신자 ID를 입력합니다.
   - Registering on behalf of another brand/entity?(다른 브랜드/법인을 대신하여 등록하려는 경우입니까?)에 해당하는 경우 True를 선택합니다. 메시지를 보내는 최종 사용자가 아닌 경우 다른 브랜드/법인의 “대표자”로 간주됩니다.
   - Letter of authorization image – optional(위임장 이미지(선택 사항))에는 Registering on behalf of another brand/entity?(다른 브랜드/법인을 대신하여 등록하려는 경우입니까?) 확인란을 선택한 경우 전체 위임장(LOA) 이미지를 업로드합니다. 지원되는 파일 형식은 PNG이며 최대 파일 크기는 400KB입니다. 편의를 위해 LOA 템플릿을 다운로드할 수 있습니다.
   - Sender ID connection – optional(발신자 ID 연결(선택 사항))에서는 요청된 발신자 ID와 회사 이름 간의 연결에 대한 세부 정보를 선택적으로 추가할 수 있습니다.
8. Messaging Use Case(메시징 사용 사례)에서 다음을 수행합니다.
   - 월별 SMS 볼륨(Monthly SMS Volume)은 매월 될 SMS 메시지 수를 선택합니다.
   - Use Case Category(사용 사례 범주)는 전화번호에 대해 다음 사용 사례 유형 중 하나를 선택합니다.
     - Two-factor authentication(이중 인증) - 이중 인증 코드를 보낼 때 사용합니다.
     - One-time passwords(일회용 비밀번호) - 사용자에게 일회용 암호를 보낼 때 사용합니다.
     - 알림(Notifications) - 사용자에게 중요한 알림을 보내려는 경우에만 사용합니다.
     - 폴링 및 설문(Polling and surveys) - 사용자의 기본 설정을 폴링할 때 사용합니다.
     - 온디맨드 정보(Info on demand) - 요청을 보낸 후 사용자에게 메시지를 보낼 때 사용합니다.
     - 프로모션 및 마케팅(Promotions and Marketing) - 사용자에게 마케팅 메시지를 보낼 때만 사용합니다.
     - 기타(Other) - 사용 사례가 다른 범주에 속하지 않는 경우 이 옵션을 사용합니다. 이 옵션의 Use Case Details(사용 사례 세부 정보)를 작성해야 합니다.
   - Use Case Details – optional(사용 사례 세부 정보(선택 사항))을 작성하여 선택한 Use Case Category(사용 사례 범주)에 추가 컨텍스트를 제공합니다.
9. 메시징 샘플(Messaging Samples) 섹션에서는 다음을 수행합니다.

- Message Sample 1(메시지 샘플 1)에는 최종 사용자에게 보낼 SMS 메시지 본문의 예제 메시지를 입력합니다.
- Message Sample 2(메시지 샘플 2)(선택 사항) 및 Message Sample 3(메시지 샘플 3)(선택 사항)에는 필요한 경우 보낼 SMS 메시지 본문의 간단한 예제 메시지를 입력할 수 있습니다.
- 각 Message Sample(메시지 샘플) 텍스트 상자의 최대 문자 수는 306자입니다.

10.마쳤으면 Submit registration(등록 제출)을 선택합니다.

> **Important**
>
> 싱가포르 발신자 ID 등록 상태의 지침에 따라 등록 상태를 확인할 수 있습니다.
> 발신자 ID 등록을 제출한 후 승인이나 상태 변경을 기다리지 말고 바로 싱가포르 네트워크 정보 센터(SGNIC)에 발신자 ID 등록 섹션으로 이동하세요.

싱가포르 발신자 ID 등록을 완료하기 위한 수동 등록 절차

계정이 Amazon Pinpoint 리전에 속해 있지 않을 때 이 단계를 사용하여 발신자 ID를 등록하세요. 계정이 Amazon Pinpoint 리전에 있는 경우 싱가포르 발신자 ID를 Amazon Pinpoint에 등록하기의 내용을 참조하세요.

1. Singapore_Sender_ID_Registration_LOA_Template.zip을 다운로드하고 필요한 정보를 입력합니다.
2. AWS Support에서 사례를 생성합니다.
3. Open support case(지원 사례 열기) 탭에서 Creat case(사례 생성)를 선택합니다.
4. Looking for service limit increases(서비스 한도 증가 찾기)를 선택하고 한도 유형으로 SNS Text Messaging(SNS 문자 메시지)을 선택합니다.
5. Resource Type(리소스 유형)에서 Sender ID Registration(발신자 ID 등록)을 선택합니다.
6. LOA 문서를 첨부하고 요청을 제출합니다.

싱가포르 네트워크 정보 센터(SGNIC)에 발신자 ID 등록

> **Warning**
>
> 이러한 단계를 순서대로 수행하지 않으면 서비스에서 발신자 ID가 차단되거나 발신자 ID가 모바일 디바이스에 유지되지 않을 수 있습니다.

1. 먼저 계정의 싱가포르(SG) 발신자 ID를 AWS에 등록해야 합니다(Amazon Pinpoint 콘솔을 통해 등록 또는 Amazon Pinpoint 외 리전의 경우 수동 등록). 이 단계가 완료되면 다음 단계로 진행할 수 있습니다.
2. SGNIC SMS Sender ID Registry(SGNIC SMS 발신자 ID 레지스트리)의 프로세스를 따라 SGNIC를 통해 발신자 ID를 등록합니다.
   - 프로세스를 완료할 때는 다음을 모두 참여 집계자로 나열해야 합니다.
     - AMCS SG Private Limited(Amazon Media 커뮤니케이션 서비스)
     - Nexmo PTE LTD
     - Sinch Singapore PTE LTD
     - Telesign Singapore PTE LTD
     - Twilio Singapore PTD LTD

> **Note**
>
> 발신자 ID를 사용해야 하는 각 개별 AWS 계정에서 발신자 ID 등록을 제출해야 합니다.

싱가포르 발신자 ID 등록 상태

Amazon SNS 싱가포르 발신자 ID를 등록하면 해당 등록은 다음과 같은 다섯 가지 상태 중 하나가 됩니다.

- Created(생성됨) - 등록이 생성되었지만 제출되지는 않았습니다.
- Submitted(제출됨) - 등록이 제출되었으며 확인 중입니다.
- 검토 중(Reviewing) - 등록이 승인되었으며 검토 중입니다. 1~3주가 소요될 수 있으며 경우에 따라 검토가 완료되는 데 더 오래 걸릴 수 있습니다.
- Complete(완료) - 등록이 승인되었으며 발신자 ID를 사용하여 메시지를 보낼 수 있습니다.
- Requires Updates(업데이트 필요) - 등록을 수정하고 다시 제출해야 합니다. 자세한 정보는 싱가포르 발신자 ID 등록 편집 섹션을 참조하세요. 업데이트가 필요한 필드에는 경고 아이콘과 문제에 대한 간단한 설명이 표시됩니다.

Amazon Pinpoint 리전을 제외한 모든 리전의 경우 AWS Support 부서에서 등록 확인 이메일을 보내거나 AWS Support 사례를 생성합니다.

- Open support case(지원 사례 열기) 탭에서 Creat case(사례 생성)를 선택합니다.

- 서비스 한도 증가(Service Limit increase)를 선택합니다.
- 리소스 유형에서 Sender ID Registration(발신자 ID 등록)을 선택하고 한도에 대해 General Inquiry(일반 문의)를 선택합니다.

등록 상태 확인

1. AWS 관리 콘솔에 로그인한 후 https://console.aws.amazon.com/pinpoint/에서 Amazon Pinpoint 콘솔을 엽니다.
2. 탐색 창의 SMS 및 음성(SMS and voice)에서 전화번호(Phone numbers)를 선택합니다.
3. Sender ID Registrations(발신자 ID 등록) 탭에서 Sender ID(발신자 ID)를 선택합니다.
4. 각 Sender ID(발신자 ID) 등록 상태를 확인할 수 있습니다.

싱가포르 발신자 ID 등록 편집

Amazon Pinpoint 등록에 문제가 있는 경우 등록을 제출한 후 등록 상태가 Requires Updates(업데이트 필요)로 표시됩니다. 이 상태에서는 등록 양식을 편집할 수 있습니다. 업데이트가 필요한 필드에는 경고 아이콘과 문제에 대한 간단한 설명이 있습니다.

발신자 ID를 편집하려면

1. https://console.aws.amazon.com/pinpoint/에서 Amazon Pinpoint 콘솔을 엽니다.
2. 탐색 창의 SMS 및 음성(SMS and voice)에서 전화번호(Phone numbers)를 선택합니다.
3. Sender ID Registration(발신자 ID 등록) 탭에서 편집하려는 번호를 선택하고 Registration ID(등록 ID)를 선택합니다.
4. Update registration(등록 업데이트)을 선택하여 양식을 편집하고 경고 아이콘이 있는 필드를 수정합니다.
5. 다른 브랜드/법인을 대신하여 등록하는 경우 이전에 제출한 Letter of authorization image – optional(위임장 이미지(선택 사항)) 파일을 다시 업로드해야 합니다.
6.

> **Important**
>
> 모든 필드를 다시 확인하여 올바른지 확인합니다.

7. 마쳤으면 Submit registration(등록 제출)을 선택하여 다시 제출합니다.

싱가포르 발신자 ID 등록 삭제

싱가포르 발신자 ID 등록을 더 이상 진행하지 않으려면 등록을 삭제할 수 있습니다. 등록은 상태가 Created(생성됨) 또는 Requires Updates(업데이트 필요함)인 경우에만 삭제할 수 있습니다.

등록을 삭제하려면

1. https://console.aws.amazon.com/pinpoint/에서 Amazon Pinpoint 콘솔을 엽니다.
2. 탐색 창의 SMS 및 음성(SMS and voice)에서 전화번호(Phone numbers)를 선택합니다.
3. Sender ID(발신자 ID) 탭에서 삭제하려는 등록 ID를 선택한 다음 Delete Registration(등록 삭제)을 선택합니다.

싱가포르 등록 문제

Amazon Pinpoint에서 싱가포르 발신자 ID를 수락하지 않는 경우 등록 거부 사유를 설명하는 메시지가 표시됩니다. 이러한 거부에 대해 모범 사례에 나와 있지 않은 의문 사항이 있는 경우 지원 팀에 요청을 제출할 수 있습니다.

거부된 싱가포르 발신자 ID 관련 정보 요청을 제출하려면

1. https://console.aws.amazon.com/pinpoint/에서 Amazon Pinpoint 콘솔을 엽니다.
2. 지원 메뉴를 선택한 다음 지원 센터를 선택합니다.
3. 지원 페이지에서 사례 생성을 선택합니다.
4. 사례 유형(Case type)으로 서비스 한도 증가(Service limit increase)를 선택합니다.
5. Limit type(한도 유형)에서 Pinpoint SMS를 선택합니다.
6. 리소스 섹션에서 다음과 같이 하십시오.
   - Resource Type(리소스 유형)에서 Sender ID Registation(발신자 ID 등록)을 선택합니다.
   - Limit(한도)에서 Registration Rejection Query(등록 거부 쿼리)를 선택합니다.
7. Use case description(사용 사례 설명)에 거부된 싱가포르 발신자 ID와 제공된 거부 사유를 입력합니다.
8. Contact options(연락처 옵션)에서 Preferred contact language(기본 설정 연락처 언어)에 대해 AWS Support 팀과 통신할 때 사용할 언어를 선택합니다.
9. Contact Method(연락 방법)에서 AWS Support 팀과 통신할 때 사용할 방법을 선택합니다.
10. Submit(제출)을 선택합니다.

AWS Support 팀이 AWS Support 사례에서 발신자 ID 등록이 거부된 이유에 대한 정보를 제공합니다.

싱가포르 발신자 ID 등록 FAQ

Amazon Pinpoint를 통한 싱가포르 발신자 ID 번호 등록 절차 관련 FAQ

현재 싱가포르 발신자 ID가 있나요?

싱가포르 발신자 ID를 소유하고 있는지 확인하려면

1. https://console.aws.amazon.com/pinpoint/에서 Amazon Pinpoint 콘솔을 엽니다.
2. 탐색 창의 SMS 및 음성(SMS and voice)에서 전화번호(Phone numbers)를 선택합니다.
3. Sender ID Registration(발신자 ID 등록) 탭에서 보려는 발신자 ID를 선택하고 Registration ID(등록 ID)를 선택합니다.

등록은 얼마나 걸리나요?

일반적인 검토에는 1~3주가 소요되지만 경우에 따라 정부 기관에 정보를 확인하는 데 최대 5주 이상 걸릴 수 있습니다.

고유 법인 번호(UEN)란 무엇이며 어떻게 얻을 수 있나요?

UEN은 회계 및 법인 규제 기관(ACRA)에서 발행한 싱가포르 비즈니스 ID입니다. 싱가포르 현지 기업 및 법인은 ACRA를 통해 신청하여 UEN을 받을 수 있습니다. 등록 및 표준 법인 설립 절차를 통과하면 발급됩니다. Bizfile을 통해 ACRA로 UEN을 신청할 수 있습니다.

싱가포르 발신자 ID를 등록해야 합니까?

예. 2023년 1월 30일까지 싱가포르 발신자 ID를 등록하지 않은 경우 발신자 ID를 사용하여 보낸 모든 메시지의 ID는 LIKELY-SCAM으로 변경됩니다.

싱가포르 발신자 ID를 Amazon Pinpoint에 등록하려면 어떻게 해야 합니까?

싱가포르 발신자 ID 등록하기의 지침에 따라 Amazon Pinpoint에 발신자 ID를 등록하십시오.

싱가포르 발신자 ID 등록에는 어떤 상태가 있으며 어떤 의미가 있습니까?

싱가포르 발신자 ID 등록 상태에 있는 지침에 따라 등록 및 상태를 확인할 수 있습니다.

어떤 정보를 제공해야 합니까?

회사 주소, 비즈니스 연락처 및 사용 사례를 제공해야 합니다. 필요한 정보는 Amazon Pinpoint에 싱가포르 발신자 ID 등록하기에서 확인할 수 있습니다.

싱가포르 발신자 ID가 거부되면 어떻게 됩니까?

등록이 거부되면 상태가 업데이트 필요(Requires Updates)로 변경되며 싱가포르 발신자 ID 등록 편집의 지침에 따라 업데이트할 수 있습니다.

필요한 권한은 무엇입니까?

Amazon Pinpoint 콘솔을 방문하는 데 사용하는 IAM 사용자/역할에는 `'sms-voice:*'` 권한이 있어야 합니다.

발신 번호

발신 번호는 SMS 메시지 발신자의 전화번호를 식별하는 숫자 문자열입니다. 발신 번호를 사용하여 SMS 메시지를 보낼 때 수신자의 디바이스는 발신 번호를 발신자의 전화번호로 표시합니다. 사용 사례에 따라 다른 발신 번호를 지정할 수 있습니다.

> **Tip**
>
> AWS 계정에 있는 모든 기존 발신 번호 목록을 보려면 Amazon SNS 콘솔의 탐색 창에서 발신 번호를 선택합니다.

현지 법률에 따라 발신 번호 대신 발신자 ID를 사용해야 하는 국가에서는 발신 번호에 대한 지원을 사용할 수 없습니다.

주제

- 10DLC
- 수신자 부담 전화번호
- 단축 코드
- 개인 대 개인(P2P) 긴 코드
- 미국 제품 번호 비교

10DLC

미국 통신 사업자는 더 이상 등록되지 않은 로컬 긴 코드를 통한 애플리케이션 대 개인(A2P) SMS 메시징 사용을 지원하지 않습니다. 대용량 A2P SMS 메시징의 경우 미국 통신 사업자는 대신 10DLC(10-digit long code)라는 새로운 유형의 긴 코드를 제공합니다.

Important

2023년 1월 26일부터 Amazon SNS의 SMS 공급업체는 미국 통신사에서 제기한 SMS 스팸 문제를 해결하기 위해 10DLC 캠페인에 새로운 수동 검토 프로세스를 도입했습니다. 10DLC의 대안으로 단축 코드 및 무료 전화 번호를 사용하여 미국에서 SMS를 보낼 수 있습니다.
현재 SMS 공급업체는 10DLC 캠페인 검토에 걸리는 시간에 대한 서비스 수준 목표를 제시하지는 않았습니다. 번호가 10DLC 캠페인에 연결되면 검토가 트리거됩니다. Amazon SNS가 이전에 전달한 예상 시간인 14일보다 검토가 더 오래 걸리고 있습니다.
Amazon SNS는 매일 SMS 공급업체와 협력하여 다음을 보장합니다.

- 공급업체가 대기 중인 10DLC 캠페인 검토를 가능한 한 빨리 완료합니다.
- 공급업체가 백로그에서 AWS 요청의 우선 순위를 지정합니다.

10DLC 캠페인의 안내에 따라 10DLC 캠페인의 상태를 확인할 수 있습니다. 10DLC 캠페인을 승인하는 데 추가 정보가 필요한 경우 AWS 지원 팀에서 알려드립니다.
10DLC 번호를 받는 것보다 더 빨리 미국 무료 전화번호를 등록할 수 있습니다. 미국 무료 전화번호 및 등록 절차에 대한 자세한 내용은 수신자 부담 전화번호 등록 요구 사항 및 프로세스을 참조하세요.

10DLC란 무엇입니까?

10DLC는 10자리 전화번호 형식을 사용하여 대용량 A2P SMS 메시징을 지원하기 위해 통신 사업자에 등록된 긴 코드 유형입니다. Amazon SNS는 더 이상 SMS 제품으로 로컬 긴 코드를 제공하지 않고 대신 10DLC를 제공합니다. 10DLC는 짧은 코드와 수신자 부담 전화번호만 사용하는 경우 영향을 미치지 않습니다.

10DLC는 미국에서만 사용되는 10자리 전화번호입니다. 10DLC에서 수신자에게 보낸 메시지는 발신자로 10자리 숫자를 표시합니다. 수신자 부담 전화번호와 달리 10DLC는 트랜잭션 및 프로모션 메시징을 모두 지원하며 모든 미국 지역 코드를 포함할 수 있습니다.

기존 로컬 긴 코드가 있는 경우 해당 로컬 긴 코드가 10DLC에 대해 사용되도록 요청할 수 있습니다. 이 작업을 수행하려면 10DLC 등록 프로세스를 완료한 다음 지원 티켓을 제출하세요. 10DLC에 대해 긴 코드를 사용하는 데 문제가 있는 경우 Amazon Pinpoint(Amazon SNS 아님) 콘솔을 통해 새 10DLC를 요청하라는 알림과 지시를 받습니다. 긴 코드를 변환하기 위해 지원 티켓을 제출하는 방법에 대한 자세한 내용은 긴 코드와 10DLC 캠페인 연결를 참조하세요.

10DLC 번호를 사용하려면 먼저 회사를 등록하고 Amazon Pinpoint(Amazon SNS 아님) 콘솔을 사용하여 10DLC 캠페인을 생성하세요. AWS는 이 정보를 기반으로 귀하의 등록을 승인하거나 거부하는 서드 파티인 The Campaign Registry와 이 정보를 공유합니다. 경우에 따라 등록이 즉시 이뤄집니다. 예를 들어, 이전에 The Campaign Registry에 등록했다면 이미 해당 정보가 있을 수 있습니다. 그러나 일부 캠페인은 승인에 1주일 이상이 소요될 수 있습니다. 회사 및 10DLC 캠페인이 승인된 후 10DLC 번호를 구입하여 캠페인에 연결할 수 있습니다. 10DLC를 요청하는 작업에도 승인에 최대 일주일이 걸릴 수 있습니다. 여러 10DLC를 단일 캠페인과 연결할 수 있지만 여러 캠페인에서 동일한 10DLC를 사용할 수는 없습니다. 생성하는 각 캠페인에 대해 고유한 10DLC가 있어야 합니다.

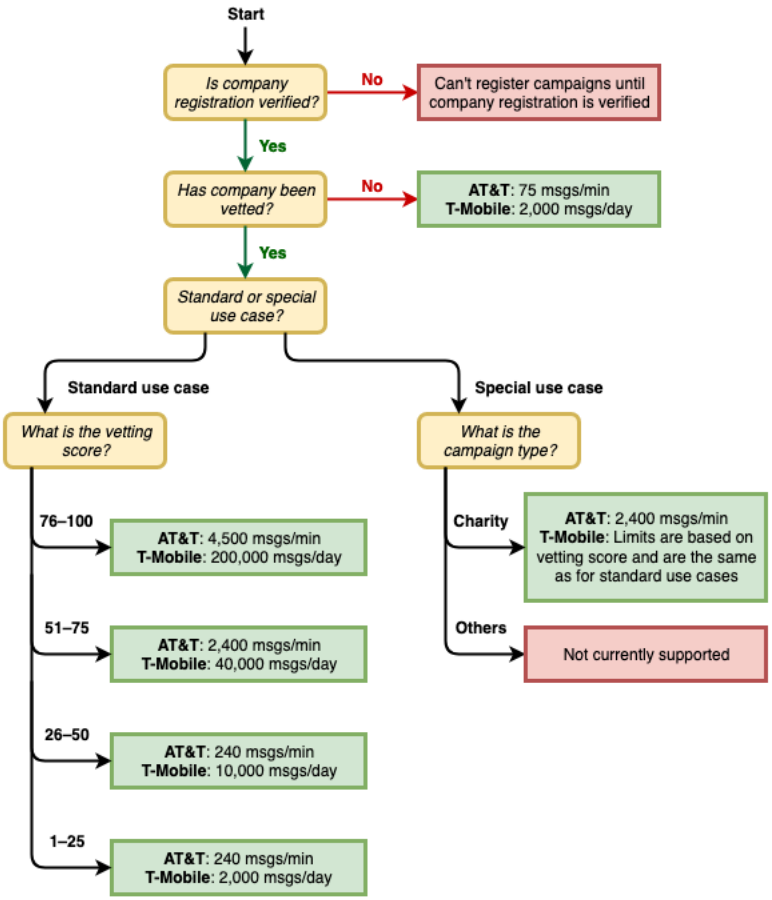
10DLC 기능

10DLC 전화번호의 기능은 수신자가 이용하는 이동 통신사에 따라 달라집니다. AT&T는 각 캠페인에서 분당 보낼 수 있는 메시지 부분의 수에 대한 한도를 제공합니다. T-Mobile은 분당 보낼 수 있는 메시지 부분의 수에 제한 없이 각 회사에서 보낼 수 있는 일일 메시지 한도를 제공합니다. Verizon은 처리량 한도를 공지하지 않았지만, 실제 메시지 처리량에는 그다지 중점을 두지 않고 스팸, 원치 않는 메시지 및 부정 사용 콘텐츠를 제거하도록 설계된 필터링 시스템을 10DLC에 사용합니다.

심사되지 않은 회사에 연결된 새 10DLC 캠페인에서는 AT&T를 이용하는 수신자에게 분당 75개의 메시지 부분을, T-Mobile을 이용하는 수신자에게 일일 2,000개의 메시지를 보낼 수 있습니다. 회사 한도는 회사의 모든 10DLC 캠페인에서 공유됩니다. 예를 들어 한 회사와 두 개의 캠페인을 등록한 경우 T-Mobile 고객에게 보낼 수 있는 하루 2,000개의 메시지 할당이 전체 캠페인에서 공유됩니다. 마찬가지로 두 개 이상의 AWS 계정에서 동일한 회사를 등록하는 경우 일일 할당이 전체 계정에서 공유됩니다.

처리량이 이 한도를 초과하는 경우 회사 등록에 대한 심사를 요청할 수 있습니다. 회사 등록을 심사하는 경우 서드 파티 확인 공급자가 회사 세부 정보를 분석합니다. 그런 다음 확인 공급자는 10DLC 캠페인의 기능을 결정하는 심사 점수를 제공합니다. 심사 서비스에는 일회성 요금이 부과됩니다. 자세한 내용은 Amazon SNS 10DLC 등록 심사 섹션을 참조하세요.

실제 처리율은 회사의 심사 여부, 캠페인 유형 및 심사 점수와 같은 다양한 요인에 따라 달라집니다. 다음 흐름 차트에서는 다양한 상황에 따른 처리율을 보여줍니다.



10DLC의 처리율은 Campaign Registry와 협력하여 미국 이동 통신사가 결정합니다. Amazon SNS 및 기타 SMS 전송 서비스는 이러한 처리율 이상으로 10DLC 처리량을 늘릴 수 없습니다. 모든 미국 이동 통신사에서 높은 처리율과 높은 배송률이 필요한 경우 단축 코드를 사용하는 것이 좋습니다. 단축 코드를 받는 방법에 대한 자세한 내용은 Amazon SNS에서 SMS 메시징에 대한 전용 단축 코드 요청 단원을 참조하세요.

10DLC 시작하기

Amazon Pinpoint 콘솔(Amazon SNS 아님)을 사용하여 10DLC를 요청하세요. 다음 단계에 따라 10DLC 캠페인에 사용할 10DLC를 설정하세요.

1. 회사를 등록합니다.

   10DLC를 요청하려면 먼저 The Campaign Registry에 회사를 등록해야 합니다. 자세한 내용은 회사 등록에서 확인하세요. The Campaign Registry에 추가 정보가 필요하지 않는 한 등록은 일반적으로 즉시 이루어집니다. 해당 회사를 등록하기 위한 일회성 등록 요금이 등록 페이지에 표시됩니다. 이 일회성 요금은 캠페인 및 10DLC에 대한 월별 요금과 별도로 지불됩니다.

> **Note**
>
> Amazon SNS SMS 메시징은 현재 Amazon Pinpoint가 지원되지 않는 리전에서 사용할 수 있습니다. 두 가지 사례가 있습니다.
>
> a. 상용 클라우드 계정을 사용하는 경우 미국 동부(버지니아 북부) 리전에서 Amazon Pinpoint 콘솔을 열어 10DLC 회사 및 캠페인을 등록해야 합니다. 10DLC 번호는 요청하지 마세요.
>
> b. 대신 AWS Service Quotas 콘솔을 사용하여 해당 리전의 10DLC 번호를 요청하는 동안 서비스 한도 증가 사례를 생성합니다. Amazon Pinpoint를 사용할 수 있는 리전에 대한 자세한 내용은 AWS 일반 참조의 Amazon Pinpoint 엔드포인트 및 할당량을 참조하세요.
>
> c. AWS GovCloud (US) 계정을 사용하는 경우 미국 서부 리전에서 Amazon Pinpoint 콘솔을 열어 10DLC 회사 및 캠페인을 등록합니다. 10DLC 번호는 요청하지 마세요. 대신 AWS Service Quotas 콘솔을 사용하여 해당 리전의 10DLC 번호를 요청하는 동안 서비스 한도 증가 사례를 생성합니다. Amazon Pinpoint를 사용할 수 있는 리전에 대한 자세한 내용은 AWS 일반 참조의 Amazon Pinpoint 엔드포인트 및 할당량을 참조하세요.

2. (선택 사항이지만 권장) 심사 신청

   회사 등록에 성공한 경우 소량 혼합 사용 10DLC 캠페인을 생성하기 시작할 수 있습니다. 이러한 캠페인에서 AT&T를 이용하는 수신자에게 분당 75개의 메시지를 보낼 수 있으며, 등록된 회사에서 T-Mobile을 이용하는 수신자에게 일일 2,000개의 메시지를 보낼 수 있습니다. 사용 사례에 이러한 값을 초과하는 처리율이 필요한 경우 회사 등록에 대한 심사를 신청할 수 있습니다. 회사 등록을 심사하는 경우 회사 및 캠페인의 처리율이 증가할 수 있지만 증가한다는 것이 보장되지는 않습니다. 심사에 대한 자세한 내용은 Amazon SNS 10DLC 등록 심사 단원을 참조하세요.

3. 캠페인을 등록합니다.

   회사를 등록한 후 10DLC 캠페인을 만들고 등록된 회사 중 하나와 연결하십시오. 이 캠페인은 승인을 위해 The Campaign Registry에 제출됩니다. The Campaign Registry에서 추가 정보를 요구하지 않는 한 대부분의 경우 10DLC 캠페인 승인은 즉시 이루어집니다. 자세한 내용은 10DLC 캠페인 등록 섹션을 참조하세요.

4. 10DLC 번호를 요청합니다.

   10DLC 캠페인이 승인된 후 10DLC를 요청하고 해당 번호를 승인된 캠페인과 연결할 수 있습니다. 10DLC 캠페인은 승인된 숫자만 사용할 수 있습니다. Amazon SNS를 통한 SMS 메시징을 위한 10DLC 번호, 수신자 부담 전화번호 및 P2P 긴 코드 요청 섹션을 참조하세요.

10DLC 등록 및 월별 요금

회사 등록 및 10DLC 캠페인과 같이 10DLC 사용과 관련된 등록 및 월별 요금이 있습니다. 이 요금들은 AWS에서 청구하는 다른 월별 요금과 별도입니다. 자세한 정보는 Amazon SNS 전 세계 SMS 요금 페이지를 참조하세요.

회사 등록

10DLC를 요청하려면 먼저 The Campaign Registry에 회사를 등록해야 합니다.

> **Note**
>
> Amazon SNS SMS 메시징은 현재 Amazon Pinpoint가 지원되지 않는 리전에서 사용할 수 있습니다. 이러한 경우 미국 동부(버지니아 북부) 리전에서 Amazon Pinpoint 콘솔을 열어 10DLC 회사 및 캠페인을 등록하되 10DLC 번호는 요청하지 마십시오. 대신 AWS Service Quotas 콘솔을 사용하여 해당 리전의 10DLC 번호를 요청하는 동안 서비스 한도 증가 사례를 생성합니다. Amazon Pinpoint를 사용할 수 있는 리전에 대한 자세한 내용은 AWS 일반 참조의 Amazon Pinpoint 엔드포인트 및 할당량을 참조하세요.

10DLC 회사 등록 상태

회사 또는 브랜드를 등록하면 확인되지 않음(Unverified) 또는 확인됨(Verified)이라는 두 가지 상태 중 하나가 반환됩니다. 회사 등록 상태가 확인되지 않음(Unverified)인 경우 등록에 문제가 있음을 의미합니다. 예를 들어 제공한 등록 회사 이름이 제공한 세금 ID에 연결된 회사의 등록된 이름과 정확히 일치하지 않을 수 있습니다. 회사 등록 세부 정보에 문제가 있는 경우 정보를 수정할 수 있습니다. 회사 등록 세부 정보 수정에 대한 자세한 내용은 등록된 회사 편집 또는 삭제 단원을 참조하세요.

회사 등록 상태가 확인됨(Verified)이면 제공한 등록 세부 정보가 정확한 것이며 10DLC 캠페인 생성을 시작할 수 있습니다.

회사 또는 브랜드 등록

회사를 한 번만 등록하면 됩니다. 등록한 후 회사 및 연락처 정보를 편집할 수 있습니다. 등록된 회사를 삭제하려면 AWS Support에 사례를 생성합니다. 회사 세부 정보 편집 또는 삭제에 대한 자세한 내용은 등록된 회사 편집 또는 삭제 단원을 참조하세요.

회사를 등록하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/pinpoint/에서 Amazon Pinpoint 콘솔을 엽니다.

2. 탐색 창의 SMS 및 음성(SMS and voice)에서 전화번호(Phone numbers)를 선택합니다.
3. 10DLC 캠페인(10DLC campaigns) 탭에서 회사 등록(Register company)을 선택합니다.

> **Note**
>
> 회사 등록(Register your company) 페이지에 등록 요금(Registration fee)이 표시됩니다. 회사 등록과 관련된 일회성 요금입니다. 이 비용은 다른 월별 비용 또는 요금과 별개입니다. 회사를 등록하거나 기존 회사 등록의 세부 정보를 수정하면 요금이 부과됩니다.

4. 회사 정보(Company info) 섹션에서 다음을 수행합니다.

- 법적 회사 이름(Legal company name)에 회사의 등록된 이름을 입력합니다. 입력하는 이름은 제공하는 세금 ID에 연결된 회사 이름과 정확히 일치해야 합니다.

> **Important**
>
> 회사의 정확한 법적 이름을 사용해야 합니다. 이 정보를 제출한 후에는 변경할 수 없습니다. 올바르지 않거나 불완전한 정보로 인해 등록이 지연되거나 거부될 수 있습니다.

- 이 조직의 법적 형식 유형은 무엇입니까(What type of legal form is this organization)에서 회사를 가장 잘 설명하는 옵션을 선택합니다.

> **Note**
>
> 미국 정부(US government)와 비영리(Not-for-profit) 옵션은 미국에 본사를 둔 조직을 등록하는 데만 사용할 수 있습니다. 미국 이외 국가에 본사를 둔 조직은 조직의 실제 법적 형식과 상관없이 비공개 영리(Private for-profit)로 등록해야 합니다.

- 앞 단계에서 공개 영리(Public for profit)를 선택한 경우 회사의 주식 기호와 회사가 상장된 증권 거래소를 입력합니다.
- 등록 국가(Country of registration)에서 회사가 등록된 국가를 선택합니다.
- DBA(Doing Business As) 또는 브랜드 이름(Doing Business As (DBA) or brand name)에 회사에서 비즈니스 운영에 사용하는 다른 이름을 입력합니다.

- 세금 ID(Tax ID)에 회사의 세금 ID를 입력합니다. 입력하는 ID는 회사가 등록된 국가에 따라 달라집니다.
  - IRS 고용주 식별 번호(EIN)가 있는 미국 또는 미국 외 법인을 등록하는 경우 9자리 EIN을 입력합니다. 입력하는 법적 회사 이름, EIN 및 실제 주소는 모두 IRS에 등록된 회사 정보와 일치해야 합니다.
  - 캐나다 법인을 등록하는 경우 연방 또는 지방 법인 번호를 입력합니다. CRA에서 제공하는 사업자 번호(BN)를 입력하지 않도록 합니다. 입력하는 법적 회사 이름, 법인 번호 및 실제 주소는 모두 Corporations Canada에 등록된 회사 정보와 일치해야 합니다.
  - 다른 국가에 본사를 둔 법인을 등록하는 경우 해당 국가의 기본 세금 ID를 입력합니다. 많은 국가에서 이는 VAT ID 번호의 숫자 부분입니다.
- 업종(Vertical)에서 등록하려는 회사를 가장 잘 설명하는 범주를 선택합니다.

5. 연락처 정보(Contact info) 섹션에서 다음을 수행합니다.

- 주소/거리(Address/Street)에 회사에 연결된 실제 거리 주소를 입력합니다.
- 구/군/시(City)에 실제 주소지 도시를 입력합니다.
- 시/도 또는 리전(State or region)에 주소지 시/도 또는 리전을 입력합니다.
- 우편 번호(Zip Code/Postal Code)에 주소지 우편 번호를 입력합니다.
- 회사 웹 사이트(Company website)에 회사 웹 사이트의 전체 URL을 입력합니다. 주소 시작 부분에 ‘http://’ 또는 ‘https://’를 포함합니다.
- 지원 이메일(Support email)에 이메일 주소를 입력합니다.
- 지원 전화번호(Support phone number)에 국가 번호를 포함한 전화번호를 입력합니다.

> **Note**
>
> Campaign Registry에서는 회사 대표에게 등록 정보를 확인해야 할 경우에 대비해 연락처 이메일 주소와 전화번호를 요구합니다.

6. 마쳤으면 [Create]를 선택합니다. 회사 등록이 Campaign Registry에 제출됩니다. 대부분의 경우 등록이 즉시 수락되고 상태가 제공됩니다.

회사 등록 상태가 확인됨(Verified)인 경우 소량 혼합 사용 10DLC 캠페인을 생성하기 시작할 수 있습니다. 이 캠페인 유형으로 AT&T를 이용하는 수신자에게 분당 최대 75개의 메시지를 보낼 수 있으며, 등록된 회사에서 T-Mobile을 이용하는 수신자에게 하루에 2,000개의 메시지를 보낼 수 있습니다. Verizon 및 US Cellular 등 다른 미국 통신 사업자를 이용하는 수신자에게도 메시지를 보낼 수 있습니

다. 이러한 통신 사업자는 처리량 한도를 엄격하게 적용하지는 않지만 스팸 및 부정 사용 징후가 있는지 10DLC 메시지를 집중 모니터링합니다.

사용 사례에 이러한 값을 초과하는 처리율이 필요한 경우 회사 등록에 대한 추가 심사를 신청할 수 있습니다. 브랜드 등록 심사에 대한 자세한 내용은 Amazon SNS 10DLC 등록 심사 단원을 참조하세요.

회사 등록 상태가 확인되지 않음(Unverified)인 경우 제공한 정보에 문제가 있음을 나타냅니다. 제공한 정보를 확인하고 모든 필드에 올바른 정보가 포함되어 있는지 확인합니다. Amazon Pinpoint 콘솔에서 회사 등록의 일부 정보를 변경할 수 있습니다. 회사 등록 세부 정보 수정에 대한 자세한 내용은 10DLC 회사 등록 편집 단원을 참조하세요.

Amazon SNS 10DLC 등록 심사

회사 등록이 성공했고 더 높은 처리량 기능이 있는 캠페인을 등록하고자 하는 경우 회사 등록의 심사가 필요합니다.

등록을 심사하는 경우 서드 파티 조직이 제공한 회사 세부 정보를 분석하고 심사 점수를 반환합니다. 심사 점수가 높으면 10DLC 회사 및 회사에 연결된 캠페인의 처리율이 높아질 수 있습니다. 그러나 심사를 통해 처리량이 늘어난다는 것이 보장되지는 않습니다.

심사 점수는 소급 적용되지 않습니다. 즉, 10DLC 캠페인을 이미 생성한 후 나중에 회사 등록을 심사한 경우 기존 캠페인에 심사 점수가 자동으로 적용되지 않습니다. 이러한 이유로 10DLC 캠페인을 만들기 전에 회사 또는 브랜드를 심사해야 합니다.

> **Note**
>
> 회사 또는 브랜드 심사 시 환불 불가능한 40 USD의 요금이 부과됩니다.

회사 등록을 심사하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/pinpoint/에서 Amazon Pinpoint 콘솔을 엽니다.
2. 탐색 창의 SMS 및 음성(SMS and voice)에서 전화번호(Phone numbers)를 선택합니다.
3. 10DLC 캠페인(10DLC campaigns) 탭에서 심사할 10DLC 회사(10DLC company)를 선택합니다.
4. 회사 세부 정보 페이지에서 페이지 하단에 있는 심사 신청(Apply for vetting)을 선택합니다.
5. 추가 심사 신청(Apply for additional vetting) 창에서 제출(Submit)을 선택합니다.

미국에 본사를 둔 회사의 경우 일반적으로 심사 절차를 완료하는 데 약 1분 정도 걸립니다. 미국 이외 국가에 본사를 둔 회사의 경우 해당 국가에서 데이터를 얼마나 쉽게 사용할 수 있는지에 따라 심사 절차에 훨씬 더 많은 시간이 걸릴 수 있습니다.

심사 요청을 제출한 후 회사 세부 정보 페이지로 돌아갑니다. 회사 심사 결과(Company vetting results) 섹션에 심사 요청의 상태와 결과가 표시됩니다. 심사 절차가 완료되면 이 표의 점수(Score) 열에 심사 점수가 표시됩니다. 심사 점수에 따라 10DLC 처리량 기능이 결정됩니다. 처리량은 생성하는 캠페인 유형에 따라 달라집니다. 혼합 사용 또는 마케팅 관련 10DLC 캠페인을 생성하는 경우 다른 캠페인 유형에 필요한 것보다 더 높은 심사 점수가 있어야 높은 처리율을 달성할 수 있습니다. 10DLC 전화번호의 기능에 대한 자세한 내용은 10DLC 기능 단원을 참조하세요.

심사 절차를 완료한 후 회사 등록 세부 정보를 변경하는 경우 등록 심사를 다시 요청할 수 있습니다. 회사 등록에서 업종(Vertical)만 변경하는 경우 심사 점수는 변경되지 않습니다. 업종(Vertical) 이외의 세부 정보를 변경하면 심사 결과가 변경될 수 있습니다. 두 경우 모두 일회성 심사 요금이 다시 부과됩니다.

등록된 회사 편집 또는 삭제

Amazon Pinpoint 콘솔에서 직접 회사의 10DLC 등록 정보 중 일부를 편집할 수 있습니다. AWS Support Center에서 사례를 생성하여 10DLC 회사 등록을 삭제할 수도 있습니다.

10DLC 회사 등록 편집

회사에 대한 10DLC 등록 프로세스를 완료한 후 등록 세부 정보를 편집할 수 있습니다.

회사 등록 세부 정보를 편집한 후 오류 메시지가 표시되면 등록에 다른 문제가 있을 수 있습니다. AWS Support에서 티켓을 열어 추가 정보를 요청할 수 있습니다.

회사 등록을 편집하려면

1. https://console.aws.amazon.com/sms-voice/ 에서 AWS SMS 콘솔을 엽니다.
2. Amazon Pinpoint SMS 사용 설명서의 등록 편집 지침을 따르십시오.

10DLC 회사 등록 삭제

회사 등록을 삭제하려면

1. https://console.aws.amazon.com/sms-voice/ 에서 AWS SMS 콘솔을 엽니다.
2. Amazon Pinpoint SMS 사용 설명서의 등록 삭제 지침을 따르십시오.

10DLC 캠페인 등록

10DLC 캠페인을 등록할 때 사용 사례에 대한 설명과 사용하려는 메시지 템플릿을 제공합니다. 10DLC 캠페인을 만들고 등록하려면 먼저 회사를 등록해야 합니다. 회사 등록에 대한 자세한 내용은 회사 등록 단원을 참조하세요.

> **Note**
>
> 회사를 등록하면 Amazon Pinpoint에 확인됨(Verified) 또는 확인되지 않음(Unverified)이라는 두 가지 등록 상태 중 하나가 표시됩니다. 회사 등록 상태가 확인됨(Verified)인 경우에만 10DLC 캠페인 등록 프로세스를 완료할 수 있습니다. 소량의 혼합 사용 캠페인을 만들 수 있습니다.
>
> 확인되지 않음(Unverified) 상태는 대개 회사를 등록할 때 제공한 일부 데이터가 올바르지 않음을 의미합니다. 회사가 이 상태가 있는 동안에는 10DLC 캠페인을 만들 수 없습니다. 회사 등록을 수정하여 회사 등록과 관련된 문제를 해결할 수 있습니다. 10DLC 회사 등록 수정에 대한 자세한 내용은 등록된 회사 편집 또는 삭제 단원을 참조하세요.

이 페이지에서 먼저 10DLC 캠페인을 만들고 있는 회사에 대한 세부 정보를 제공한 다음 캠페인 자체의 사용 사례 세부 정보를 제공합니다. 이 페이지의 정보는 승인을 위해 The Campaign Registry에 제공됩니다.

이 섹션에서는 10DLC 캠페인을 만들 회사를 선택하고 추가 세부 정보를 제공합니다.

10DLC 캠페인을 등록하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/pinpoint/에서 Amazon Pinpoint 콘솔을 엽니다.
2. SMS 및 음성(SMS and voice)에서 전화번호(Phone numbers)를 선택합니다.
3. 10DLC 캠페인(10DLC campaigns) 탭에서 10DLC 캠페인 생성(Create a 10DLC campaign)을 선택합니다.
4. 10DLC 캠페인 생성(Create a 10DLC campaign) 페이지의 캠페인 정보(Campaign info) 섹션에서 다음을 수행합니다.

    a. 회사 이름에서 이 캠페인을 만들 회사를 선택합니다. 회사를 아직 등록하지 않은 경우 먼저 회사를 등록해야 합니다. 회사 등록에 대한 자세한 내용은 회사 등록 단원을 참조하세요.

    b. 10DLC campaign name(10DLC 캠페인 이름)에 캠페인의 이름을 입력합니다.

    c. 업종(Vertical)에 회사를 가장 잘 나타내는 옵션을 선택합니다.

d. 도움말 메시지(Help message)에 고객이 10DLC 전화번호로 'HELP' 키워드를 보낼 경우 수신하게 될 메시지를 입력합니다.

e. 중지 메시지(Stop message)에 고객이 10DLC 전화번호로 'STOP' 키워드를 보낼 경우 수신하게 될 메시지를 입력합니다.

> **Tip**
>
> 고객은 'HELP'라는 단어로 메시지에 회신하여 수신하는 메시지에 대해 자세히 알아볼 수 있습니다. 또한 'STOP'으로 회신하여 메시지 수신을 거부할 수도 있습니다. 미국 이동 통신사에서는 이 두 키워드에 대한 응답을 제공하도록 요구합니다.
> 다음은 미국 이동 통신사의 요구 사항을 준수하는 HELP 응답의 예입니다.
> **`ExampleCorp Account Alerts: For help call 1-888-555-0142 or go to example.com. Msg&data rates may apply. Text STOP to cancel.`**
> 다음은 규정을 준수하는 STOP 응답의 예입니다.
> **`You are unsubscribed from ExampleCorp Account Alerts. No more messages will be sent. Reply HELP for help or call 1-888-555-0142.`**
> 이러한 키워드에 대한 응답은 160자 이내여야 합니다.

5. 캠페인 사용 사례(Campaign use case) 섹션에서 다음을 수행합니다.

a. 사용 사례 유형(Use case type)에서 자선 관련 사용 사례가 있는 경우 특수(Special)를 선택합니다. 그렇지 않은 경우 표준(Standard)을 선택합니다.

b. 사용 사례에서 사전 설정된 사용 사례 목록에서 캠페인과 가장 유사한 사용 사례를 선택합니다. 각 사용 사례에 대한 월별 요금이 사용 사례 이름 옆에 나타납니다.

> **Note**
>
> 10DLC 캠페인 등록에 대한 월별 요금은 각 사용 사례 유형 옆에 표시됩니다. 대부분의 10DLC 캠페인 유형은 월별 요금이 동일합니다. 소량 혼합 사용 사례의 등록 요금은 다른 사용 사례 유형보다 낮습니다. 그러나 소량 혼합 캠페인은 다른 캠페인 유형보다 낮은 처리율을 지원합니다.

c. 샘플 SMS 메시지(Sample SMS message)를 하나 이상 입력합니다. 이 메시지는 고객에게 보내려는 샘플 메시지입니다. 이 10DLC 캠페인에 여러 메시지 템플릿을 사용하려는 경우 해당 템플릿도 포함합니다.

> **Important**
>
> 샘플 메시지에 자리 표시자 텍스트를 사용해서는 안 됩니다. 제공하는 예제 메시지에는 전송하려는 실제 메시지가 최대한 정확하게 반영되어야 합니다.

6. 캠페인 및 콘텐츠 속성(Campaign and content attributes) 섹션에는 캠페인의 특정 기능과 관련하여 예(Yes) 또는 아니요(No)로 답하는 일련의 질문이 포함되어 있습니다. 일부 속성은 필수이므로 기본값을 변경할 수 없습니다.

   해당 캠페인에 맞는 정확한 속성을 선택해야 합니다.

   다음 각 항목이 등록하는 캠페인에 적용되는지 여부를 표시합니다.

   - 구독자 옵트인 - 구독자는 이 캠페인에 대한 메시지를 수신하도록 선택할 수 있습니다.
   - 구독자 옵트아웃 – 구독자는 이 캠페인에 대한 메시지 수신을 거부할 수 있습니다.
   - 구독자 도움말 – 구독자는 HELP 키워드를 보낸 후 메시지 발신자에게 연락할 수 있습니다.
   - 번호 풀링 – 이 10DLC 캠페인은 50개 이상의 전화번호를 사용합니다.
   - 직접 대출 또는 대출 계약 – 캠페인에는 직접 대출 또는 기타 대출 계약에 대한 정보가 포함됩니다.
   - 임베디드 링크 – 이 10DLC 캠페인에는 임베디드 링크가 포함되어 있습니다. TinyUrl 또는 Bit.ly와 같은 일반적인 URL 단축기의 링크는 허용되지 않습니다. 그러나 사용자 지정 도메인을 제공하는 URL 단축기를 사용할 수 있습니다.
   - 임베디드 전화번호(Embedded phone number) - 이 캠페인에는 고객 지원 번호가 아닌 임베디드 전화번호가 포함되어 있습니다.
   - 제휴 마케팅 - 이 10DLC 캠페인에는 제휴 마케팅 정보가 포함되어 있습니다.
   - 연령 제한 콘텐츠 - 이 10DLC 캠페인에는 통신 사업자 및 이동통신 및 인터넷 연결 협회(CTIA) 지침에 정의된 연령 제한 콘텐츠가 포함됩니다.

7. 생성을 선택합니다.

   캠페인에 대한 등록 세부 정보를 제출하면 SMS 및 음성(SMS and voice) 페이지가 열립니다. 캠페인이 제출되었으며 검토 중임을 나타내는 메시지가 나타납니다. 10DLC 캠페인 탭에서 요청 상태를 확인할 수 있습니다. 다음 중 하나인 10DLC 탭에서 등록 상태를 확인할 수 있습니다.

   - 활성 – 해당 10DLC 캠페인이 승인되었습니다. 10DLC 전화번호를 요청하고 해당 번호를 캠페인에 연결할 수 있습니다. 자세한 내용은 Amazon SNS를 통한 SMS 메시징을 위한 10DLC 번호, 수신자 부담 전화번호 및 P2P 긴 코드 요청 섹션을 참조하세요.

- 대기 중(Pending) - 해당 10DLC 캠페인이 아직 승인되지 않았습니다. 경우에 따라 승인에 1주일 이상 걸릴 수 있습니다. 상태가 변경되면 Amazon Pinpoint 콘솔에 해당 변경 사항이 반영됩니다. 상태 변경에 대해서는 알려드리지 않습니다.
- 거부됨(Rejected) - 해당 10DLC 캠페인이 거부되었습니다. 자세한 내용을 보려면 거부된 캠페인의 캠페인 ID가 포함된 지원 요청을 제출하세요.
- 일시 중단됨 – 하나 이상의 통신 사업자에서 10DLC 캠페인을 일시 중단했습니다. 자세한 내용을 보려면 일시 중단된 캠페인의 캠페인 ID가 포함된 지원 요청을 제출하세요. Amazon Pinpoint에는 콘솔에 일시 중지 이유가 포함되어 있지 않으며 캠페인이 일시 중지된 경우에도 알림을 보내지 않습니다.

8. 10DLC가 승인되면 해당 캠페인과 연결할 10DLC 번호를 요청할 수 있습니다. 10DLC 번호 요청에 대한 자세한 내용은 Amazon SNS를 통한 SMS 메시징을 위한 10DLC 번호, 수신자 부담 전화번호 및 P2P 긴 코드 요청에서 확인하세요.

여러 AWS 리전에서 10DLC 캠페인 사용

회사를 등록하면 해당 회사를 모든 AWS 리전의 AWS 계정에서 사용할 수 있습니다. 그러나 10DLC 캠페인의 경우는 그렇지 않습니다. 10DLC 캠페인은 캠페인이 등록된 AWS 리전에서만 사용할 수 있습니다.

둘 이상의 AWS 리전에서 10DLC를 사용하려는 경우 각 리전에서 별도의 10DLC 캠페인을 등록해야 합니다. 이 단계는 통신 사업자 요구 사항을 준수하기 위해 필요합니다. 사용 사례가 정확히 동일하더라도 등록한 각 캠페인에 대해 요금이 부과됩니다.

AT&T는 각 캠페인에 대해 10DLC 처리율을 제공하기 때문에 여러 캠페인을 등록하면 AT&T를 이동 통신사로 이용하는 수신자에게 보내는 메시지의 처리율을 높일 수 있다는 추가 이점이 있습니다. 한편, T-Mobile은 캠페인 수에 관계없이 각 회사의 일일 메시지 할당을 기반으로 10DLC 처리량을 처리합니다.

10DLC 캠페인 편집 또는 삭제

Amazon Pinpoint 콘솔을 사용하여 10DLC 캠페인의 HELP 응답, STOP 응답 및 샘플 메시지를 편집할 수 있습니다. 또한 콘솔을 사용해 10DLC 캠페인을 삭제할 수 있습니다.

10DLC 캠페인 편집

캠페인이 승인되면 HELP, STOP 및 샘플 메시지를 수정할 수 있습니다. 또한 더 많은 샘플 메시지를 추가할 수 있습니다. 이러한 필드를 변경하는 데는 Campaign Registry 또는 통신 사업자의 재승인이 필요하지 않습니다. 10DLC 캠페인이 승인된 후에는 다른 필드를 수정할 수 없습니다.

최대 5개의 샘플 메시지가 있을 수 있습니다. 처음에 등록한 샘플 메시지 수는 줄일 수 없습니다. 예를 들어 3개의 샘플 SMS 메시지로 캠페인을 등록한 경우 샘플 SMS 메시지 수를 3개 미만으로 줄일 수 없습니다.

> **Note**
>
> HELP, STOP 및 샘플 메시지 이외의 필드를 수정하려면 먼저 10DLC 캠페인을 삭제한 다음 업데이트된 정보를 포함하도록 캠페인을 다시 만들어야 합니다.

10DLC 캠페인을 편집하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/pinpoint/에서 Amazon Pinpoint 콘솔을 엽니다.
2. 탐색 창의 SMS 및 음성(SMS and voice)에서 전화번호(Phone numbers)를 선택합니다.
3. 10DLC 캠페인(10DLC campaigns) 탭에서 편집할 10DLC 캠페인을 선택합니다.
4. 캠페인 세부 정보 페이지의 캠페인 메시지(Campaign messages) 섹션에서 편집(Edit)을 선택합니다.
5. 다음과 같은 필드를 업데이트합니다.

   - 도움말 메시지(Help message)
   - 중지 메시지(Stop message)
   - 샘플 SMS 메시지(Sample SMS message)

   이전에 추가한 샘플 메시지를 삭제하거나 필드가 비어 있도록 샘플 메시지의 내용을 삭제할 수 없습니다. 메시지의 내용을 바꾸지 않고 해당 내용을 삭제하면 업데이트 시 원래 메시지가 사용됩니다.
6. 업데이트(Update)를 선택합니다. 캠페인 메시지가 업데이트되었음을 알리는 확인 배너가 나타납니다.

10DLC 캠페인 삭제

Amazon Pinpoint 콘솔을 사용해 10DLC 캠페인을 삭제할 수 있습니다. 10DLC 캠페인을 삭제하려면 먼저 해당 캠페인에 연결된 전화번호를 모두 제거해야 합니다.

> **Important**
>
> 캠페인에서 10DLC 번호를 제거하면 더 이상 해당 번호에 액세스할 수 없습니다. 또한 삭제된 10DLC 캠페인은 복원할 수 없습니다.

10DLC 캠페인을 삭제하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/pinpoint/에서 Amazon Pinpoint 콘솔을 엽니다.
2. 탐색 창의 SMS 및 음성(SMS and voice)에서 전화번호(Phone numbers)를 선택합니다.
3. 10DLC 캠페인(10DLC campaigns) 탭에서 편집할 10DLC 캠페인을 선택합니다.
4. 전화번호(Phone numbers) 섹션에서 해당 캠페인에 연결된 전화번호를 기록해 둡니다.
5. 전화번호(Phone numbers) 탭에서 제거할 10DLC 번호를 선택한 후전화번호 제거(Remove phone number)를 선택합니다.

   > **Note**
   >
   > 이 단계는 캠페인에 연결된 10DLC 전화번호가 여러 개인 경우에만 필요합니다. 10DLC 캠페인에 연결된 전화번호가 하나만 있는 경우 해당 번호가 10DLC 캠페인(10DLC campaigns) 탭에 표시됩니다. 탭에 나온 번호를 기록해 둡니다.

6. 확인 상자에 **`delete`**를 입력한 다음 확인(Confirm)을 선택합니다. SMS 및 음성(SMS and voice) 페이지 상단에 성공 메시지가 표시됩니다.
7. 캠페인에 연결된 각 10DLC 번호에 대해 앞의 두 단계를 반복합니다.
8. 10DLC 캠페인에 연결된 번호를 모두 제거한 후 10DLC 캠페인(10DLC campaigns) 탭을 선택합니다.
9. 삭제할 10DLC 캠페인을 선택합니다.
10. 10DLC 캠페인 세부 정보(10DLC campaign details) 페이지의 오른쪽 상단에서 삭제(Delete)를 선택합니다.
11. 확인 상자에 **`delete`**를 입력한 다음 확인(Confirm)을 선택합니다. SMS 및 음성(SMS and voice) 페이지 상단에 성공 메시지가 표시됩니다.

긴 코드와 10DLC 캠페인 연결

기존의 긴 코드가 있는 경우 지원 요청을 제출하여 해당 긴 코드를 현재 10DLC 캠페인 중 하나와 연결할 수 있습니다. 10DLC 캠페인과 연결된 긴 코드는 해당 캠페인에만 사용할 수 있으며 다른 10DLC 캠페인에는 사용할 수 없습니다. 긴 코드가 10DLC로 마이그레이션되는 동안에도 계속 사용할 수 있습니다. 그러나 승인을 받기 전까지는 10DLC 캠페인에서 사용할 수 없습니다.

요청을 제출하는 경우 다음이 필요합니다.

- 10DLC 캠페인과 연결할 긴 코드
- 긴 코드와 연결할 10DLC 캠페인 ID

> **Note**
>
> 긴 코드를 캠페인에 연결하려면 먼저 해당 10DLC 캠페인을 등록해야 합니다. 아직 10DLC 캠페인을 만들고 등록하지 않은 경우 10DLC 캠페인 등록에서 확인하세요.

긴 코드를 10DLC에 할당하려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/pinpoint/에서 Amazon Pinpoint 콘솔을 엽니다.
2. 설정의 SMS 및 음성에서 전화번호 탭을 선택합니다.
3. 10DLC로 변환하려는 긴 코드를 선택합니다.
4. 지원 센터를 열려면 10DLC 캠페인에 할당을 선택합니다.
5. 사례 유형으로 서비스 한도 증가를 선택합니다.
6. Limit type(한도 유형)에서 Pinpoint를 선택합니다.
7. 요청(Requests) 섹션에서 리전(Region)을 선택하고 한도(Limit)에서 10 DLC - 10 DLC 캠페인에 기존 미국 긴 코드 연결(10 DLC - Associate existing US long code to 10DLC campaign)을 선택합니다.
8. 사례 설명 아래의 사용 사례 설명에 10DLC 캠페인 ID와 해당 캠페인에 연결할 긴 코드 번호를 포함해야 합니다. 요청에 여러 긴 코드를 포함할 수 있지만 캠페인 ID는 하나만 포함해야 합니다.
9. Contact options(연락처 옵션)에서 Preferred contact language(기본 설정 연락처 언어)에 대해 AWS Support 팀과 통신할 때 사용할 언어를 선택합니다.
10. Contact Method(연락 방법)에서 AWS Support 팀과 통신할 때 사용할 방법을 선택합니다.

11. 제출(Submit)을 선택합니다.

10DLC 교차 계정 액세스

각 10DLC 전화번호는 단일 AWS 리전의 단일 계정에 연결되어 있습니다. 동일한 10DLC 전화번호를 사용하여 둘 이상의 계정 또는 리전에서 메시지를 전송하려는 경우 다음 두 가지 옵션이 있습니다.

1. 각 AWS 계정에서 동일한 회사 및 캠페인을 등록할 수 있습니다. 이러한 등록은 별도로 관리되고 요금이 부과됩니다. 동일한 회사를 여러 AWS 계정에 등록하는 경우 하루에 T-Mobile 고객에게 전송할 수 있는 메시지 수가 각 계정에서 공유됩니다.
2. 하나의 AWS 계정에서 10DLC 등록 프로세스를 완료하고 AWS Identity and Access Management(IAM)를 사용하여 다른 계정에 10DLC 번호를 통해 전송할 수 있는 권한을 부여할 수 있습니다.

> **Note**
>
> 이 옵션을 사용하면 10DLC 전화번호에 대한 진정한 교차 계정 액세스가 가능합니다. 그러나 보조 계정에서 전송된 메시지는 기본 계정에서 전송된 것처럼 취급됩니다. 할당량 및 청구는 보조 계정이 아닌 기본 계정에 대해 계산됩니다.

IAM 정책을 사용하여 교차 계정 액세스 권한 설정

IAM 역할을 사용하여 다른 계정을 기본 계정에 연결할 수 있습니다. 그런 다음 기본 계정의 10DLC 번호에 대한 액세스 권한을 부여하여 기본 계정의 액세스 권한을 보조 계정에 위임할 수 있습니다.

기본 계정의 10DLC 번호에 대한 액세스 권한을 부여하려면

1. 아직 완료하지 않은 경우 기본 계정에서 10DLC 등록 프로세스를 완료합니다. 이 프로세스는 다음 세 단계로 이루어집니다.
   - 회사를 등록합니다. 자세한 내용은 10DLC에 사용할 회사 또는 브랜드 등록을 참조하세요.
   - 10DLC 캠페인(사용 사례)을 등록합니다. 자세한 정보는 10DLC 캠페인 등록을 참조하십시오.
   - 10DLC 캠페인에 전화번호를 연결합니다. 자세한 정보는 긴 코드와 10DLC 캠페인 연결을 참조하십시오.
2. 다른 계정이 10DLC 전화번호에 대한 `Publish` API 작업을 호출할 수 있도록 허용하는 IAM 역할을 기본 계정에 생성합니다. 역할 생성에 대한 자세한 내용은 IAM 사용 설명서의 IAM 역할 생성을 참조하세요.

3. 10DLC 번호를 사용해야 하는 다른 계정과 함께 IAM 역할을 사용하여 기본 계정의 액세스 권한을 위임하고 테스트합니다. 예를 들어, 프로덕션 계정에서 개발 계정으로 액세스 권한을 위임할 수 있습니다. 권한 위임 및 테스트에 대한 자세한 내용은 IAM 사용 설명서의 IAM 역할을 사용하여 AWS 계정 전반에 액세스 권한 위임을 참조하세요.
4. 새 역할을 사용하여 기본 계정에서 10DLC 번호를 사용하여 메시지를 보냅니다. 역할 사용 방법에 대한 자세한 내용은 IAM 사용 설명서의 IAM 역할 사용을 참조하세요.

10DLC 등록 문제에 대한 정보 얻기

경우에 따라 회사 또는 10DLC 캠페인을 등록하려고 할 때 오류 메시지가 수신될 수 있습니다.

회사 등록 문제

회사를 등록하면 확인됨(Verified) 또는 확인되지 않음(Unverified)이라는 두 가지 등록 상태 중 하나가 표시됩니다. 회사 등록 상태가 확인됨(Verified)인 경우 회사 등록이 성공한 것입니다. 10DLC 캠페인 생성을 시작할 수 있습니다.

회사 등록 상태가 확인되지 않음(Unverified)인 경우 제공한 정보에 문제가 있음을 나타냅니다. Amazon Pinpoint 콘솔에 회사 등록이 이 상태를 받은 이유에 대한 정보가 표시됩니다.

10DLC 회사 등록에 대한 등록 문제를 보려면

1. AWS Management Console에 로그인한 후 https://console.aws.amazon.com/pinpoint/에서 Amazon Pinpoint 콘솔을 엽니다.
2. 탐색 창의 SMS에서 전화번호를 선택합니다.
3. 10DLC 캠페인(10DLC campaigns) 탭의 캠페인 목록에서 자세한 정보를 검토할 회사의 이름을 선택합니다.
4. 회사 세부 정보 페이지에는 파악된 등록 관련 문제에 대한 정보가 포함되어 있습니다. 회사 정보(Company info) 섹션의 특정 필드에 경고 기호가 표시되어 있는 경우 등록 문제가 해당 필드의 정보와 관련된 것입니다.

   제공한 정보를 확인하고 모든 필드에 올바른 정보가 포함되어 있는지 확인합니다. Amazon Pinpoint 콘솔에서 회사 등록을 편집할 수 있습니다. 회사 등록 세부 정보 수정에 대한 자세한 내용은 등록된 회사 편집 또는 삭제 단원을 참조하세요.

캠페인 등록 문제

10DLC 캠페인을 등록하면 특정 상황에서 오류 메시지가 표시될 수 있습니다.

등록 관련 문제를 파악할 수 없는 경우 AWS Support Center에 사례를 생성하여 추가 정보를 요청할 수 있습니다. 다음 절차에 따라 AWS Support 사례를 생성합니다. AWS Support 팀에서 10DLC 캠페인 등록이 거부된 이유에 대한 정보를 제공합니다.

거부된 10DLC 캠페인에 대한 정보 요청을 제출하려면

1. https://console.aws.amazon.com/deepracer에서 AWS Management Console에 로그인합니다.
2. 지원 메뉴에서 지원 센터를 선택합니다.
3. 지원 사례 창에서 사례 생성을 선택합니다.
4. 서비스 한도 증가를 원하시나요? 링크를 선택한 후 다음 작업을 완료합니다.
   - 한도 유형에서 Pinpoint SMS를 선택합니다.
5. Requests(요청)에서 다음 섹션을 작성합니다.
   - 리전에서 캠페인을 등록하려고 한 AWS 리전을 선택합니다.

     > **Note**
     >
     > 요청 섹션에는 리전이 필요합니다. 사례 세부 정보 섹션에 이 정보를 제공했더라도 여기에도 포함해야 합니다.

   - 리소스 유형(Resource Type)에서 10DLC 등록(10DLC Registration)을 선택합니다.
   - 한도에서 회사 또는 10DLC 캠페인 등록 거부를 선택합니다.
6. 새 한도 값에서 해당 한도 유형에 대한 한도 증가를 선택합니다. 일반적으로 이 값은 **1**입니다.
7. 사례 설명에 거부된 10DLC 캠페인 ID를 입력합니다.
8. (선택 사항) 추가 요청을 제출하려면 다른 요청 추가를 선택합니다. 여러 요청을 포함할 경우 각 요청에 대해 필요한 정보를 제공합니다. 필요한 정보는 Amazon SNS로 SMS 메시징 지원 요청의 다른 단원을 참조하십시오.
9. 연락처 옵션의 선호하는 문의 언어에서 이 사례에 대한 커뮤니케이션을 수신할 언어를 선택합니다.
10. 마쳤으면 Submit(제출)을 선택합니다.

수신자 부담 전화번호

수신자 부담 전화번호(TFN)는 800, 888, 877, 866, 855, 844 또는 833 지역 번호 중 하나로 시작하는 10자리 숫자입니다. TFN을 사용하여 트랜잭션 메시지만 보낼 수 있습니다.

Important

미국 이동 통신사는 최근 규정을 변경했으며 모든 수신자 부담 전화번호(TFN)는 2022년 9월 30일 이전에 규제 기관에 등록 절차를 완료해야 합니다. the section called “수신자 부담 전화번호 등록 상태”로 이동하여 TFN 상태를 확인합니다. 회사 등록에 대한 자세한 내용은 the section called “수신자 부담 전화번호 등록”에서 확인하세요.
제출 후 등록이 처리되는 데 영업일 기준 최대 15일까지 소요될 수 있습니다.
2023년 3월 3일 업데이트: 2023년 4월 1일부터 등록되지 않은 무료 전화번호를 통해 전송되는 메시지에 대해 다음과 같은 업계 전반의 임계값을 적용합니다.

- 일일 한도: 하루 메시지 500개, 오전 12시(태평양 표준시)에 재설정됩니다.
- 주간 한도: 매주 메시지 1,000개, 일요일 오전 12시(태평양 표준시)에 재설정됩니다.
- 월간 한도: 매월 메시지 2,000개, 월말 오전 12시(태평양 표준시)에 재설정됩니다.

2022년 9월 19일 업데이트: 2022년 10월 1일부터 등록되지 않은 무료 전화번호를 통해 전송되는 메시지에 대해 다음과 같은 업계 전반의 임계값을 적용합니다.

- 일일 한도: 메시지 2,000개
- 주간 한도: 메시지 12,000개
- 월간 한도: 메시지 25,000개

가능한 빨리 등록을 완료하는 것이 좋습니다. 미등록 TFN으로 전송되는 메시지는 최대한으로 제공됩니다. 통신사가 미등록 트래픽을 계속 제한할 경우 시간이 경과함에 따라 메시지의 필터링 및 차단이 강화됩니다.

주제

- 수신자 부담 전화번호 사용 지침
- 수신자 부담 전화번호 구매
- 수신자 부담 전화번호 등록 요구 사항 및 프로세스
- 수신자 부담 전화번호 등록 상태

- 등록 편집, 폐기 및 삭제
- 등록 문제
- 수신자 부담 전화번호 관련 자주 묻는 질문
- 수신자 부담 전화번호의 장점 및 단점

수신자 부담 전화번호 사용 지침

TFN은 일반적으로 등록 확인 또는 일회용 암호 전송과 같은 트랜잭션 메시징 목적으로 미국 내에서만 사용됩니다. 음성 메시징과 SMS에 모두 사용할 수 있습니다. 평균 처리량은 초당 3개의 메시지 파트(MPS)입니다. 그러나 이 처리량은 문자 인코딩의 영향을 받습니다. 문자 인코딩이 메시지 부분에 미치는 영향에 대한 자세한 내용은 Amazon SNS의 SMS 문자 수 한도 섹션을 참조하세요. TFN 등록에 대한 자세한 내용은 수신자 부담 전화번호 등록 요구 사항 및 프로세스 섹션을 참조하세요.

각 고객 계정은 최대 5개의 TFN을 보유할 수 있습니다. 초당 15개 이상 100개 미만의 문자 메시지를 보내는 경우 하나 이상의 10DLC 발신 ID를 등록하는 것이 좋습니다. 사용 사례에서 초당 100개 이상의 문자 메시지를 보내야 하는 경우 하나 이상의 단축 코드를 구입하여 등록하는 것이 좋습니다.

TFN을 발신 번호로 사용하는 경우 다음 지침을 따르세요.

- 이러한 메시지는 스팸으로 필터링될 가능성이 높으므로 서드 파티 URL 단축 프로그램에서 생성된 단축 URL을 사용하지 마십시오.

  단축 URL을 사용해야 하는 경우 10DLC 번호 또는 단축 코드를 사용하는 것이 좋습니다. 단축 코드 및 10DLC를 사용하려면 단축 URL을 지정할 수 있는 메시지 템플릿을 등록해야 합니다.
- 키워드 옵트아웃(STOP) 및 옵트인(UNSTOP) 응답은 통신 사업자 수준에서 설정됩니다. 이러한 키워드 또는 기타 키워드의 수정은 불가능합니다. 또한 사용자가 STOP 및 UNSTOP으로 응답할 때 전송되는 메시지는 수정할 수 없습니다.
- 여러 개의 TFN을 사용하여 동일하거나 유사한 메시지 내용을 보내지 마세요. 통신 사업자는 이 관행을 스노우슈잉 또는 번호 풀링이라고 부르고 이러한 메시지를 필터링의 대상으로 합니다.
- 다음 업종과 관련된 모든 메시지는 제한된 것으로 간주될 수 있으며 과중한 필터링 또는 완전한 차단이 적용됩니다. 여기에는 제한된 범주와 관련된 서비스에 대한 일회용 암호(OTP) 및 다중 인증(MFA)이 포함될 수 있습니다.

  규정을 준수하지 않는 사용 사례로 등록이 거부되었는데 이 지정이 잘못되었다고 생각되는 경우 지원을 통해 요청을 제출할 수 있습니다. 이 작업을 수행하는 자세한 방법은 등록 문제 단원을 참조하십시오.

  다음 표에서는 제한된 콘텐츠 유형에 대해 설명합니다.

범주	예
도박	• 앱/웹사이트 • 카지노 • 경품행사
고위험 금융 서비스	• 자동차 대출 • Cryptocurrency • 채권 추심 • 월급 대출 • 단기 고이자 대출 • 모기지론 • 학생 대출 • 주식 알림
부채 탕감	• 부채 통합 • 부채 감면 • 신용 회복 프로그램
G et-rich-quick 스킴	• W 프로그램 ork-from-home • 위험-투자 기회 • 피라미드 또는 다단계 마케팅 방식
금지/통제 물질	• 대마초/CBD
피싱	• 사용자가 개인 정보 또는 웹 사이트 로그인 정보를 공개하도록 시도
S.H.A.F.T.	• Sex • 증오 • 알코올 • 총기 • 담배/베이프

수신자 부담 전화번호 구매

TFN을 구매하려면 https://console.aws.amazon.com/sms-voice/ 에서 Amazon Pinpoint 콘솔을 사용하십시오. 자세한 설명은 수신자 부담 전화번호 등록 요구 사항 및 프로세스 섹션을 참조하세요.

현재 Amazon Pinpoint SMS는 음성 메시지와 SMS 메시지 모두에 대해 무료 전화 번호를 지원합니다. Amazon SNS는 SMS 메시징만 지원합니다.

수신자 부담 전화번호 등록 요구 사항 및 프로세스

> **Important**
>
> TFN이 지정된 사용 사례 이외의 다른 용도로 사용되는 경우 취소될 수 있습니다.

수신자 부담 전화번호 금지 사용 사례

Amazon SNS는 메시지가 차단된 경우(예: 규제 약물 또는 피싱과 관련된 사용 사례) 또는 높은 수준의 필터링이 예상되는 경우(예: 고위험 금융 메시지), 메시지 전송 기능이 제한됩니다. 수신자 부담 전화번호 사용 지침에 정의된 제한된 콘텐츠 사용 사례와 관련된 TFN은 등록하지 못할 수 있습니다.

수신자 부담 전화번호 등록

TFN을 구매한 후에는 번호를 등록해야 합니다. 이를 수행하는 방법에 대한 지침은 Amazon Pinpoint SMS 사용 설명서의 무료 전화 번호 등록 프로세스를 참조하십시오.

Amazon Pinpoint SMS 지역의 무료 전화 번호를 위한 셀프 서비스 등록

Amazon Pinpoint SMS 지역에서 TFN을 요청한 경우, Amazon Pinpoint SMS 사용 설명서의 미국 무료 전화 번호 등록 양식에 있는 지침을 사용하여 Amazon Pinpoint SMS 콘솔에서 직접 회사 등록 프로세스를 완료하십시오.

TFN을 등록할 때 누락된 부분이 없고 정확한 정보인지 확인하세요. 그렇지 않으면 등록이 거부될 수 있습니다. 입력하는 정보는 기업 본사의 정보와 정확히 일치해야 합니다.

Amazon Pinpoint SMS 지역 이외의 지역에 있는 무료 전화 번호에 대한 수동 양식 기반 등록 프로세스

1. 이 US_TFN_Registration.zip 파일을 다운로드하고 예제 등록 양식 (AWS 미국 무료 등록 양식-비즈니스 - Final.docx) 을 사용하여 TFN 등록 CSV 파일 (BulkustFN - Final.csv) 의 필수 정보를 작성하십시오.

각 등록 요청 또는 사용 사례는 최대 5개의 TFN만 포함할 수 있습니다. 이 규칙의 면제 대상이 된다고 생각되면 고려 사항에 대한 자세한 설명을 제공합니다. 등록 또는 사용 사례와 관련된 모든 전화번호를 나열합니다.

2. AWS Support에서 사례를 생성합니다. 작성한 CSV 파일을 사례에 첨부하고 TFN 등록 요청을 제출합니다.
3. 사례 생성을 선택한 다음 서비스 한도 증가를 원하시나요?를 선택합니다.
4. 한도 유형에서 SNS 문자 메시지를 선택합니다.
5. Resource Type(리소스 유형)에서 10DLC or Toll-free number registration(10DLC 또는 수신자 부담 번호 등록)을 선택합니다.
6. US_TFN_registration 문서를 첨부하고 요청을 제출합니다.

유의 사항

1. 필수 정보가 모두 제출된 후 등록을 처리하는 데 최대 2주가 소요될 수 있습니다. 정보가 누락되거나 불완전할 경우 등록 절차가 지연됩니다. 등록이 거부된 경우 등록 거부 사유를 찾고 캠페인을 개선하여 등록할 수 있는 방법을 제안해 드립니다.
2. TFN은 제한된 처리량이 필요한 다중 인증(MFA)과 같은 트랜잭션 사용 사례에 적합합니다. 각 TFN은 초당 최대 3개의 문자 메시지 파트를 보낼 수 있으며, 각 고객 계정은 최대 5개의 TFN을 보유할 수 있습니다. 초당 15개 이상 100개 미만의 문자 메시지 파트를 보내는 경우 하나 이상의 10DLC 발신 ID를 등록하는 것이 좋습니다. 사용 사례에서 초당 100개 이상의 문자 메시지를 보내야 하는 경우 하나 이상의 단축 코드를 구입하여 등록하는 것이 좋습니다. 자세한 내용은 수신자 부담 전화번호 사용 지침를 참조하세요.

수신자 부담 전화번호 등록 상태

등록 상태를 확인하려면 Amazon Pinpoint SMS 사용 설명서의 등록 상태 확인을 참조하십시오.

등록 편집, 폐기 및 삭제

Amazon Pinpoint SMS 사용 설명서를 사용하여 다음 작업을 수행하십시오.

- 등록을 편집하십시오.
- 등록을 취소하세요.
- 등록을 삭제하세요.
- 등록 리소스 보기

등록 문제

수신자 부담 번호 등록이 수락되지 않는 경우 등록 거부 사유를 설명하는 메시지가 표시됩니다.

거부된 수신자 부담 전화번호에 대한 정보 요청을 제출하려면

1. https://console.aws.amazon.com/ AWS Management Console 에서 로그인하십시오.
2. 지원 메뉴에서 지원 센터를 선택합니다.
3. 지원 사례 창에서 사례 생성을 선택합니다.
4. 서비스 한도 증가를 원하시나요? 링크를 선택한 후 다음 작업을 완료합니다.
   - 한도 유형에서 Pinpoint SMS를 선택합니다.
5. 요청에서 다음 섹션을 작성합니다.
   - 리전에서 캠페인을 등록하려고 한 리전을 선택합니다.

     > **Note**
     >
     > 요청 섹션에는 리전이 필요합니다. 사례 세부 정보 섹션에 이 정보를 제공했더라도 여기에도 포함해야 합니다.

   - 리소스 유형에서 10DLC 또는 TFN 등록을 선택합니다.
   - 한도에서 회사 또는 캠페인 등록 거부를 선택합니다.
6. 새 한도 값에서 해당 한도 유형에 대한 한도 증가를 선택합니다. 일반적으로 이 값은 **1**입니다.
7. (선택 사항) 추가 요청을 제출하려면 다른 요청 추가를 선택합니다. 필요한 정보는 Amazon SNS로 SMS 메시징 지원 요청의 다른 단원을 참조하십시오.
8. 사례 설명에 거부된 수신자 부담 전화번호를 입력합니다.
9. 연락처 옵션의 선호하는 문의 언어에서 이 사례에 대한 커뮤니케이션을 수신할 언어를 선택합니다.
10. 마쳤으면 제출을 선택합니다.

수신자 부담 전화번호 관련 자주 묻는 질문

TFN 등록 절차 관련 FAQ

현재 내가 수신자 부담 전화번호를 보유하고 있습니까?

수신자 부담 전화번호를 보유하고 있는지 확인하려면

- https://console.aws.amazon.com/sms-voice/ 에서 Amazon Pinpoint SMS 콘솔을 엽니다.
- 탐색 창의 SMS에서 전화번호를 선택합니다.
- TFN type(유형)은 toll-free(수신자 부담)로 나열됩니다.

수신자 부담 전화번호를 등록해야 합니까?

예. 현재 보유하고 있는 TFN을 계속 사용하려면 2022년 9월 30일 이전에 TFN을 등록해야 합니다. 2022년 9월 30일 이후에 새 TFN을 구매하는 경우 등록해야만 메시지를 보낼 수 있습니다.

수신자 부담 전화번호는 어떻게 구매합니까?

Amazon Pinpoint SMS 콘솔을 사용하여 전화번호 요청의 지침을 따라 TFN을 구매하십시오.

수신자 부담 전화번호는 어떻게 등록합니까?

TFN을 등록하려면 the section called “수신자 부담 전화번호 등록”의 지침을 따릅니다.

수신자 부담 전화번호의 등록 상태란 무엇이며 어떤 의미가 있습니까?

the section called “수신자 부담 전화번호 등록 상태”에 있는 지침에 따라 등록 및 상태를 확인할 수 있습니다.

어떤 정보를 제공해야 합니까?

TFN을 사용하는 회사 주소, 비즈니스 연락처 및 사용 사례를 제공해야 합니다. 필요한 정보는 the section called “수신자 부담 전화번호 등록”에서 찾을 수 있습니다.

등록이 거부되면 어떻게 됩니까?

등록이 거부되면 상태가 Requires Updates(업데이트 필요)로 변경됩니다. 업데이트하려면 the section called “등록 편집, 폐기 및 삭제” 섹션을 참조하세요.

필요한 권한은 무엇입니까?

Amazon Pinpoint SMS 콘솔을 방문하는 데 사용하는 IAM 사용자/역할에는 *“sms-voice: *”* 권한이 있어야 합니다. 그렇지 않으면 액세스 거부 오류가 발생합니다.

수신자 부담 전화번호의 장점 및 단점

장점

수신자 부담 발신자는 긴 코드에 비해 MPS가 높고 발송률이 좋습니다.

단점

옵트아웃 및 옵트인은 이동 통신사 수준에서 관리되므로 제어할 수 없습니다.

메시지에 단축 URL을 포함하거나 해당 전화번호를 사용하여 홍보 메시지를 전송해서는 안 됩니다. 대신 10DLC 번호 또는 단축 코드를 사용하세요. 단축 코드 또는 10DLC 번호를 사용하는 경우 단축 URL을 포함할 수 있고 홍보 메시지가 될 수 있는 메시지 템플릿을 등록해야 합니다. 단축 코드에 대한 자세한 내용은 단축 코드 섹션을 참조하세요. 10DLC에 대한 자세한 내용은 10DLC 섹션을 참조하세요.

단축 코드

단축 코드는 일반 전화번호보다 짧은 번호 배열입니다. 예를 들어 미국과 캐나다의 표준 전화번호(긴 코드)는 11자리인데, 단축 코드에는 5자리 또는 6자리가 배정됩니다. Amazon SNS는 전용 단축 코드를 지원합니다.

전용 단축 코드

미국이나 캐나다에 있는 수신자에게 SMS 메시지를 대량 발송하려는 경우 전용 단축 코드를 구입할 수 있습니다. 공유 풀의 단축 코드와 달리 전용 단축 코드는 단독으로 사용할 수 있습니다.

장점

기억에 남을 만한 단축 코드를 사용하면 신뢰 구축에 도움이 됩니다. 일회용 비밀번호처럼 민감한 정보를 발송해야 하는 경우, 이 메시지가 실제로 귀하에게서 온 것인지 고객이 금방 판단할 수 있도록 단축 코드를 사용하여 발송하는 것이 좋습니다.

신규 고객 확보 캠페인을 진행 중인 경우 단축 코드로 키워드를 발송하여 잠재 고객을 초대할 수 있습니다(예: "축구 소식과 정보를 받아보려면 10987번으로 'FOOTBALL'이라는 문자를 보내세요"). 단축 코드는 긴 코드보다 기억하기가 쉽고, 고객이 디바이스에 입력하기가 더 쉽습니다. 마케팅 프로그램에 등록할 때 고객의 수고를 덜어줌으로써 캠페인 효과를 높일 수 있습니다.

단축 코드를 새로 사용하기 전에 모바일 통신사의 승인이 필요하기 때문에 단축 코드는 원치 않는 메시지로 처리될 가능성이 더 낮습니다.

단축 코드로 SMS 메시지를 발송할 때는 다른 유형의 발신 자격 증명을 사용할 때보다 24시간 기준으로 더 많은 양의 메시지를 발송할 수 있습니다. 즉, 발신 할당량이 더 높습니다. 초당 메시지도 훨씬 더 많이 보낼 수 있습니다. 즉, 발송 속도가 더 빠릅니다.

단점

단축 코드를 확보하려면 추가 요금이 있으며, 실제로 사용하기까지 오랜 시간이 걸립니다. 예를 들어, 미국에서 각 단축 코드의 일회 설정 요금은 $650.00(USD)이고, 단축 코드 회선당 매월 기본 요금 $995.00가 추가로 발생합니다. 모든 통신 사업자 네트워크에서 단축 코드가 활성화되려면 8-12주가 걸릴 수 있습니다. 다른 국가 또는 리전의 가격 및 프로비저닝 시간을 확인하려면 Amazon SNS에서 SMS 메시징에 대한 전용 단축 코드 요청에 설명된 절차를 완료합니다.

개인 대 개인(P2P) 긴 코드

> **Important**
>
> 2023년 8월 31일부터 미국 및 미국 영토(푸에르토리코, 괌, 미국령 사모아 제도, US 버진 아일랜드)로 SMS 문자 메시지를 보내려면 10DLC 번호 또는 수신자 부담 전화번호와 같은 전용 번호가 필요합니다. 이러한 리전의 위치로 미국을 사용하는 경우 긴 코드 요청이 거부됩니다.

> **Important**
>
> 2021년 6월 1일부로 미국 통신 사업자는 더 이상 미국 대상으로의 A2P(애플리케케이션 대 개인) 통신에 P2P(개인 대 개인) 긴 코드 사용을 지원하지 않습니다. 대신 이러한 메시지에 다른 유형의 발신 ID를 사용해야 합니다. 자세한 내용은 10DLC 섹션을 참조하세요.

P2P 긴 코드는 수신자가 위치한 국가 또는 리전의 번호 형식을 사용하는 전화번호입니다. P2P 긴 코드는 긴 번호 또는 가상 모바일 번호라고도 합니다. 예를 들어 미국과 캐나다에서 P2P 긴 코드에는 11자리가 들어가는데 숫자 1(국가 코드), 3자리 지역 코드, 7자리 전화번호로 구성됩니다.

P2P 긴 코드 신청에 대한 자세한 내용은 Amazon SNS를 통한 SMS 메시징을 위한 10DLC 번호, 수신자 부담 전화번호 및 P2P 긴 코드 요청에서 확인하세요.

장점

전용 P2P 긴 코드는 Amazon SNS 계정에서만 사용할 수 있도록 예약되어 있으며 다른 사용자와 공유되지 않습니다. 전용 P2P 긴 코드를 사용할 경우 각 메시지를 전송할 때 어떤 P2P 긴 코드를 사용할지 지정할 수 있습니다. 동일 고객에게 복수의 메시지를 보낼 경우, 동일한 전화번호에서 각 메시지를 발송했는지 확인할 수 있습니다. 이 때문에 전용 P2P 긴 코드는 브랜드 또는 자격 증명 확립에 유용합니다.

단점

US 대상으로의 A2P 통신에는 P2P 긴 코드가 지원되지 않습니다.

전용 P2P 긴 코드에서 하루에 수백 통의 메시지를 발송할 경우, 원치 않는 메시지를 보내는 번호로 모바일 통신사가 인식할 수 있습니다. P2P 긴 코드에 플래그가 지정된 경우, 메시지가 수신자에게 전송되지 않을 수 있습니다.

P2P 긴 코드에도 처리량 제한이 있습니다. 최대 전송 요금은 국가별로 다릅니다. 자세한 정보는 AWS Support에 문의하세요. SMS 메시지를 다량 발송할 계획이거나 초당 1개 메시지 이상의 속도로 발송할 계획이라면 전용 단축 코드를 구입해야 합니다.

미국을 포함하여 일부 통신 사업자는 P2P 긴 코드를 사용하여 A2P SMS 메시지를 보내는 것을 허용하지 않습니다. A2P SMS는 고객이 자신의 휴대 전화번호를 애플리케이션에 제공할 때 모바일 디바이스에 발송되는 메시지입니다. A2P 메시지는 단방향 대화입니다. 마케팅 메시지, 일회용 비밀번호, 예약 알림 같은 예가 대표적입니다. A2P 메시지를 발송할 계획이라면 전용 단축 코드를 구입하거나(고객이 미국 또는 캐나다에 있는 경우), 발신자 ID를 구입해야 합니다(수신자가 발신자 ID가 지원되는 국가 또는 리전에 있는 경우).

10DLC 번호는 미국 내에서 메시지를 보내는 데만 사용됩니다. 10DLC 번호는 회사 브랜드와 번호를 연결하려는 캠페인을 등록해야 사용 가능합니다. 승인되면 https://console.aws.amazon.com/pinpoint/에 있는 Amazon Pinpoint 콘솔의 SMS 및 음성 페이지에서 10DLC 전화번호를 요청할 수 있습니다. 요청 후 승인을 받는 데 걸리는 시간은 7~10일입니다. 이 번호는 다른 캠페인에서는 사용할 수 없습니다.

미국 제품 번호 비교

이 표에서는 미국 전화번호 유형에 대한 지원 비교를 보여 줍니다.

제품 특징	단축 코드	수신자 부담 전화번호	10DLC	
숫자 형식	5-6자리	10자리 숫자	10자리 숫자	
채널 지원	SMS	SMS	SMS	
SMS 트래픽 유형	프로모션 및 트랜잭션	트랜잭션	프로모션 및 트랜잭션	
심사 필요	예	아니요	예	

제품 특징	단축 코드	수신자 부담 전화 번호	10DLC
예상 프로비저닝 시간	12주[1]	15영업일	1주
SMS 처리량(초당 SMS 메시지 수)[2]	초당 100개의 메시지 부분 - 추가 요금을 지불하고 더 높은 처리량을 사용할 수 있습니다.	초당 3개의 메시지 부분	10DLC 등록에 따라 다릅니다. 초당 최대 100개의 메시지 부분을 지원합니다.
필수 키워드	옵트인, 옵트아웃 및 도움말	중지, 중지 해제 이 항목들은 네트워크 관리형입니다. 옵트아웃 및 옵트인 복구 메시지는 변경할 수 없습니다.	옵트인, 옵트아웃 및 도움말

[1] 프로비저닝 예상값에는 승인 시간이 포함되지 않습니다.

[2] SMS 메시지의 최대 크기에 대한 자세한 정보는 휴대폰에 게시에서 확인하세요.

Amazon SNS로 SMS 메시징 지원 요청

Amazon SNS의 특정 SMS 옵션은 AWS Support에 문의할 때까지 AWS 계정에 사용할 수 없습니다. 다음을 요청하려면 AWS Support 센터에서 사례를 생성하세요.

- 월별 SMS 지출 임계값 증가

  기본적으로 월별 지출 임계값은 $1.00(USD)입니다. 지출 임계값에 따라 Amazon SNS에서 보낼 수 있는 메시지의 볼륨이 결정됩니다. SMS 사용 사례에 대한 예상 월별 메시지 볼륨에 맞는 지출 임계값을 요청할 수 있습니다.
- SMS 샌드박스에서 이동하여 제한 없이 SMS 메시지를 보낼 수 있습니다. 자세한 내용은 SMS 샌드박스 환경에서 나가기 섹션을 참조하세요.
- 전용 발신 번호

- 전용 발신자 ID

  발신자 ID는 수신자 디바이스에 발신자로 표시되는 사용자 지정 ID입니다. 예를 들어, 기업 브랜드를 사용하여 메시지 소스를 더 인식하기 쉽게 만들 수 있습니다. 발신자 ID에 대한 지원은 국가 또는 리전별로 다릅니다. 자세한 내용은 지원되는 국가 및 리전 섹션을 참조하세요.

AWS Support 센터에서 사례를 생성할 때 제출하는 요청 유형에 필요한 모든 정보를 포함해야 합니다. 그렇지 않으면 AWS Support에서 진행하기 전에 이 정보를 확인하기 위해 연락해야 합니다. 요청이 지체 없이 이행되도록 요청을 제출할 때 자세하게 작성해 주십시오. 특정 SMS 요청 유형에 필요한 세부 정보는 다음 주제를 참조하세요.

주제

- Amazon SNS에서 SMS 메시징에 대한 전용 단축 코드 요청
- Amazon SNS를 통한 SMS 메시징을 위한 10DLC 번호, 수신자 부담 전화번호 및 P2P 긴 코드 요청
- Amazon SNS로 SMS 메시징에 대한 발신자 ID 요청
- Amazon SNS에 대한 월별 SMS 지출 할당량 증가 요청

Amazon SNS에서 SMS 메시징에 대한 전용 단축 코드 요청

단축 코드는 대용량 SMS 메시지 전송에 사용할 수 있는 번호입니다. 단축 코드는 흔히 A2P(application-to-person), 2팩터 인증(2FA), 마케팅에 사용됩니다. 단축 코드는 기반을 둔 국가 또는 리전에 따라 일반적으로 3~7자리로 구성됩니다.

단축 코드로 해당 단축 코드가 기반한 국가와 동일한 국가의 수신자에게만 메시지를 보낼 수 있습니다. 둘 이상의 국가에서 단축 코드를 사용해야 하는 경우, 수신자가 위치한 국가별로 다른 단축 코드를 요청해야 합니다.

단축 코드 요금에 대한 자세한 내용은 Amazon SNS 요금을 참조하세요.

> **Important**
>
> Amazon SNS에서 SMS 메시징을 처음 사용하는 경우 SMS 사용 사례의 예상 수요에 맞게 월별 SMS 지출 임계값을 요청하세요. 기본적으로 월별 지출 임계값은 $1.00(USD)입니다. 단축 코드에 대한 요청을 포함하는 동일한 지원 사례에서 지출 임계값을 늘리도록 요청할 수 있습니다. 또한 별도 사례를 사용할 수 있습니다. 자세한 내용은 Amazon SNS에 대한 월별 SMS 지출 할당량 증가 요청 섹션을 참조하세요.

또한 개인 건강 정보(PHI)를 포함하거나 포함할 수 있는 메시지를 보내기 위해 전용 단축 코드를 요청하는 경우, 아래에 설명된 대로 지원 사례를 열 때 사례 설명에서 이 목적을 식별해야 합니다.

Amazon SNS 단축 코드 사례 열기

다음 단계를 수행하여 AWS Support에서 사례를 생성합니다.

> **Note**
>
> 요청 양식의 일부 필드는 “선택 사항”으로 표시되어 있습니다. 그러나 AWS Support에서 요청을 처리하려면 아래의 단계에 언급된 정보가 전부 필요합니다. 필요한 정보를 모두 제공하지 않으면 요청 처리가 지연될 수 있습니다.

전용 단축 코드를 요청하려면

1. AWS 지원 센터로 이동합니다.
2. https://console.aws.amazon.com/deepracer에서 AWS Management Console에 로그인합니다.
3. 지원 메뉴에서 지원 센터를 선택합니다.
4. 지원 사례 창에서 사례 생성을 선택합니다.
5. 서비스 한도 증가를 원하시나요? 링크를 선택한 후 다음 작업을 완료합니다.
   - 한도 유형에서 SNS 문자 메시지를 선택한 후 다음 작업을 완료합니다.
   - (선택 사항) SMS 메시지를 보낼 사이트 또는 앱 링크 제공에서 대상 구성원이 SMS 메시지 수신을 선택한 사이트에 대한 링크 또는 애플리케이션의 이름을 제공합니다.
   - (선택 사항) 보낼 메시지의 유형에서 긴 코드를 사용하여 보낼 메시지의 유형을 선택합니다.
     - 일회용 암호 – 고객이 웹 사이트 또는 애플리케이션에서 인증을 위해 사용해야 하는 암호가 담긴 메시지입니다.
     - 프로모션 – 비즈니스 또는 서비스(예: 특가 행사, 공지 사항)를 홍보하는 중요하지 않은 메시지입니다.
     - 트랜잭션 – 고객 트랜잭션을 지원하는 중요한 정보 메시지(예: 주문 확인, 계정 알림)입니다. 트랜잭션 메시지에 홍보 또는 마케팅 콘텐츠를 포함해서는 안 됩니다.
   - (선택 사항) 메시지를 보낼 AWS 리전에서 메시지를 보낼 리전을 선택합니다.
   - (선택 사항)메시지를 보낼 국가에 SMS 메시지를 보낼 국가 또는 리전을 입력합니다.

- (선택 사항) 고객이 메시지 수신에 옵트인하는 방법에서 고객이 메시지 수신에 옵트인하는 방법에 대한 설명을 제공합니다.
- (선택 사항) 고객에게 메시지를 보내는 데 사용할 메시지 템플릿을 제공하십시오 필드에 사용할 메시지 템플릿을 포함시킵니다.

6. Requests(요청)에서 다음 섹션을 작성합니다.

   - 리전에서 단축 코드를 요청할 AWS 리전을 선택합니다.

     > **Note**
     >
     > 요청 섹션에는 리전이 필요합니다. 사례 세부 정보 섹션에 이 정보를 제공했더라도 여기에도 포함해야 합니다.

   - 리소스 유형에서 Dedicated SMS Short Codes(전용 SMS 단축 코드)를 선택합니다.
   - Limit(제한)에서 사용 사례와 가장 유사한 옵션을 선택합니다.

7. 새 한도 값에서 요청할 발신자 ID의 번호를 선택합니다. 일반적으로 이 값은 **1**입니다.
8. (선택 사항) 추가 요청을 제출하려면 다른 요청 추가를 선택합니다. 필요한 정보는 Amazon SNS로 SMS 메시징 지원 요청의 다른 단원을 참조하십시오.
9. 사례 설명에서 사용 사례를 요약하고 단축 코드로 보낸 메시지에 수신자가 등록하는 방법을 포함하고 다음 정보를 제공합니다.

   - 회사 정보:
     - 회사 이름
     - 회사 우편 주소
     - 요청에 대한 기본 담당자의 이름 및 전화번호
     - 회사의 지원 관련 이메일 주소 및 수신자 부담 전화번호
     - 회사 납세자 번호
     - 제품 또는 서비스의 이름

   - 사용자 등록 프로세스:
     - 고객이 사용자의 단축 코드로부터 메시지를 수신하기 위해 로그인할 웹 사이트 또는 회사 웹 사이트

- 사용자가 사용자 단축 코드로부터 메시지를 수신하기 위해 등록하는 방법입니다. 다음 옵션 중 하나 이상을 지정합니다.
  - **Text messages**
  - **Website**
  - **Mobile app**
  - **Other** 기타를 선택한 경우 설명을 입력하세요.
- 웹 사이트, 앱 등에서 메시지에 대해 등록할 옵션에 대한 텍스트
- 더블 옵트인에 사용하려는 메시지의 시퀀스. 다음 정보를 모두 제공합니다.
  1. 사용자가 등록하면 보낼 SMS 메시지. 이 메시지는 반복 메시지에 대한 사용자의 동의를 요청합니다. 예: ExampleCorp: 계정 트랜잭션 알림을 수신하려면 YES(예)로 응답합니다. 메시지 및 데이터 요금이 부과될 수 있습니다.
  2. 사용자로부터 기대되는 옵트인 응답입니다. 일반적으로 키워드(예: YES)입니다.
  3. 고객이 이 키워드를 단축 코드로 보내면 보낼 확인 메시지. 예: 이제 ExampleCorp의 계정 알림에 등록되었습니다. 메시지 및 데이터 요금이 부과될 수 있습니다. 취소하려면 STOP, 정보를 보려면 HELP라고 메시지를 보내세요.

- 메시지의 목적:
  - 단축 코드를 사용하여 보낼 메시지의 목적입니다. 다음 옵션 중 하나를 지정하세요.
    - **Promotions and marketing**
    - **Location-based services**
    - **Notifications**
    - **Information on demand**
    - **Group chat**
    - **Two-factor authentication (2FA)**
    - **Polling and surveys**
    - **Sweepstakes or contests**
    - **Other** 기타를 선택한 경우 설명을 입력하세요.
  - 본인 소유가 아닌 비즈니스에 대한 프로모션 또는 마케팅 메시지를 보내는 데 단축 코드를 사용할지 여부
  - HIPAA(미국 건강 보험 양도 및 책임에 관한 법)와 관련 법률 및 규정에 의해 정의된 대로 개인 건강 정보(PHI)를 포함하거나 포함할 수 있는 메시지를 보내기 위해 약식 코드를 사용할 계획인지 여부.

- 메시지 콘텐츠:
  - 고객이 특정 키워드를 전송하여 메시지를 옵트인할 때 고객에게 전송하는 메시지. 이 키워드와 메시지를 지정할 때 주의하세요. 이 메시지를 변경하는 데 몇 주가 소요될 수 있습니다. 단축 코드가 생성되면 단축 코드를 사용하는 국가의 휴대폰 통신사에 키워드와 메시지가 등록됩니다. 메시지는 다음 예와 유사할 수 있습니다. 예: *ProductName* 알림에 오신 것을 환영합니다! 메시지 및 데이터 요금이 적용됩니다. 매월 메시지 *2*건. 지원을 받으려면 HELP, 취소하려면 STOP이라고 답장하세요.
  - 고객이 HELP 키워드로 메시지에 응답할 경우 보낼 응답. 이 메시지에는 고객 지원 연락처 정보가 포함되어야 합니다. 예: *ProductName* 알림: 지원을 받으려면 *example.com/help* 또는 *(800) 555-0199*로 연락하세요. 메시지 및 데이터 요금이 적용됩니다. 매월 메시지 *2*건. 취소하려면 'STOP'이라고 답장을 보내주세요.
  - 고객이 STOP 키워드로 메시지에 응답할 경우 보낼 응답. 이 메시지는 사용자가 더 이상 메시지를 수신하지 않을 것임을 확인해야 합니다. 예: *ProductName* 알림이 구독 해제되었습니다. 더 이상 메시지가 전송되지 않습니다. 지원을 받으려면 HELP라고 답장하거나 *(800) 555-0199*로 연락하세요.
  - 사용자가 메시지를 구독했음을 나타내는 정기 알림으로 보낼 텍스트. 예: 알림: 고객님은 ExampleCorp의 계정 알림을 구독하고 있습니다. 메시지 및 데이터 요금이 부과될 수 있습니다. 취소하려면 STOP, 정보를 보려면 HELP라고 메시지를 보내세요.
  - 단축 코드를 사용하여 보낼 각 메시지 유형의 예. 예를 3가지 이상 제공합니다. 네 개 이상의 메시지 유형을 보내려는 경우, 모든 유형에 대한 예를 제공합니다.

> **Important**
>
> 이동통신 사업자는 단축 코드를 프로비저닝하려면 위에 나열된 모든 정보가 필요합니다. 사용자가 이 정보를 모두 제공할 때까지 요청을 처리할 수 없습니다.

10. (선택 사항) 추가 요청을 제출하려면 다른 요청 추가를 선택합니다. 여러 요청을 포함할 경우 각 요청에 대해 필요한 정보를 제공합니다. 필요한 정보는 Amazon SNS로 SMS 메시징 지원 요청의 다른 단원을 참조하십시오.
11. 연락처 옵션의 선호하는 문의 언어에서 이 사례에 대한 커뮤니케이션을 수신할 언어를 선택합니다.
12. 마쳤으면 Submit(제출)을 선택합니다.

요청이 접수되면 24시간 안에 첫 번째 회신을 합니다. 추가 정보를 요청하기 위해 연락을 드릴 수도 있습니다. 단축 코드를 제공할 수 있는 경우, 요청에 지정된 국가 또는 리전에서 단축 코드를 받는 데 따르는 비용 정보를 보내 드립니다. 또한 해당 국가 또는 리전에서 단축 코드를 프로비저닝하는 데 필요한 예상 시간도 알려 드립니다. 단축 코드를 프로비저닝하는 데는 보통 몇 주가 걸리며, 해당 단축 코드의 국가 또는 리전에 따라 지연 시간이 훨씬 더 짧아지거나 길어질 수 있습니다.

> **Note**
>
> 단축 코드 사용과 관련된 수수료는 귀하의 단축 코드를 통신사에 요청한 후 바로 시작됩니다. 아직 단축 코드가 완전히 프로비저닝되지 않았더라도 이러한 비용을 지불해야 합니다.

시스템이 원치 않는 콘텐츠 또는 악성 콘텐츠를 전송하지 않도록 하기 위해 각 요청을 신중하게 고려해야 합니다. 정책에 맞지 않는 사용 사례인 경우, 요청에 대한 권한을 부여하지 않을 수도 있습니다.

다음 단계

무선통신 사업자에 단축 코드를 등록하고 Amazon SNS 콘솔에서 설정을 검토했습니다. 이제 Amazon SNS에서 단축 코드를 발신 번호로 사용하여 SMS 메시지를 보낼 수 있습니다.

Amazon SNS를 통한 SMS 메시징을 위한 10DLC 번호, 수신자 부담 전화번호 및 P2P 긴 코드 요청

> **Important**
>
> 2021년 6월 1일부터 미국 통신 사업자는 미국 목적지로의 (A2P) 통신을 위한 person-to-person application-to-person (P2P) 롱 코드 사용을 더 이상 지원하지 않습니다. 대신 이러한 메시지에 다른 유형의 발신 ID를 사용해야 합니다. 자세한 설명은 10DLC 섹션을 참조하세요.

10DLC 번호, 수신자 부담 전화번호 및 P2P 긴 코드를 요청하려면 Amazon Pinpoint 콘솔을 사용하세요. 자세한 지침은 Amazon Pinpoint 사용 설명서의 번호 요청을 참조하세요.

> **Important**
>
> 미국 이동 통신사는 최근 규정을 변경했으며 모든 수신자 부담 전화번호(TFN)는 2022년 9월 30일 이전에 규제 기관에 등록 절차를 완료해야 합니다. 수신자 부담 전화번호에 대한 자세한 내용은 수신자 부담 전화번호 등록에서 확인하세요.

2022년 9월 30일 또는 그 이전에 수신자 부담 전화번호를 구매한 경우 등록을 완료하고 상태가 완료됨으로 설정된 상태로 반환되지 않는 한 해당 상태는 2022년 10월 1일까지 활성 상태입니다. 그렇지 않으면 번호를 등록하거나 등록이 반환되거나 등록이 활성 상태로 설정될 때까지 메시지가 보류 중 상태가 되며 메시지를 보낼 수 없습니다.
등록 절차는 최대 15일이 소요될 수 있습니다.

10DLC 회사와 캠페인을 등록하려면 미국 동부 (버지니아 북부) 지역에서 Amazon Pinpoint 콘솔을 여십시오. 10DLC 번호를 요청하는 대신 Service AWS Quotas 콘솔을 사용하여 해당 지역의 10DLC 번호를 요청하는 동안 서비스 한도 증가 사례를 생성하십시오. Amazon Pinpoint를 사용할 수 있는 리전에 대한 자세한 내용은 AWS 일반 참조의 Amazon Pinpoint 엔드포인트 및 할당량을 참조하세요.

Note

Amazon SNS에서 SMS 메시징을 처음 사용하는 경우 SMS 사용 사례의 예상 수요를 충족하는 월별 SMS 지출 임계값도 요청해야 합니다. 기본적으로 월별 지출 임계값은 $1.00(USD)입니다. 자세한 내용은 Amazon SNS에 대한 월별 SMS 지출 할당량 증가 요청에서 확인하세요.

Amazon SNS로 SMS 메시징에 대한 발신자 ID 요청

Important

Amazon SNS에서 SMS 메시징을 처음 사용하는 경우 SMS 사용 사례의 예상 수요에 맞게 월별 SMS 지출 임계값을 요청하세요. 기본적으로 월별 지출 임계값은 $1.00(USD)입니다. 발신자 ID에 대한 요청이 포함된 동일한 지원 사례에서 지출 임계값 증가를 요청할 수 있습니다. 원한다면 별도의 사례를 개설해도 됩니다. 자세한 내용은 Amazon SNS에 대한 월별 SMS 지출 할당량 증가 요청 섹션을 참조하세요.

SMS 메시징에서 발신자 ID는 수신자의 디바이스에서 메시지 발신자로 표시되는 이름입니다. 발신자 ID는 메시지 수신자에게 자신을 식별하는 유용한 방법입니다.

발신자 ID에 대한 지원은 국가 별로 다릅니다. 예를 들어, 미국의 이동 통신 업체는 발신자 ID를 전혀 지원하지 않지만 인도의 이동 통신 업체는 발신자에게 발신자 ID 사용을 요구합니다. 발신자 ID를 지원하는 국가 의 전체 목록은 지원되는 국가 및 리전에서 확인하세요.

> **Important**
>
> 일부 국가에서는 메시지를 보내기 전에 발신자 ID를 등록해야 합니다. 국가 에 따라 이 등록 프로세스는 몇 주가 걸릴 수 있습니다. 사전 등록된 발신자 ID가 필요한 국가 은 지원되는 국가 페이지의 표에 나와 있습니다.
>
> SMS 메시지에 대해 여러 AWS 계정에서 동일한 발신자 ID를 사용하고 등록할 수 있습니다. 엔터프라이즈 지원을 받고 있고 여러 계정에 여러 템플릿을 등록하는 경우 아래 단계를 따르고 테크니컬 어카운트 관리자와 협력하여 온보딩 경험이 조정되도록 합니다.

발신자 ID가 지원되는 국가 의 수신자에게 메시지를 보내고, 해당 국가 에서 발신자 ID를 등록할 필요가 없는 경우 추가 단계를 수행할 필요가 없습니다. 발신자 ID 값이 포함된 메시지를 즉시 보낼 수 있습니다.

발신자 ID 등록이 필요한 국가 으로 메시지를 보내려는 경우에만 이 페이지의 절차를 완료하면 됩니다.

1단계: Amazon SNS SMS 사례 열기

발신자 ID가 필요한 국가에서 수신자에게 메시지를 보내려는 경우 AWS 지원 센터에서 새 사례를 생성하여 발신자 ID를 요청할 수 있습니다.

> **Note**
>
> 발신자 ID가 허용되지만 필수 사항이 아닌 국가에서 수신자에게 메시지를 보내려는 경우 지원 센터에서 사례를 열 필요가 없습니다. 발신자 ID를 사용하는 메시지는 즉시 보낼 수 있습니다.

발신자 ID를 요청하려면

1. https://console.aws.amazon.com/에서 AWS Management Console에 로그인합니다.
2. 지원 메뉴에서 지원 센터를 선택합니다.
3. 지원 사례 창에서 사례 생성을 선택합니다.
4. 서비스 한도 증가를 원하시나요? 링크를 선택한 후 다음 작업을 완료합니다.
   - Limit type(한도 유형)에서 Pinpoint SMS를 선택합니다.
   - (선택 사항) SMS 메시지를 보낼 사이트 또는 앱 링크 제공에서 대상 구성원이 SMS 메시지 수신을 선택한 웹 사이트 또는 애플리케이션을 지정합니다.

- (선택 사항) 보낼 메시지의 유형에서 긴 코드를 사용하여 보낼 메시지의 유형을 선택합니다.
  - 일회용 암호 – 고객이 웹 사이트 또는 애플리케이션에서 인증을 위해 사용해야 하는 암호가 담긴 메시지입니다.
  - 프로모션 – 비즈니스 또는 서비스(예: 특가 행사, 공지 사항)를 홍보하는 중요하지 않은 메시지입니다.
  - 트랜잭션 – 고객 트랜잭션을 지원하는 중요한 정보 메시지(예: 주문 확인, 계정 알림)입니다. 트랜잭션 메시지에 홍보 또는 마케팅 콘텐츠를 포함해서는 안 됩니다.
- (선택 사항) 메시지를 보낼 AWS 리전에서 SMS 메시지를 보낼 AWS 리전을 선택합니다.
- (선택 사항) 메시지를 보낼 국가에 발신자 ID를 등록할 국가 을 입력합니다. 발신자 ID 지원 및 발신자 ID 등록 요구 사항은 국가 별로 다릅니다. 자세한 내용은 지원되는 국가 및 리전 섹션을 참조하세요.

  국가 목록이 이 텍스트 상자에 허용된 문자 수를 초과하는 경우 대신 사례 설명 섹션에 국가을 나열할 수 있습니다.
- (선택 사항) 고객이 메시지 수신에 옵트인하는 방법에서 고객이 메시지 수신에 옵트인하는 방법에 대한 설명을 제공합니다.
- (선택 사항) 고객에게 메시지를 보내는 데 사용할 메시지 템플릿을 제공하십시오 필드에 사용할 메시지 템플릿을 모두 포함시킵니다.

5. Requests(요청)에서 다음 섹션을 작성합니다.

   - 리전에서 발신자 ID의 AWS 리전을 선택합니다.

     > **Note**
     >
     > 요청 섹션에는 리전이 필요합니다. 사례 세부 정보 섹션에 이 정보를 제공했더라도 여기에도 포함해야 합니다.

   - 리소스 유형에서 General Limits(일반 한도)를 선택합니다.
   - 한도에서 SMS 프로덕션 액세스를 선택합니다.
6. 새 한도 값에서 요청할 발신자 ID의 번호를 선택합니다. 일반적으로 이 값은 **1**입니다.
7. (선택 사항) 추가 요청을 제출하려면 다른 요청 추가를 선택합니다. 필요한 정보는 Amazon SNS로 SMS 메시징 지원 요청의 다른 단원을 참조하십시오.
8. Case description(사례 설명)의 Use case description(사용 사례 설명)에 다음 세부 정보를 입력합니다.

- 등록하려는 발신자 ID입니다.
- SMS 메시지에 사용하려는 템플릿입니다.
- 한 달에 각 수신자에게 보내려는 메시지 수입니다.
- 고객이 메시지를 수신하도록 옵트인하는 방법에 대한 정보입니다.
- 회사 또는 조직의 이름입니다.
- 회사 또는 조직과 관련된 주소입니다.
- 회사 또는 조직의 기반이 되는 국가입니다.
- 회사 또는 조직의 전화 번호입니다.
- 회사 또는 조직의 웹 사이트 URL입니다.

9. 연락처 옵션의 선호하는 문의 언어에서 이 사례에 대한 커뮤니케이션을 수신할 언어를 선택합니다.
10. 마쳤으면 Submit(제출)을 선택합니다.

요청이 접수되면 24시간 안에 첫 번째 회신을 합니다. 추가 정보를 요청하기 위해 연락을 드릴 수도 있습니다. 발신자 ID를 제공할 수 있는 경우, 제공에 필요한 예상 소요 시간을 알려 드립니다.

시스템이 원치 않는 콘텐츠 또는 악성 콘텐츠를 전송하지 않도록 하기 위해 각 요청을 신중하게 고려해야 합니다. 정책에 맞지 않는 사용 사례인 경우, 요청에 대한 권한을 부여하지 않을 수도 있습니다.

2단계: Amazon SNS 콘솔에서 SMS 설정 업데이트

발신자 ID 확보 프로세스가 완료되면 해당 사례에 회신합니다. 이 알림을 받으면, 계정을 사용해 전송한 모든 메시지의 기본 발신자 ID로써 발신자 ID를 사용하기 위해 Amazon SNS를 구성하도록 이 섹션의 단계를 완료합니다. 또는 메시지를 게시할 때 사용할 발신자 ID를 지정할 수 있습니다.

1. Amazon SNS 콘솔에 로그인합니다.
2. 탐색 창에서 모바일, 문자 메시지(SMS)를 차례로 선택합니다.
3. 문자 메시지 기본 설정 섹션에서 편집을 선택합니다.
4. 세부 정보 섹션에 있는 기본 발신자 ID 필드에, 계정의 모든 메시지에서 기본 사용하기 위해 제공된 발신자 ID를 입력합니다.
5. 변경 작업을 마치면 변경 사항 저장을 선택합니다.

다음 단계

Amazon SNS 콘솔에서 발신자 ID를 등록하고 설정을 업데이트했습니다. 이제 Amazon SNS를 사용하여 발신자 ID로 SMS 메시지를 전송할 수 있습니다. 지원되는 국가에 있는 SMS 수신자의 디바이스에 발신자 ID가 메시지 발신자로 표시됩니다. 메시지를 게시할 때 다른 발신자 ID를 사용할 경우, 여기서 구성된 기본 ID를 재정의합니다.

Amazon SNS에 대한 월별 SMS 지출 할당량 증가 요청

Amazon SNS에서는 계정을 사용해 SMS를 전송하여 발생하는 최대 월별 비용을 관리하는 데 도움이 되는 지출 할당량을 제공합니다. 지출 할당량은 악의적인 공격 발생 시의 위험을 제한하고, 업스트림 애플리케이션이 예상보다 많은 메시지를 전송하지 못하도록 차단합니다. SMS 메시지를 보내면 이번 달 지출 할당량을 초과하는 비용이 발생할 것으로 판단되는 경우 SMS 메시지 게시를 중지하도록 Amazon SNS를 구성할 수 있습니다.

운영에 영향을 미치지 않도록 프로덕션 워크로드를 지원할 수 있을 만큼 지출 할당량을 높게 요청하는 것이 좋습니다. 자세한 내용은 1단계: Amazon SNS SMS 사례 생성을 참조하세요. 할당량을 받으면 2단계: SMS 설정 업데이트에 설명된 대로 전체 할당량이나 그보다 작은 값을 적용하여 위험을 관리할 수 있습니다. 작은 값을 적용하면 필요에 따라 크기 조정 옵션을 사용하여 월별 지출을 제어할 수 있습니다.

> **Important**
>
> Amazon SNS는 분산 시스템이므로 지출 할당량이 초과되는 경우 몇 분 이내에 SMS 메시지 전송을 중지합니다. 이 시간 동안 SMS 메시지를 계속 보내면 할당량을 초과하는 비용이 발생할 수 있습니다.

모든 새 계정의 지출 할당량은 매월 1.00 USD로 설정됩니다. 이것은 Amazon SNS의 메시지 전송 기능을 테스트할 수 있도록 마련된 할당량입니다. 계정의 SMS 지출 할당량에 대한 증가를 요청하려면 AWS 지원 센터에서 할당량 증가 사례를 개설하세요.

1단계: Amazon SNS SMS 사례 열기

AWS 지원 센터에서 서비스 할당량 증가 사례를 열어 월별 지출 할당량에 대한 증가를 요청할 수 있습니다.

> **Note**
>
> 요청 양식의 일부 필드는 “선택 사항”으로 표시되어 있습니다. 그러나 AWS Support에서 요청을 처리하려면 아래의 단계에 언급된 정보가 전부 필요합니다. 필요한 정보를 모두 제공하지 않으면 요청 처리가 지연될 수 있습니다.

1. https://console.aws.amazon.com/deepracer에서 AWS Management Console에 로그인합니다.
2. 지원 메뉴에서 지원 센터를 선택합니다.
3. 지원 사례 창에서 사례 생성을 선택합니다.
4. 서비스 한도 증가를 원하시나요? 링크를 선택한 후 다음 작업을 완료합니다.
   - 한도 유형에서 SNS 문자 메시지를 선택합니다.
   - (선택 사항) SMS 메시지를 보낼 사이트 또는 앱의 링크 제공에 SMS 메시지를 보낼 웹 사이트, 애플리케이션 또는 서비스에 대한 정보를 입력합니다.
   - (선택 사항) 보낼 메시지의 유형에서 긴 코드를 사용하여 보낼 메시지의 유형을 선택합니다.
     - 일회용 암호 – 고객이 웹 사이트 또는 애플리케이션에서 인증을 위해 사용해야 하는 암호가 담긴 메시지입니다.
     - 프로모션 – 비즈니스 또는 서비스(예: 특가 행사, 공지 사항)를 홍보하는 중요하지 않은 메시지입니다.
     - 트랜잭션 – 고객 트랜잭션을 지원하는 중요한 정보 메시지(예: 주문 확인, 계정 알림)입니다. 트랜잭션 메시지에 홍보 또는 마케팅 콘텐츠를 포함해서는 안 됩니다.
   - (선택 사항) 메시지를 보낼 AWS 리전에서 메시지를 보낼 리전을 선택합니다.
   - (선택 사항) 메시지를 보낼 국가에 단축 코드를 구입할 국가 또는 리전을 입력합니다.
   - (선택 사항) 고객이 메시지 수신에 옵트인하는 방법에서 옵트인 프로세스에 대한 세부 정보를 제공합니다.
   - (선택 사항) 고객에게 메시지를 보내는 데 사용할 메시지 템플릿을 제공하십시오 필드에 사용할 메시지 템플릿을 포함시킵니다.
5. Requests(요청)에서 다음 섹션을 작성합니다.
   - 리전에서 메시지를 보낼 리전을 선택합니다.

> **Note**
>
> 요청 섹션에는 리전이 필요합니다. 사례 세부 정보 섹션에 이 정보를 제공했더라도 여기에도 포함해야 합니다.

- 리소스 유형에서 General Limits(일반 한도)를 선택합니다.
- 한도에서 Account Spend Threshold Increase(계정 지출 임계값 증가)를 선택합니다.

6. 새 한도 값에 매월 SMS에 지출할 수 있는 최대 금액(USD)을 입력합니다.
7. Case description(사례 설명)의 Use case description(사용 사례 설명)에 다음 세부 정보를 입력합니다.

   - SMS 메시지를 보내는 회사 또는 서비스의 웹 사이트 또는 앱
   - 웹 사이트 또는 앱에서 제공되는 서비스, 서비스에 대한 SMS 메시지의 기여 방법
   - 웹 사이트, 앱 또는 다른 위치에서 사용자가 SMS 메시지를 자발적으로 수신하기 위해 등록하는 방법

   요청된 지출 할당량(New quota value(새 할당량 값)에 대해 지정한 값)이 10,000 USD를 초과할 경우 메시지를 주고 받을 각 국가에 대해 다음과 같은 추가 세부 정보를 제공합니다.

   - 발신자 ID 또는 단축 코드를 사용할지 여부. 발신자 ID를 사용할 경우 다음을 제공합니다.
     - 발신자 ID
     - 발신자 ID를 해당 국가의 무선통신 사업자에 등록할지 여부
   - 메시징에 대한 최대 예상 TPS(초당 트랜잭션 수)
   - 평균 메시지 크기
   - 국가에 전송할 메시지의 템플릿
   - (선택 사항) 문자 인코딩 요구 사항(있는 경우)
8. (선택 사항) 추가 요청을 제출하려면 다른 요청 추가를 선택합니다. 여러 요청을 포함할 경우 각 요청에 대해 필요한 정보를 제공합니다. 필요한 정보는 Amazon SNS로 SMS 메시징 지원 요청의 다른 단원을 참조하십시오.
9. 연락처 옵션의 선호하는 문의 언어에서 이 사례에 대한 커뮤니케이션을 수신할 언어를 선택합니다.
10. 마쳤으면 Submit(제출)을 선택합니다.

AWS Support 팀은 24시간 이내에 요청에 대한 초기 응답을 제공합니다.

시스템이 원치 않는 콘텐츠 또는 악성 콘텐츠를 전송하지 않도록 하기 위해 각 요청을 신중하게 고려합니다. 가능한 경우 이 24시간 기간 내에 요청이 승인될 것입니다. 그러나 사용자의 추가 정보가 필요한 경우 요청을 해결하는 데 시간이 오래 걸릴 수도 있습니다.

정책에 맞지 않는 사용 사례인 경우, 요청에 대한 권한을 부여하지 않을 수도 있습니다.

2단계: Amazon SNS 콘솔에서 SMS 설정 업데이트

월별 지출 할당량이 증가되었다는 알림을 받으면 Amazon SNS 콘솔에서 해당 계정의 지출 할당량을 조정해야 합니다.

> **Important**
>
> 다음 단계를 완료해야 합니다. 그렇지 않으면 SMS 지출 한도가 증가되지 않습니다.

콘솔에서 지출 할당량을 조정하려면

1. Amazon SNS 콘솔에 로그인합니다.
2. 왼쪽 탐색 메뉴를 열고, 모바일을 확장한 다음, 텍스트 메시징(SMS)을 선택합니다.
3. 모바일 문자 메시지(SMS) 페이지의 문자 메시지 기본 설정 섹션에서 편집을 선택합니다.
4. 문자 메시지 기본 설정 편집 페이지의 세부 정보에서 계정 지출 한도에 신규 SMS 지출 한도를 입력합니다.

   > **Note**
   >
   > 입력한 값이 기본 지출 한도보다 크다는 경고가 나타날 수 있습니다. 이 텍스트는 무시할 수 있습니다.

5. [Save changes]를 선택합니다.

   > **Note**
   >
   > “잘못된 파라미터” 오류가 발생할 경우, AWS Support에서 연락처를 확인한 뒤 올바른 신규 SMS 지출 한도를 입력했는지 확인합니다. 문제가 계속 발생하면 AWS 지원 센터에 사례를 개설합니다.

SMS 메시징 기본 설정 지정

Amazon SNS를 사용하여 SMS 메시징에 대한 기본 설정을 지정합니다. 예를 들어 비용 또는 안정성을 위해 전송을 최적화할지 여부, 월별 지출 한도, 전송을 로그하는 방법 및 일일 SMS 사용 보고서를 구독할지 여부를 지정할 수 있습니다.

이러한 기본 설정은 계정에서 보내는 모든 SMS 메시지에 적용되지만, 개별 메시지를 전송할 때 일부 설정을 무시할 수 있습니다. 자세한 정보는 휴대폰에 게시을 참조하세요.

주제

- 를 사용하여 SMS 메시징 기본 설정 지정 AWS Management Console
- 환경설정 (AWS SDK) 지정

를 사용하여 SMS 메시징 기본 설정 지정 AWS Management Console

1. Amazon SNS 콘솔에 로그인합니다.
2. SMS 메시징을 지원하는 리전을 선택합니다.
3. 탐색 창에서 모바일, 문자 메시지(SMS)를 선택합니다.
4. 모바일 문자 메시지(SMS) 페이지의 문자 메시지 기본 설정 섹션에서 편집을 선택합니다.
5. Edit text messaging preferences(문자 메시지 기본 설정 편집) 페이지의 세부 정보 섹션에서 다음을 수행합니다.

   a. 기본 메시지 유형에서 다음 중 하나를 선택합니다.

      - 프로모션 – 중요하지 않은 메시지(예: 마케팅)입니다. Amazon SNS는 최소 비용이 발생하도록 메시지 전송을 최적화합니다.
      - 트랜잭션(기본값) – 고객 트랜잭션을 지원하는 중요한 메시지입니다(예: 멀티 팩터 인증을 위한 일회용 암호). Amazon SNS는 최고의 안정성을 달성하도록 메시지 전송을 최적화합니다.

      프로모션 및 트랜잭션 메시지에 대한 요금 정보는 글로벌 SMS 요금에서 확인하세요.

   b. (선택 사항) 계정 지출 한도에 매월 SMS 메시지에 지출하려는 최대 금액(USD)을 입력합니다.

> **Important**
>
> - 기본적으로 지출 할당량은 1.00USD로 설정됩니다. 서비스 할당량을 늘리려면 요청을 제출하세요.
> - 콘솔에서 설정된 금액이 서비스 할당량을 초과할 경우 Amazon SNS에서 SMS 메시지 게시를 중지합니다.
> - Amazon SNS는 분산 시스템이므로 초과되는 지출 할당량(분) 이내에 SMS 메시지 전송을 중지합니다. 이 기간 사이에 SMS 메시지 전송을 계속하는 경우 할당량 초과 비용이 발생할 수 있습니다.

6. (선택 사항) 기본 발신자 ID에 기업 브랜드와 같은 사용자 지정 ID를 입력합니다. 이 ID는 수신하는 디바이스의 발신자로 표시됩니다.

> **Note**
>
> 발신자 ID에 대한 지원은 국가별로 다릅니다.

7. (선택 사항) Amazon S3 bucket name for usage reports(사용 보고서용 Amazon S3 버킷 이름)의 이름을 입력합니다.

> **Note**
>
> S3 버킷 정책은 Amazon SNS에 쓰기 액세스 권한을 부여해야 합니다.

8. 변경 사항 저장를 선택합니다.

환경설정 (AWS SDK) 지정

AWS SDK 중 하나를 사용하여 SMS 기본 설정을 지정하려면 Amazon SNS API의 `SetSMSAttributes` 요청에 해당하는 해당 SDK의 작업을 사용하십시오. 이 요청을 통해 월 지출 할당량 및 기본 SMS 유형(프로모션 또는 트랜잭션) 등의 다양한 SMS 속성에 값을 할당합니다. 모든 SMS 속성은 Amazon Simple Notification Service API 참조의 SetSMSAttributes를 참조하세요.

다음 코드 예제는 `SetSMSAttributes`의 사용 방법을 보여 줍니다.

C++

SDK for C++

> **Note**
>
> 자세한 내용은 여기에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon SNS를 사용하여 DefaultSMSType 속성을 설정하는 방법입니다.

```
//! Set the default settings for sending SMS messages.
/*!
  \param smsType: The type of SMS message that you will send by default.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::setSMSType(const Aws::String &smsType,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SetSMSAttributesRequest request;
    request.AddAttributes("DefaultSMSType", smsType);

    const Aws::SNS::Model::SetSMSAttributesOutcome outcome =
 snsClient.SetSMSAttributes(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "SMS Type set successfully " << std::endl;
    }
    else {
        std::cerr << "Error while setting SMS Type: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 SetSMSAttributes를 참조하십시오.

CLI

AWS CLI

SMS 메시지 속성을 설정하려면

다음 `set-sms-attributes` 예제에서는 SMS 메시지의 기본 발신자 ID를 `MyName`으로 설정합니다.

```
aws sns set-sms-attributes \
    --attributes DefaultSenderID=MyName
```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 참조의 SetSMSAttributes를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String,
 String> attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                    .attributes(attributes)
                    .build();

            SetSmsAttributesResponse result =
 snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ".
 Status was "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 SetSMSAttributes를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

별도의 모듈에서 클라이언트를 생성하고 내보냅니다.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
import { SetSMSAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {"Transactional" | "Promotional"} defaultSmsType
 */
export const setSmsType = async (defaultSmsType = "Transactional") => {
  const response = await snsClient.send(
    new SetSMSAttributesCommand({
      attributes: {
        // Promotional – (Default) Noncritical messages, such as marketing
 messages.
        // Transactional – Critical messages that support customer transactions,
        // such as one-time passcodes for multi-factor authentication.
        DefaultSMSType: defaultSmsType,
      },
    }),
  );
  console.log(response);
  // {
```

```
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '1885b977-2d7e-535e-8214-e44be727e265',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 SetSMSAttributes를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->SetSMSAttributes([
        'attributes' => [
            'DefaultSMSType' => 'Transactional',
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
```

```
    error_log($e->getMessage());
}
```

- 자세한 정보는 AWS SDK for PHP 개발자 안내서를 참조하십시오.
- API에 대한 세부 정보는 AWS SDK for PHP API 참조의 SetSMSAttributes를 참조하세요.

SMS 메시지 전송

이 섹션에서는 SMS 메시지를 보내는 방법에 대해 설명합니다.

주제

- 주제에 게시
- 휴대폰에 게시

주제에 게시

해당 전화번호에서 Amazon SNS 주제를 구독하면 단일 SMS 메시지를 수많은 전화번호에 한 번에 게시할 수 있습니다. SNS 주제는 구독자를 추가한 다음 해당 구독자 모두에게 메시지를 게시할 수 있는 통신 채널입니다. 사용자가 구독을 취소하거나 구독자가 AWS 계정의 SMS 메시지 수신을 옵트아웃할 때까지 구독자는 주제에 게시된 모든 메시지를 수신합니다.

주제

- 주제로 메시지 전송(콘솔)
- 주제로 메시지 전송(AWS SDK)

주제로 메시지 전송(콘솔)

주제를 생성하려면

SMS 메시지를 전송할 주제가 아직 없는 경우 다음 단계를 완료합니다.

1. Amazon SNS 콘솔에 로그인합니다.
2. 콘솔 메뉴에서 SMS 메시징을 지원하는 AWS 리전을 선택합니다.
3. 탐색 창에서 주제를 선택합니다.
4. 주제 페이지에서 주제 생성을 선택합니다.

5. 주제 생성 페이지의 세부 정보에서 다음을 수행합니다.

   a. 유형에서 표준을 선택합니다.

   b. 이름에 주제 이름을 입력합니다.

   c. (선택 사항) 표시 이름에 SMS 메시지의 사용자 지정 접두사를 입력합니다. 주제에 메시지를 전송할 때 Amazon SNS는 오른쪽 꺾쇠괄호(>) 및 공백 다음에 표시 이름을 접두사로 추가합니다. 표시 이름은 대/소문자로 구분하지 않으며 Amazon SNS는 표시 이름을 대문자로 전환합니다. 예를 들어, 주제의 표시 이름이 `MyTopic`이고 메시지가 `Hello World!`인 경우 메시지는 다음과 같이 나타납니다.

   ```
   MYTOPIC> Hello World!
   ```

6. 주제 생성을 선택합니다. 주제 이름과 Amazon 리소스 이름(ARN)은 주제 페이지에 나타납니다.

SMS 구독을 생성하려면

구독을 사용하여 메시지를 주제에 한 번 게시하여 SMS 메시지를 여러 수신자에게 전송할 수 있습니다.

> **Note**
>
> Amazon SNS를 사용하여 SMS 메시지를 보내기 시작하는 경우 AWS 계정이 SMS 샌드박스에 있습니다. SMS 샌드박스는 SMS 발신자로서의 평판을 위협하지 않고 Amazon SNS 기능을 사용해 볼 수 있는 안전한 환경을 제공합니다. 계정이 SMS 샌드박스에 있는 동안 Amazon SNS의 모든 기능을 사용할 수 있지만 SMS 메시지는 확인된 대상 전화번호로만 보낼 수 있습니다. 자세한 내용은 SMS 샌드박스 섹션을 참조하세요.

1. Amazon SNS 콘솔에 로그인합니다.
2. 탐색 창에서 구독을 선택합니다.
3. 구독 페이지에서 구독 생성을 선택합니다.
4. 구독 생성 페이지의 세부 정보에서 다음을 수행합니다.

   a. 주제 ARN에 SMS 메시지를 보낼 주제의 Amazon 리소스 이름(ARN)을 입력하거나 선택합니다.

   b. 프로토콜에서 SMS를 선택합니다.

   c. 엔드포인트에서 주제를 구독할 전화번호를 입력합니다.

5. 구독 생성을 선택합니다. 구독 정보는 구독 페이지에 나타납니다.

   전화번호를 더 추가하려면 다음 단계를 반복합니다. 다른 유형의 구독을 추가할 수도 있습니다(예: 이메일).

메시지를 전송하려면

메시지를 주제에 게시하면 Amazon SNS는 주제를 구독하는 모든 전화번호로 해당 메시지를 전송합니다.

1. Amazon SNS 콘솔의 주제 페이지에서 SMS 메시지를 보낼 주제의 이름을 선택합니다.
2. 주제 세부 정보 페이지에서 [Publish to topic]을 선택합니다.
3. 주제에 메시지 게시 페이지의 메시지 세부 정보에서 다음을 수행합니다.

   a. 주제에 이메일 구독이 포함되고 이메일 및 SMS 구독에 모두 게시하려는 경우가 아닌 한 제목 필드를 비워 둡니다. Amazon SNS는 귀하가 이메일 제목 줄로 입력한 제목을 사용합니다.

   b. (선택 사항) 유지 시간(TTL)에 Amazon SNS가 SMS 메시지를 모바일 애플리케이션 엔드포인트 구독자에게 전송해야 하는 시간(초)을 입력합니다.
4. 메시지 본문에서 다음을 수행합니다.

   a. 메시지 구조에서 모든 전송 프로토콜에 대해 동일한 페이로드를 선택하여 주제를 구독하는 모든 프로토콜 유형에 동일한 메시지를 보냅니다. 또는 각 전송 프로토콜에 대한 사용자 지정 페이로드를 선택하여 다양한 프로토콜 유형의 구독자에 대한 메시지를 사용자 지정합니다. 예를 들어, 전화번호 구독자에 대한 기본 메시지와 이메일 구독자에 대한 사용자 지정 메시지를 입력할 수 있습니다.

   b. 엔드포인트로 보낼 메시지 본문에 메시지 또는 전송 프로토콜당 사용자 지정 메시지를 입력합니다.

      주제에 표시 이름이 있는 경우 Amazon SNS는 해당 이름을 메시지에 추가하므로 메시지 길이가 증가합니다. 표시 이름 길이는 이름에 포함된 문자 수 더하기 Amazon SNS가 추가하는 오른쪽 꺾쇠괄호(>) 및 공백에 해당하는 2자입니다.

      SMS 메시지의 크기 할당량에 대한 자세한 내용은 휴대폰에 게시에서 확인하세요.
5. (선택 사항) 메시지 속성에 타임스탬프, 서명, ID와 같은 메시지 메타데이터를 추가합니다.
6. 메시지 게시를 선택합니다. Amazon SNS는 SMS 메시지를 전송하고 성공 메시지를 표시합니다.

주제로 메시지 전송(AWS SDK)

AWS SDK를 사용하려면 자격 증명을 사용하여 구성해야 합니다. 자세한 정보는 AWS SDK 및 도구 참조 가이드의 공유 구성 및 자격 증명 파일을 참조하세요.

다음 코드 예제에서는 작업 방법을 보여줍니다.

- Amazon SNS 주제를 생성합니다.
- 전화번호를 주제에 구독시킵니다.
- 모든 구독 전화번호가 한 번에 메시지를 받을 수 있도록 주제에 SMS 메시지를 게시합니다.

Java

SDK for Java 2.x

> **Note**
>
> GitHub에 더 많은 내용이 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

주제를 만들고 해당 ARN을 반환합니다.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:    <topicName>

                Where:
                   topicName - The name of the topic to create (for example,
 mytopic).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                    .name(topicName)
                    .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

주제에 엔드포인트를 구독 설정합니다.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn> <phoneNumber>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   phoneNumber - A mobile phone number that receives
 notifications (for example, +1XXX5550100).
                """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subTextSNS(snsClient, topicArn, phoneNumber);
        snsClient.close();
    }
```

```
    public static void subTextSNS(SnsClient snsClient, String topicArn, String
 phoneNumber) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("sms")
                    .endpoint(phoneNumber)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() +
 "\n\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

발신자의 ID, 최고 가격 및 유형과 같은 메시지의 속성을 설정합니다. 메시지 속성은 선택 사항입니다.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetSMSAttributes {
```

```
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String,
 String> attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                    .attributes(attributes)
                    .build();

            SetSmsAttributesResponse result =
 snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ".
 Status was "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

주제에 메시지를 게시합니다. 메시지는 모든 구독자에게 전송됩니다.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <phoneNumber>

                Where:
                   message - The message text to send.
                   phoneNumber - The mobile phone number to which a message is
 sent (for example, +1XXX5550100).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }

    public static void pubTextSMS(SnsClient snsClient, String message, String
 phoneNumber) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .phoneNumber(phoneNumber)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
```

```
                    .println(result.messageId() + " Message sent. Status was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

휴대폰에 게시

Amazon SNS를 사용하여 전화번호에서 Amazon SNS 주제를 구독하지 않고도 휴대폰으로 SMS 메시지를 직접 전송할 수 있습니다.

> **Note**
>
> 하나의 메시지를 여러 전화번호에 한 번에 전송하려는 경우에도 전화번호에서 주제를 구독하면 유용할 수 있습니다. 주제에 SMS 메시지 게시에 대한 자세한 내용은 주제에 게시에서 확인하세요.

메시지를 전송할 때 메시지가 비용에 맞게 최적화되는지 또는 안정적 전송을 위해 최적화되는지를 제어할 수 있습니다. 발신자 ID 또는 발신 번호를 지정할 수도 있습니다. Amazon SNS API 또는 AWS SDK를 사용하여 프로그래밍 방식으로 메시지를 보내는 경우 메시지 전송의 최고 가격을 지정할 수 있습니다.

각 SMS 메시지는 140바이트까지 포함할 수 있으며 문자 할당량은 인코딩 체계에 따라 달라집니다. 예를 들어 SMS 메시지에는 다음과 같이 포함될 수 있습니다.

- 160 GSM 문자
- 140 ASCII 문자
- 70 UCS-2 문자

크기 할당량을 초과하는 메시지를 게시하면 Amazon SNS에서 해당 메시지를 각각 크기 할당량 내에 맞춘 여러 메시지로 보냅니다. 메시지는 단어 중간에 잘리지 않고 대신 전체 단어의 경계선에서 잘립니다. 단일 SMS 게시 작업에 대한 전체 크기 할당량은 1,600바이트입니다.

SMS 메시지를 보낼 때 국제 통신에 사용되는 표준 전화번호 구조인 E.164 형식을 사용하여 전화번호를 지정합니다. 이 형식을 따르는 전화번호는 최대 15자리 숫자를 사용할 수 있으며 더하기 기호(+) 및 국가 코드가 접두사로 추가됩니다. 예를 들어, E.164 형식의 미국 전화번호는 +1XXX5550100로 표시됩니다.

주제

- 메시지 전송(콘솔)
- 메시지 전송 (SDK)AWS

메시지 전송(콘솔)

1. Amazon SNS 콘솔에 로그인합니다.
2. 콘솔 메뉴에서 SMS 메시징을 지원하는AWS 리전을 선택합니다.
3. 탐색 창에서 문자 메시지(SMS)를 선택합니다.
4. 모바일 문자 메시징(SMS) 페이지에서 문자 메시지 게시를 선택합니다.
5. SMS 메시지 게시의 메시지 유형에서 다음 중 하나를 선택합니다.
   - 프로모션 – 중요하지 않은 메시지입니다(예: 마케팅 메시지).
   - 트랜잭션 – 고객 트랜잭션을 지원하는 중요한 메시지입니다(예: 멀티 팩터 인증을 위한 일회용 암호).

   > **Note**
   >
   > 이 메시지 수준 설정은 계정 수준 기본 메시지 유형을 재정의합니다. 모바일 문자 메시징(SMS) 페이지의 문자 메시징 기본 설정 섹션에서 계정 수준의 기본 메시지 유형을 설정할 수 있습니다.

   프로모션 및 트랜잭션 메시지에 대한 요금 정보는 전 세계 SMS 요금을 참조하세요.
6. 대상 전화번호에 메시지를 전송하려는 전화번호를 입력합니다.
7. 메시지에 전송할 메시지를 입력합니다.
8. (선택 사항) 발신 자격 증명에서 수신자에게 귀하를 식별하는 방법을 지정합니다.

- 발신자 ID에 공백 없이 최소 1자의 문자를 포함하여 3~11자의 영숫자로 이루어진 사용자 지정 ID를 입력합니다. 발신자 ID는 수신 디바이스에 메시지 발신자로 표시됩니다. 예를 들어, 기업 브랜드를 사용하여 메시지 소스를 더 인식하기 쉽게 만들 수 있습니다.

  발신자 ID에 대한 지원은 국가 및/또는 리전별로 다릅니다. 예를 들어, 미국 전화번호로 전달된 메시지에는 발신자 ID가 표시되지 않습니다. 발신자 ID를 지원하는 국가 및 리전을 확인하려면 지원되는 국가 및 리전에서 확인하세요.

  발신자 ID를 지정하지 않으면 다음 중 하나가 발신 자격 증명으로 표시됩니다.

  - 긴 코드를 지원하는 국가에서는 긴 코드가 표시됩니다.
  - 발신자 ID만 지원되는 국가에서는 NOTICE(알림)가 표시됩니다.

  이 메시지 수준 발신자 ID는 [Text messaging preferences] 페이지에서 설정하는 기본 발신자 ID보다 우선적으로 적용됩니다.

- 발신 번호를 지정하려면 수신자 디바이스에 발신자의 전화번호로 표시할 5-14개의 숫자 문자열을 입력합니다. 이 문자열은 대상 국가의 AWS 계정 에서 구성된 발신 번호와 일치해야 합니다. 발신 번호는 10DLC 번호, 무료 전화 번호, 긴 코드 또는 단축 코드일 수 있습니다. person-to-person 자세한 정보는 SMS 메시지에 대한 발신 자격 증명을 참조하세요.

  발신 번호를 지정하지 않는 경우, Amazon SNS는 구성에 따라 SMS 문자 메시지에 사용할 발신 번호를 선택합니다. AWS 계정

9. 인도의 수신자에게 SMS 메시지를 보내는 경우 국가별 속성을 확장하고 다음 속성을 지정합니다.

   - 엔터티 ID – 인도의 수신자에게 SMS 메시지를 보내기 위한 엔터티 ID 또는 보안 주체 엔터티(PE) ID입니다. 이 ID는 인도 통신 규제 당국(TRAI)에서 TRAI에 등록한 엔터티를 식별하기 위해 제공하는 1-50자의 고유한 문자열입니다.
   - 템플릿 ID – 인도에 있는 수신자에게 SMS 메시지를 보내기 위한 템플릿 ID입니다. 이 ID는 TRAI에 등록한 템플릿을 식별하는 1–50자의 고유한 TRAI 제공 문자열입니다. 템플릿 ID는 메시지에 대해 지정한 발신자 ID와 연결되어야 합니다.

   인도의 수신자에게 SMS 메시지를 보내는 방법에 대한 자세한 정보는 인도 발신자 ID 등록 요구 사항에서 확인하세요.

10. 메시지 게시를 선택합니다.

> **Tip**
>
> 발신 번호에서 SMS 메시지를 보내려면 Amazon SNS 콘솔 탐색 패널에서 발신 번호를 선택할 수도 있습니다. 기능 열에 SMS가 포함된 발신 번호를 선택한 다음 문자 메시지 게시를 선택합니다.

메시지 전송 (SDK)AWS

AWS SDK 중 하나를 사용하여 SMS 메시지를 보내려면 Amazon SNS API의 `Publish` 요청에 해당하는 해당 SDK의 API 작업을 사용하십시오. 이 요청을 사용하여 SMS 메시지를 전화 번호로 직접 전송할 수 있습니다. `MessageAttributes` 파라미터를 사용하여 다음 속성 이름의 값을 설정할 수 있습니다.

`AWS.SNS.SMS.SenderID`

공백 없이 최소 1자의 문자를 포함하여 3~11자의 영숫자 또는 하이픈(-)으로 이루어진 사용자 지정 ID입니다. 발신자 ID는 수신 디바이스에 메시지 발신자로 표시됩니다. 예를 들어, 기업 브랜드를 사용하여 메시지 소스를 더 인식하기 쉽게 만들 수 있습니다.

발신자 ID에 대한 지원은 국가 또는 리전별로 다릅니다. 예를 들어, 미국 전화번호로 전달된 메시지에는 발신자 ID가 표시되지 않습니다. 발신자 ID를 지원하는 국가 또는 리전 목록을 확인하려면 지원되는 국가 및 리전에서 확인하세요.

발신자 ID를 지정하지 않으면 지원되는 국가 또는 리전에서는 긴 코드가 발신자 ID로 표시됩니다. 영문자 발신자 ID가 필요한 국가 또는 리전의 경우 NOTICE가 발신자 ID로 표시됩니다.

이 메시지 수준 속성은 `SetSMSAttributes` 요청을 사용하여 설정하는 계정 수준 속성인 `DefaultSenderID`보다 우선적으로 적용됩니다.

`AWS.MM.SMS.OriginationNumber`

선택적 선행 더하기 기호(+)를 포함할 수 있는 5–14개의 숫자로 구성된 사용자 지정 문자열입니다. 이 숫자 문자열은 수신 디바이스에서 발신자의 전화번호로 나타납니다. 문자열은 대상 국가의 AWS 계정에 구성된 발신 번호와 일치해야 합니다. 출발 번호는 10DLC 번호, 무료 전화 번호, person-to-person (P2P) 긴 코드 또는 단축 코드일 수 있습니다. 자세한 정보는 발신 번호을 참조하세요.

시작 번호를 지정하지 않는 경우 Amazon SNS는 계정 구성을 기반으로 시작 번호를 선택합니다. AWS

`AWS.SNS.SMS.MaxPrice`

SMS 메시지를 전송하기 위해 지출할 의사가 있는 최대 금액(USD)입니다. Amazon SNS에서 메시지 전송 시 최고 가격을 초과하는 비용이 발생한다고 판단하면 메시지를 전송하지 않습니다.

month-to-date SMS 비용이 이미 속성에 설정된 할당량을 초과한 경우에는 이 속성이 적용되지 않습니다. `MonthlySpendLimit` `SetSMSAttributes` 요청을 사용하여 `MonthlySpendLimit` 속성을 설정할 수 있습니다.

메시지를 Amazon SNS 주제로 전송하는 경우 최고 가격은 주제를 구독하는 각 전화번호로 전송되는 각 메시지에 적용됩니다.

`AWS.SNS.SMS.SMSType`

전송하는 메시지 유형은 다음과 같습니다.

- `Promotional`(기본값) – 마케팅 메시지와 같이 중요하지 않은 메시지입니다.
- `Transactional` – 고객 트랜잭션을 지원하는 중요한 메시지입니다(예: 멀티 팩터 인증을 위한 일회용 암호).

이 메시지 수준 속성은 `SetSMSAttributes` 요청을 사용하여 설정하는 계정 수준 속성인 `DefaultSMSType`보다 우선적으로 적용됩니다.

`AWS.MM.SMS.EntityId`

이 속성은 인도의 수신자에게 SMS 메시지를 보낼 때만 필요합니다.

인도의 수신자에게 SMS 메시지를 보내기 위한 엔터티 ID 또는 보안 주체 엔터티(PE) ID입니다. 이 ID는 인도 통신 규제 당국(TRAI)에서 TRAI에 등록한 엔터티를 식별하기 위해 제공하는 1-50자의 고유한 문자열입니다.

`AWS.MM.SMS.TemplateId`

이 속성은 인도의 수신자에게 SMS 메시지를 보낼 때만 필요합니다.

인도에 있는 수신자에게 SMS 메시지를 보내기 위한 템플릿입니다. 이 ID는 TRAI에 등록한 템플릿을 식별하는 1–50자의 고유한 TRAI 제공 문자열입니다. 템플릿 ID는 메시지에 대해 지정한 발신자 ID와 연결되어야 합니다.

메시지 전송

다음 코드 예제는 Amazon SNS를 사용하여 SMS 메시지를 게시하는 방법을 보여줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> 자세한 내용은 여기에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
namespace SNSMessageExample
{
    using System;
    using System.Threading.Tasks;
    using Amazon;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    public class SNSMessage
    {
        private AmazonSimpleNotificationServiceClient snsClient;

        /// <summary>
        /// Initializes a new instance of the <see cref="SNSMessage"/> class.
        /// Constructs a new SNSMessage object initializing the Amazon Simple
        /// Notification Service (Amazon SNS) client using the supplied
        /// Region endpoint.
        /// </summary>
        /// <param name="regionEndpoint">The Amazon Region endpoint to use in
        /// sending test messages with this object.</param>
        public SNSMessage(RegionEndpoint regionEndpoint)
        {
            snsClient = new
 AmazonSimpleNotificationServiceClient(regionEndpoint);
        }

        /// <summary>
        /// Sends the SMS message passed in the text parameter to the phone
 number
        /// in phoneNum.
        /// </summary>
        /// <param name="phoneNum">The ten-digit phone number to which the text
```

```
        /// message will be sent.</param>
        /// <param name="text">The text of the message to send.</param>
        /// <returns>Async task.</returns>
        public async Task SendTextMessageAsync(string phoneNum, string text)
        {
            if (string.IsNullOrEmpty(phoneNum) || string.IsNullOrEmpty(text))
            {
                return;
            }

            // Now actually send the message.
            var request = new PublishRequest
            {
                Message = text,
                PhoneNumber = phoneNum,
            };

            try
            {
                var response = await snsClient.PublishAsync(request);
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error sending message: {ex}");
            }
        }
    }
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 Publish를 참조하십시오.

C++

SDK for C++

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Publish SMS: use Amazon Simple Notification Service (Amazon SNS) to send an
 SMS text message to a phone number.
 * Note: This requires additional AWS configuration prior to running example.
 *
 *  NOTE: When you start using Amazon SNS to send SMS messages, your AWS account
 is in the SMS sandbox and you can only
 *  use verified destination phone numbers. See https://docs.aws.amazon.com/sns/
latest/dg/sns-sms-sandbox.html.
 *  NOTE: If destination is in the US, you also have an additional restriction
 that you have use a dedicated
 *  origination ID (phone number). You can request an origination number using
 Amazon Pinpoint for a fee.
 *  See https://aws.amazon.com/blogs/compute/provisioning-and-using-10dlc-
origination-numbers-with-amazon-sns/
 *  for more information.
 *
 *  <phone_number_value> input parameter uses E.164 format.
 *  For example, in United States, this input value should be of the form:
 +12223334444
 */

//! Send an SMS text message to a phone number.
/*!
  \param message: The message to publish.
  \param phoneNumber: The phone number of the recipient in E.164 format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::publishSms(const Aws::String &message,
                             const Aws::String &phoneNumber,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetPhoneNumber(phoneNumber);

    const Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

    if (outcome.IsSuccess()) {
        std::cout << "Message published successfully with message id, '"
```

```
                  << outcome.GetResult().GetMessageId() << "'."
                  << std::endl;
    }
    else {
        std::cerr << "Error while publishing message "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 Publish를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
```

```
        final String usage = """

                Usage:    <message> <phoneNumber>

                Where:
                   message - The message text to send.
                   phoneNumber - The mobile phone number to which a message is
 sent (for example, +1XXX5550100).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }

    public static void pubTextSMS(SnsClient snsClient, String message, String
 phoneNumber) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .phoneNumber(phoneNumber)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Publish를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun pubTextSMS(messageVal: String?, phoneNumberVal: String?) {

    val request = PublishRequest {
        message = messageVal
        phoneNumber = phoneNumberVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 Publish를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Sends a text message (SMS message) directly to a phone number using Amazon
 SNS.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$phone = '+1XXX5550100';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'PhoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 자세한 정보는 AWS SDK for PHP 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for PHP API 참조의 Publish를 참조하십시오.

Python

SDK for Python(Boto3)

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def publish_text_message(self, phone_number, message):
        """
        Publishes a text message directly to a phone number without need for a
        subscription.

        :param phone_number: The phone number that receives the message. This
 must be
                             in E.164 format. For example, a United States phone
                             number might be +12065550101.
        :param message: The message to send.
        :return: The ID of the message.
        """
        try:
            response = self.sns_resource.meta.client.publish(
                PhoneNumber=phone_number, Message=message
            )
            message_id = response["MessageId"]
            logger.info("Published message to %s.", phone_number)
        except ClientError:
            logger.exception("Couldn't publish message to %s.", phone_number)
            raise
        else:
```

```
            return message_id
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 Publish를 참조하십시오.

SMS 활동 모니터링

SMS 활동을 모니터링하여 대상 전화번호, 전송 성공 또는 실패, 실패 이유, 비용 및 기타 정보를 추적할 수 있습니다. Amazon SNS는 콘솔에서 통계를 요약하고, Amazon CloudWatch에 정보를 보내고, 지정한 Amazon S3 버킷에 일일 SMS 사용 보고서를 전송하여 도움을 줍니다.

주제

- SMS 전송 통계 보기
- Amazon CloudWatch 지표 및 SMS 전송 로그 보기
- 일일 SMS 사용 보고서 보기

SMS 전송 통계 보기

Amazon SNS 콘솔을 사용하여 초신 SMS 전송에 대한 통계를 볼 수 있습니다.

1. Amazon SNS 콘솔에 로그인합니다.
2. 콘솔 메뉴에서 리전 선택기를 SMS 메시징을 지원하는 리전으로 설정합니다.
3. 탐색 창에서 문자 메시지(SMS)를 선택합니다.
4. [Text messaging (SMS)] 페이지의 [Account stats] 섹션에서 트랜잭션 및 프로모션 SMS 메시지 전송에 대한 차트를 봅니다. 각 차트에는 이전 15일에 대한 다음 데이터가 표시됩니다.
   - 전송 비율(전송 성공 비율)
   - 보냄(전송 시도 수)
   - 실패(전송 실패 수)

이 페이지에서 사용량 버튼을 선택하여 일일 사용 보고서를 저장하는 Amazon S3 버킷으로 이동할 수도 있습니다. 자세한 내용은 섹션을 참조하세요일일 SMS 사용 보고서 보기

Amazon CloudWatch 지표 및 SMS 전송 로그 보기

Amazon CloudWatch 및 Amazon CloudWatch Logs를 사용하여 SMS 메시지 전송을 모니터링할 수 있습니다.

주제

Amazon CloudWatch 지표 보기

Amazon SNS는 SMS 메시지 전송에 대한 지표를 자동으로 수집하고 Amazon CloudWatch에 푸시합니다. CloudWatch를 사용하여 이러한 측정치를 모니터링하고 지표가 임계값을 초과할 때 알리는 경보를 생성할 수 있습니다. 예를 들어 CloudWatch 지표를 모니터링하여 사용자의 SMS 전송 속도와 당월 누적 SMS 요금을 알아볼 수 있습니다.

CloudWatch 지표 모니터링, CloudWatch 경보 설정, 사용 가능한 지표 유형에 대한 자세한 내용은 CloudWatch를 사용하여 Amazon SNS 주제 모니터링에서 확인하세요.

CloudWatch Logs 보기

Amazon SNS가 Amazon CloudWatch Logs에 쓸 수 있도록 설정하여 SMS 메시지 전송 성공 및 실패에 대한 정보를 수집할 수 있습니다. 전송하는 각 SMS 메시지에 대해 Amazon SNS는 메시지 가격, 성공 또는 실패 상태, 실패 이유(메시지가 실패한 경우), 메시지 유지 시간 및 기타 정보가 포함된 로그를 작성합니다.

SMS 메시지에 대한 CloudWatch Logs를 사용하고 보려면

1. Amazon SNS 콘솔에 로그인합니다.
2. 콘솔 메뉴에서 리전 선택기를 SMS 메시징을 지원하는 리전으로 설정합니다.
3. 탐색 창에서 문자 메시지(SMS)를 선택합니다.
4. 모바일 문자 메시지(SMS) 페이지의 문자 메시지 기본 설정 섹션에서 편집을 선택합니다.
5. 다음 페이지에서 Delivery status logging(전송 상태 로깅) 섹션을 확장합니다.

6. 성공 샘플 비율에서 Amazon SNS가 CloudWatch Logs에 로그를 작성할 SMS 전송 성공 비율을 지정합니다. 예:

   - 전송 실패에 대한 로그만 작성하려면 이 값을 0으로 설정합니다.
   - 전송 성공의 10%에 대한 로그를 작성하려면 이 값을 10으로 설정합니다.

   비율을 지정하지 않으면 Amazon SNS는 모든 전송 성공에 대한 로그를 작성합니다.
7. 필요한 권한을 제공하려면 다음 중 하나를 수행하세요.

   - 새 서비스 역할을 만들려면 새 서비스 역할 생성을 선택한 다음 새 역할 생성을 선택합니다. 다음 페이지에서 허용을 선택하여 Amazon SNS에 계정 리소스에 대한 쓰기 액세스 권한을 부여합니다.
   - 기존 서비스 역할을 사용하려면 기존 서비스 역할 사용을 선택한 다음 성공 및 실패한 전송에 대한 IAM 역할 상자에 ARN 이름을 붙여넣습니다.

     지정하는 서비스 역할은 계정 리소스에 대한 쓰기 액세스를 허용해야 합니다. IAM 역할 생성에 대한 자세한 정보는 IAM 사용 설명서의 AWS 서비스에 대한 역할 생성을 참조하세요.
8. [Save changes]를 선택합니다.
9. 모바일 문자 메시징(SMS) 페이지로 돌아가서 전송 상태 로그 섹션으로 이동하여 사용 가능한 로그를 확인합니다.

> **Note**
>
> 대상 전화번호의 통신 사업자에 따라 Amazon SNS 콘솔에 전송 로그가 표시되는 데 최대 72시간이 걸릴 수 있습니다.

SMS 전송 성공에 대한 로그 예제

SMS 전송 성공에 대한 전송 상태 로그는 다음 예제와 비슷합니다.

```
{
    "notification": {
        "messageId": "34d9b400-c6dd-5444-820d-fbeb0f1f54cf",
        "timestamp": "2016-06-28 00:40:34.558"
    },
    "delivery": {
        "phoneCarrier": "My Phone Carrier",
```

```
        "mnc": 270,
        "numberOfMessageParts": 1,
        "destination": "+1XXX5550100",
        "priceInUSD": 0.00645,
        "smsType": "Transactional",
        "mcc": 310,
        "providerResponse": "Message has been accepted by phone carrier",
        "dwellTimeMs": 599,
        "dwellTimeMsUntilDeviceAck": 1344
    },
    "status": "SUCCESS"
}
```

SMS 전송 실패에 대한 로그 예제

SMS 전송 실패에 대한 전송 상태 로그는 다음 예제와 비슷합니다.

```
{
    "notification": {
        "messageId": "1077257a-92f3-5ca3-bc97-6a915b310625",
        "timestamp": "2016-06-28 00:40:34.559"
    },
    "delivery": {
        "mnc": 0,
        "numberOfMessageParts": 1,
        "destination": "+1XXX5550100",
        "priceInUSD": 0.00645,
        "smsType": "Transactional",
        "mcc": 0,
        "providerResponse": "Unknown error attempting to reach phone",
        "dwellTimeMs": 1420,
        "dwellTimeMsUntilDeviceAck": 1692
    },
    "status": "FAILURE"
}
```

SMS 전송 실패 이유

실패 이유는 `providerResponse` 속성을 통해 제공됩니다. SMS 메시지는 다음과 같은 이유로 전송에 실패할 수 있습니다.

- 전화 통신사가 스팸으로 차단함
- 대상이 차단된 목록에 있음

- 잘못된 전화번호
- 메시지 본문이 잘못됨
- 전화 통신사가 이 메시지를 차단함
- 전화 통신사가 현재 연결 불가능/이용 불가능함
- 전화에서 SMS가 차단됨
- 휴대폰이 차단된 목록에 있음
- 전화가 현재 연결 불가능/이용 불가능함
- 전화번호가 옵트아웃됨
- 이 메시지를 전송하면 최고 가격이 초과됨
- 전화 연결을 시도하는 중 알 수 없는 오류 발생

일일 SMS 사용 보고서 보기

Amazon SNS의 일일 사용 보고서를 구독하여 SMS 전송을 모니터링할 수 있습니다. 매일 하나 이상의 SMS 메시지를 전송하는 사용자의 경우, Amazon SNS는 사용 보고서를 지정된 Amazon S3 버킷에 CSV 파일로 전송합니다. SMS 사용 보고서가 S3 버킷에서 사용 가능하게 되려면 24시간이 걸립니다.

주제

일일 사용 보고서 정보

사용 보고서에는 계정에서 전송한 각 SMS 메시지에 대해 다음과 같은 정보가 포함됩니다.

보고서에는 옵트아웃한 수신자에게 전송된 메시지는 포함되지 않습니다.

- 메시지의 게시 시간(UTC)
- 메시지 ID
- 대상 전화번호
- 메시지 유형
- 전송 상태
- 메시지 가격(USD)

- 부분 번호(너무 길어서 단일 메시지로 전송할 수 없는 경우 메시지가 여러 부분으로 분할됨)
- 총 부분 수

> **Note**
>
> Amazon SNS에서 부품 번호를 받지 못한 경우 값을 0으로 설정합니다.

일일 사용 보고서 구독

일일 사용 보고서를 구독하려면 적절한 권한으로 Amazon S3 버킷을 생성해야 합니다.

일일 사용 보고서에 대한 Amazon S3 버킷을 만들려면

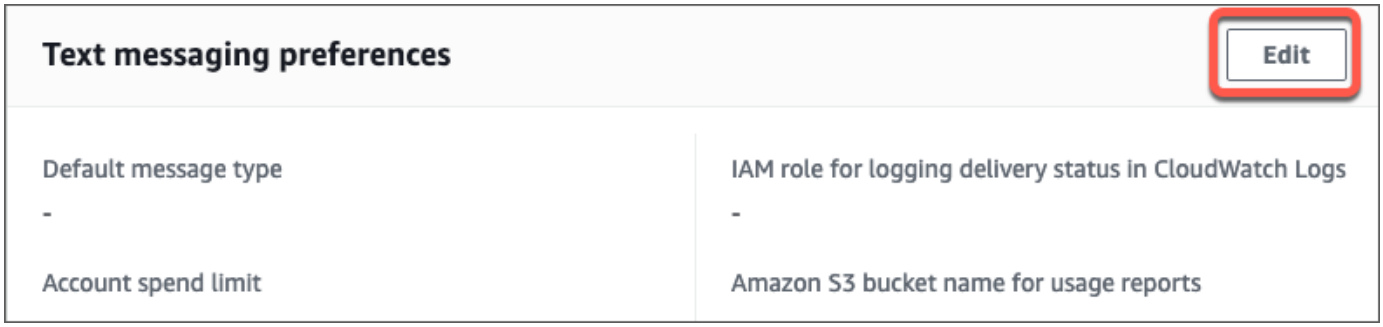
1. SMS 메시지를 보내는 AWS 계정에서 Amazon S3 콘솔에 로그인합니다.
2. 버킷 만들기를 선택합니다.
3. 버킷 이름으로는 계정과 조직에 고유한 이름을 입력하는 것이 좋습니다. 예를 들면, `<my-bucket-prefix>-<account_id>-<org-id>`. 패턴을 사용하세요.

   버킷 이름의 규칙 및 제한 사항에 대한 자세한 내용은 Amazon Simple Storage Service 사용 설명서의 버킷 명명 규칙을 참조하세요.
4. Create를 선택합니다.
5. 모든 버킷 테이블에서 버킷을 선택합니다.
6. 권한 섹션에서 버킷 정책을 선택합니다.
7. 버킷 정책 편집기 창에서 Amazon SNS 서비스 보안 주체가 버킷에 작성할 수 있도록 허용하는 정책을 제공합니다. 문제 해결 예는 버킷 정책 예제을(를) 참조하십시오.

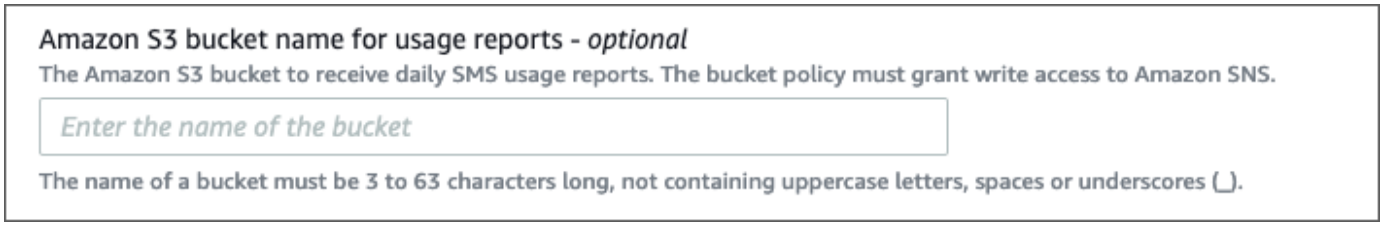
   정책 예제를 사용하는 경우 *my-s3-bucket*을 3단계에서 선택한 버킷 이름으로 대체해야 합니다.
8. 저장(Save)을 선택합니다.

일일 사용 보고서를 구독하려면

1. Amazon SNS 콘솔에 로그인합니다.
2. 탐색 창에서 문자 메시지(SMS)를 선택합니다.
3. 문자 메시지(SMS) 페이지의 문자 메시지 기본 설정 섹션에서 편집을 선택합니다.



4. 문자 메시지 기본 설정 편집 페이지의 세부 정보 섹션에서 Amazon S3 bucket name for usage reports를 지정합니다.



5. [Save changes]를 선택합니다.

버킷 정책 예제

다음 정책은 Amazon SNS 서비스 보안 주체가 `s3:PutObject`, `s3:GetBucketLocation` 및 `s3:ListBucket` 작업을 수행하도록 허용합니다.

AWS에서는 계정의 리소스에 대한 액세스 권한이 부여된 서비스 보안 주체를 사용하여 모든 서비스에 대한 도구를 제공합니다. Amazon S3 버킷 정책문의 보안 주체가 AWS 서비스 보안 주체인 경우 `aws:SourceArn` 또는 `aws:SourceAccount` 전역 조건 키를 사용하여 혼동된 대리자 문제를 방지할 수 있습니다. 버킷이 일일 사용 보고서를 수신할 수 있는 리전 및 계정을 제한하려면 아래 예와 같이 `aws:SourceArn`을 사용합니다. 이러한 보고서를 생성할 수 있는 리전을 제한하지 않으려면 `aws:SourceAccount`를 사용하여 보고서를 생성하는 계정을 기반으로 제한하십시오. 리소스의 ARN을 모르는 경우 `aws:SourceAccount`를 사용합니다.

Amazon SNS로부터 일일 SMS 사용 보고서를 수신하기 위해 Amazon S3 버킷을 생성할 때 다음 예와 같이 혼동된 대리자 방지를 포함합니다.

```
{
 "Version": "2008-10-17",
 "Statement": [{
   "Sid": "AllowPutObject",
   "Effect": "Allow",
   "Principal": {
```

```
    "Service": "sns.amazonaws.com"
   },
   "Action": "s3:PutObject",
   "Resource": "arn:aws:s3:::my-s3-bucket/*",
   "Condition": {
    "StringEquals": {
     "aws:SourceAccount": "account_id"
    },
    "ArnLike": {
     "aws:SourceArn": "arn:aws:sns:region:account_id:*"
    }
   }
  },
  {
   "Sid": "AllowGetBucketLocation",
   "Effect": "Allow",
   "Principal": {
    "Service": "sns.amazonaws.com"
   },
   "Action": "s3:GetBucketLocation",
   "Resource": "arn:aws:s3:::my-s3-bucket",
   "Condition": {
    "StringEquals": {
     "aws:SourceAccount": "account_id"
    },
    "ArnLike": {
     "aws:SourceArn": "arn:aws:sns:region:account_id:*"
    }
   }
  },
  {
   "Sid": "AllowListBucket",
   "Effect": "Allow",
   "Principal": {
    "Service": "sns.amazonaws.com"
   },
   "Action": "s3:ListBucket",
   "Resource": "arn:aws:s3:::my-s3-bucket",
   "Condition": {
    "StringEquals": {
     "aws:SourceAccount": "account_id"
    },
    "ArnLike": {
     "aws:SourceArn": "arn:aws:sns:region:account_id:*"
```

```
    }
   }
  }
 ]
}
```

> **Note**
>
> Amazon S3 정책의 Condition 요소에 지정된 AWS 계정이 소유한 Amazon S3 버킷에 사용 보고서를 게시할 수 있습니다. 다른 AWS 계정이 소유한 Amazon S3 버킷에 사용 보고서를 게시하려면 다른 AWS 계정에서 S3 객체를 복사하려면 어떻게 해야 하나요?를 참조하세요.

일일 사용 보고서 예제

매일 일일 사용 보고서를 구독한 후 Amazon SNS는 사용 데이터가 포함된 CSV 파일을 다음 위치에 저장합니다.

```
<my-s3-bucket>/SMSUsageReports/<region>/YYYY/MM/DD/00x.csv.gz
```

각 파일에는 최대 50,000개의 레코드가 포함될 수 있습니다. 1일에 대한 레코드가 이 할당량을 초과하는 경우 Amazon SNS는 여러 파일을 추가합니다.

다음 그림에서는 보고서 예제를 보여 줍니다.

```
PublishTimeUTC,MessageId,DestinationPhoneNumber,MessageType,DeliveryStatus,PriceInUSD,PartNumbe
2016-05-10T03:00:29.476Z,96a298ac-1458-4825-
a7eb-7330e0720b72,1XXX5550100,Promotional,Message has been accepted by phone
 carrier,0.90084,0,1
2016-05-10T03:00:29.561Z,1e29d394-
d7f4-4dc9-996e-26412032c344,1XXX5550100,Promotional,Message has been accepted by phone
 carrier,0.34322,0,1
2016-05-10T03:00:30.769Z,98ba941c-afc7-4c51-
ba2c-56c6570a6c08,1XXX5550100,Transactional,Message has been accepted by phone
 carrier,0.27815,0,1
```

전화번호 및 SMS 구독 관리

Amazon SNS는 계정의 SMS 메시지를 누가 수신하는지를 관리하기 위한 여러 옵션을 제공합니다. 제한된 빈도수에 한하여, 계정의 SMS 메시지 수신을 옵트아웃한 전화번호를 옵트인할 수 있습니다. SMS 구독에 메시지 전송을 중지하려면 구독 또는 구독에 게시하는 주제를 제거할 수 있습니다.

주제

SMS 메시지 수신 옵트아웃

현지 법률 및 규정에서 요구하는 경우(예: 미국 및 캐나다), SMS 수신자는 다음 서비스와 함께 메시지에 회신하는 방식으로 디바이스를 사용하여 옵트아웃할 수 있습니다.

- ARRET(프랑스어)
- CANCEL
- END
- OPT-OUT
- OPTOUT
- QUIT
- REMOVE
- STOP
- TD
- UNSUBSCRIBE

옵트아웃하려면 수신자가 Amazon SNS가 메시지를 전송하는 데 사용한 것과 동일한 발신 번호로 회신해야 합니다. 수신을 거부하면 전화번호를 AWS 계정 옵트인하지 않는 한 수신자가 더 이상 수신자가 전송한 SMS 메시지를 받지 않습니다.

전화번호에서 Amazon SNS 주제를 구독하는 경우 옵트아웃하더라도 구독이 제거되지 않지만, 전화번호를 옵트인하지 않는 한 SMS 메시지는 해당 구독으로 전송되지 않습니다.

전화번호 및 구독 관리(콘솔)

Amazon SNS 콘솔을 사용하여 어떤 전화번호가 계정의 SMS 메시지를 수신하는지를 제어할 수 있습니다.

옵트아웃한 전화번호 옵트인

어떤 전화번호가 계정의 SMS 메시지 수신을 옵트아웃했는지 볼 수 있으며, 이러한 전화번호를 옵트인하여 해당 전화번호로 메시지를 전송을 다시 시작할 수 있습니다.

30일에 한 번만 전화번호를 옵트인할 수 있습니다.

1. Amazon SNS 콘솔에 로그인합니다.
2. 콘솔 메뉴에서 리전 선택기를 SMS 메시징을 지원하는 리전으로 설정합니다.
3. 탐색 창에서 문자 메시지(SMS)를 선택합니다.
4. [Text messaging (SMS)] 페이지에서 [View opted out phone numbers]를 선택합니다. [Opted out phone numbers] 페이지에 옵트아웃된 전화번호가 표시됩니다.
5. 옵트인하려는 전화번호의 확인란을 선택하고 [Opt in]을 선택합니다. 전화번호가 더 이상 옵트아웃되지 않으며 전송되는 SMS 메시지를 수신합니다.

SMS 구독 삭제

SMS 구독을 삭제하여 주제에 게시할 때 해당 전화번호로 SMS 메시지를 전송하는 것을 중지합니다.

1. 탐색 창에서 구독을 선택합니다.
2. 삭제하려는 구독의 확인란을 선택합니다. 그런 다음 [Actions]를 선택하고 [Delete Subscriptions]를 선택합니다.
3. 삭제 창에서 삭제를 선택합니다. Amazon SNS는 구독을 삭제하고 성공 메시지를 표시합니다.

주제 삭제

메시지를 구독 엔드포인트에 더 이상 게시하지 않으려는 경우 주제를 삭제합니다.

1. 탐색 창에서 주제(Topics)를 선택합니다.
2. 삭제하려는 주제의 확인란을 선택합니다. 그런 다음 [Actions]를 선택하고 [Delete Topics]를 선택합니다.
3. 삭제 창에서 삭제를 선택합니다. Amazon SNS는 주제를 삭제하고 성공 메시지를 표시합니다.

전화번호 및 구독 관리(AWS SDK)

AWS SDK를 사용하여 Amazon SNS에 프로그래밍 방식으로 요청하고 계정에서 SMS 메시지를 수신할 수 있는 전화번호를 관리할 수 있습니다.

AWS SDK를 사용하려면 자격 증명으로 구성해야 합니다. 자세한 정보는 AWS SDK 및 도구 참조 설명서의 공유 구성 및 자격 증명 파일을 참조하세요.

옵트아웃한 모든 전화번호 보기

옵트아웃한 모든 전화번호를 보려면 Amazon SNS API를 통해 `ListPhoneNumbersOptedOut` 요청을 제출합니다.

다음 코드 예제는 `ListPhoneNumbersOptedOut`의 사용 방법을 보여 줍니다.

CLI

AWS CLI

SMS 메시지 옵트아웃을 나열하려면

다음 `list-phone-numbers-opted-out` 예제에서는 SMS 메시지 수신을 옵트아웃한 전화번호를 나열합니다.

```
aws sns list-phone-numbers-opted-out
```

출력:

```
{
    "phoneNumbers": [
        "+15555550100"
    ]
}
```

- API 세부 정보는 AWS CLI 명령 ListPhoneNumbersOptedOut참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutRequest;
import
 software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListOptOut {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listOpts(snsClient);
        snsClient.close();
    }

    public static void listOpts(SnsClient snsClient) {
        try {
            ListPhoneNumbersOptedOutRequest request =
 ListPhoneNumbersOptedOutRequest.builder().build();
            ListPhoneNumbersOptedOutResponse result =
 snsClient.listPhoneNumbersOptedOut(request);
            System.out.println("Status is " +
 result.sdkHttpResponse().statusCode() + "\n\nPhone Numbers: \n\n"
                    + result.phoneNumbers());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API ListPhoneNumbersOptedOut참조를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Returns a list of phone numbers that are opted out of receiving SMS messages
 from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listPhoneNumbersOptedOut();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 자세한 정보는 AWS SDK for PHP 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for PHP API ListPhoneNumbersOptedOut참조를 참조하십시오.

전화번호가 옵트아웃되었는지 여부 확인

전화번호가 옵트아웃되었는지 여부를 확인하려면 Amazon SNS API를 통해 `CheckIfPhoneNumberIsOptedOut` 요청을 제출합니다.

다음 코드 예제는 `CheckIfPhoneNumberIsOptedOut`의 사용 방법을 보여 줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example shows how to use the Amazon Simple Notification Service
    /// (Amazon SNS) to check whether a phone number has been opted out.
    /// </summary>
    public class IsPhoneNumOptedOut
    {
        public static async Task Main()
        {
            string phoneNumber = "+15551112222";

            IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();
```

```
            await CheckIfOptedOutAsync(client, phoneNumber);
        }

        /// <summary>
        /// Checks to see if the supplied phone number has been opted out.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS Client object used
        /// to check if the phone number has been opted out.</param>
        /// <param name="phoneNumber">A string representing the phone number
        /// to check.</param>
        public static async Task
CheckIfOptedOutAsync(IAmazonSimpleNotificationService client, string
phoneNumber)
        {
            var request = new CheckIfPhoneNumberIsOptedOutRequest
            {
                PhoneNumber = phoneNumber,
            };

            try
            {
                var response = await
client.CheckIfPhoneNumberIsOptedOutAsync(request);

                if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
                {
                    string optOutStatus = response.IsOptedOut ? "opted out" :
"not opted out.";
                    Console.WriteLine($"The phone number: {phoneNumber} is
{optOutStatus}");
                }
            }
            catch (AuthorizationErrorException ex)
            {
                Console.WriteLine($"{ex.Message}");
            }
        }
    }
```

- API 세부 정보는 AWS SDK for .NET API CheckIfPhoneNumberIsOptedOut참조를 참조하십시오.

CLI

AWS CLI

전화번호의 SMS 메시지 옵트아웃을 확인하려면

다음 `check-if-phone-number-is-opted-out` 예시는 지정된 전화번호가 현재 AWS 계정으로부터 SMS 메시지 수신을 거부했는지 여부를 확인합니다.

```
aws sns check-if-phone-number-is-opted-out \
    --phone-number +1555550100
```

출력:

```
{
    "isOptedOut": false
}
```

- API 세부 정보는 AWS CLI 명령 CheckIfPhoneNumberIsOptedOut참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
 software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutRequest;
import
 software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CheckOptOut {
    public static void main(String[] args) {

        final String usage = """

                Usage:    <phoneNumber>

                Where:
                   phoneNumber - The mobile phone number to look up (for example,
 +1XXX5550100).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String phoneNumber = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        checkPhone(snsClient, phoneNumber);
        snsClient.close();
    }

    public static void checkPhone(SnsClient snsClient, String phoneNumber) {
        try {
            CheckIfPhoneNumberIsOptedOutRequest request =
 CheckIfPhoneNumberIsOptedOutRequest.builder()
                    .phoneNumber(phoneNumber)
                    .build();

            CheckIfPhoneNumberIsOptedOutResponse result =
 snsClient.checkIfPhoneNumberIsOptedOut(request);
            System.out.println(
```

```
                    result.isOptedOut() + "Phone Number " + phoneNumber + " has
 Opted Out of receiving sns messages." +
                            "\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API CheckIfPhoneNumberIsOptedOut참조를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

별도의 모듈에서 클라이언트를 생성하고 내보냅니다.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
import { CheckIfPhoneNumberIsOptedOutCommand } from "@aws-sdk/client-sns";

import { snsClient } from "../libs/snsClient.js";
```

```
export const checkIfPhoneNumberIsOptedOut = async (
  phoneNumber = "5555555555",
) => {
  const command = new CheckIfPhoneNumberIsOptedOutCommand({
    phoneNumber,
  });

  const response = await snsClient.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '3341c28a-cdc8-5b39-a3ee-9fb0ee125732',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   isOptedOut: false
  // }
  return response;
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API CheckIfPhoneNumberIsOptedOut참조를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'vendor/autoload.php';
```

```
use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Indicates whether the phone number owner has opted out of receiving SMS
 messages from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$phone = '+1XXX5550100';

try {
    $result = $SnSclient->checkIfPhoneNumberIsOptedOut([
        'phoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 자세한 정보는 AWS SDK for PHP 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for PHP API CheckIfPhoneNumberIsOptedOut참조를 참조하십시오.

옵트아웃한 전화번호 옵트인

전화번호를 옵트인하려면 Amazon SNS API를 통해 `OptInPhoneNumber` 요청을 제출합니다.

30일에 한 번만 전화번호를 옵트인할 수 있습니다.

SMS 구독 삭제

Amazon SNS 주제에서 SMS 구독을 삭제하려면 Amazon SNS API를 통해 `ListSubscriptions` 요청을 제출하여 구독 ARN을 가져온 다음 ARN을 `Unsubscribe` 요청에 전달합니다.

다음 코드 예제는 `Unsubscribe`의 사용 방법을 보여 줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

구독 ARN으로 주제 구독을 취소합니다.

```
/// <summary>
/// Unsubscribe from a topic by a subscription ARN.
/// </summary>
/// <param name="subscriptionArn">The ARN of the subscription.</param>
/// <returns>True if successful.</returns>
public async Task<bool> UnsubscribeByArn(string subscriptionArn)
{
    var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
        new UnsubscribeRequest()
        {
            SubscriptionArn = subscriptionArn
        });
    return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 Unsubscribe를 참조하십시오.

C++

SDK for C++

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
//! Delete a subscription to an Amazon Simple Notification Service (Amazon SNS)
 topic.
/*!
  \param subscriptionARN: The Amazon Resource Name (ARN) for an Amazon SNS topic
 subscription.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::unsubscribe(const Aws::String &subscriptionARN,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::UnsubscribeRequest request;
    request.SetSubscriptionArn(subscriptionARN);

    const Aws::SNS::Model::UnsubscribeOutcome outcome =
 snsClient.Unsubscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Unsubscribed successfully " << std::endl;
    }
    else {
        std::cerr << "Error while unsubscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 Unsubscribe를 참조하십시오.

CLI

AWS CLI

주제 구독을 취소하려면

다음 unsubscribe 예제에서는 주제에서 지정된 구독을 삭제합니다.

```
aws sns unsubscribe \
    --subscription-arn arn:aws:sns:us-west-2:0123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f
```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 참조의 Unsubscribe를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class Unsubscribe {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionArn>

                Where:
                   subscriptionArn - The ARN of the subscription to delete.
                """;

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        unSub(snsClient, subscriptionArn);
        snsClient.close();
    }

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                    .subscriptionArn(subscriptionArn)
                    .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode()
                    + "\n\nSubscription was removed for " +
 request.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

```
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Unsubscribe를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

별도의 모듈에서 클라이언트를 생성하고 내보냅니다.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
import { UnsubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} subscriptionArn - The ARN of the subscription to cancel.
 */
const unsubscribe = async (
  subscriptionArn = "arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic:xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxxx",
) => {
  const response = await snsClient.send(
    new UnsubscribeCommand({
      SubscriptionArn: subscriptionArn,
    }),
```

```
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '0178259a-9204-507c-b620-78a7570a44c6',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 Unsubscribe를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun unSub(subscriptionArnVal: String) {

    val request = UnsubscribeRequest {
        subscriptionArn = subscriptionArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for ${request.subscriptionArn}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 Unsubscribe를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Deletes a subscription to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription = 'arn:aws:sns:us-east-1:111122223333:MySubscription';

try {
    $result = $SnSclient->unsubscribe([
        'SubscriptionArn' => $subscription,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
```

```
}
```

- 자세한 정보는 AWS SDK for PHP 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for PHP API 참조의 Unsubscribe를 참조하십시오.

Python

SDK for Python(Boto3)

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def delete_subscription(subscription):
        """
        Unsubscribes and deletes a subscription.
        """
        try:
            subscription.delete()
            logger.info("Deleted subscription %s.", subscription.arn)
        except ClientError:
            logger.exception("Couldn't delete subscription %s.",
 subscription.arn)
            raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 Unsubscribe를 참조하십시오.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        lo_sns->unsubscribe( iv_subscriptionarn = iv_subscription_arn ).
        MESSAGE 'Subscription deleted.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Subscription does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snsinvalidparameterex.
        MESSAGE 'Subscription with "PendingConfirmation" status cannot be
deleted/unsubscribed. Confirm subscription before performing unsubscribe
operation.' TYPE 'E'.
    ENDTRY.
```

- API에 대한 세부 정보는 SAP ABAP용AWS SDK API 참조의 Unsubscribe를 참조하세요.

주제 삭제

주제 및 해당 주제의 모든 구독을 삭제하려면 Amazon SNS API를 통해 `ListTopics` 요청을 제출하여 주제 ARN을 가져온 다음 ARN을 `DeleteTopic` 요청에 전달합니다.

다음 코드 예제는 `DeleteTopic`의 사용 방법을 보여 줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

주제 ARN으로 주제를 삭제합니다.

```
/// <summary>
/// Delete a topic by its topic ARN.
/// </summary>
/// <param name="topicArn">The ARN of the topic.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteTopicByArn(string topicArn)
{
    var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
        new DeleteTopicRequest()
        {
            TopicArn = topicArn
        });
    return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- API 세부 정보는 AWS SDK for .NET API DeleteTopic참조를 참조하십시오.

C++

SDK for C++

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
//! Delete an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::deleteTopic(const Aws::String &topicARN,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::DeleteTopicRequest request;
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::DeleteTopicOutcome outcome =
 snsClient.DeleteTopic(request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted the Amazon SNS topic " << topicARN <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting topic " << topicARN << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API DeleteTopic참조를 참조하십시오.

CLI

AWS CLI

SNS 주제를 삭제하려면

다음 `delete-topic` 예제에서는 지정된 SNS 주제를 삭제합니다.

```
aws sns delete-topic \
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic"
```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 DeleteTopic참조를 참조하십시오.

Go

SDK for Go V2

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// 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(topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(context.TODO(), &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 DeleteTopic참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:     <topicArn>

                Where:
                   topicArn - The ARN of the topic to delete.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
```

```
                .region(Region.US_EAST_1)
                .build();

        System.out.println("Deleting a topic with name: " + topicArn);
        deleteSNSTopic(snsClient, topicArn);
        snsClient.close();
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                    .topicArn(topicArn)
                    .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API DeleteTopic참조를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

별도의 모듈에서 클라이언트를 생성하고 내보냅니다.

```
import { SNSClient } from "@aws-sdk/client-sns";
```

```
// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
import { DeleteTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic to delete.
 */
export const deleteTopic = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new DeleteTopicCommand({ TopicArn: topicArn }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'a10e2886-5a8f-5114-af36-75bd39498332',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API DeleteTopic참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteSNSTopic(topicArnVal: String) {

    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was successfully deleted.")
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 DeleteTopic.

PHP

SDK for PHP

> **Note**
>
> 자세한 내용은 여기에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

```
/**
 * Deletes an SNS topic and all its subscriptions.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->deleteTopic([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- API 세부 정보는 AWS SDK for PHP API DeleteTopic참조를 참조하십시오.

Python

SDK for Python(Boto3)

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class SnsWrapper:
```

```
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def delete_topic(topic):
        """
        Deletes a topic. All subscriptions to the topic are also deleted.
        """
        try:
            topic.delete()
            logger.info("Deleted topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't delete topic %s.", topic.arn)
            raise
```

- API에 대한 자세한 내용은 파이썬용AWS SDK (Boto3) API 레퍼런스를 참조하십시오 DeleteTopic.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 자세한 내용은 여기에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        lo_sns->deletetopic( iv_topicarn = iv_topic_arn ).
        MESSAGE 'SNS topic deleted.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
```

```
    ENDTRY.
```

- API 세부 정보는 SAP용AWS SDK ABAP API 참조를 참조하십시오 DeleteTopic.

지원되는 국가 및 리전

> **Important**
>
> 2023년 8월 31일부터 미국 및 미국 영토(괌, 푸에르토리코, 미국령 사모아 제도, US 버진 아일랜드)로 SMS 메시지를 보내려면 10DLC 번호 또는 수신자 부담 전화번호와 같은 전용 번호가 필요합니다.

현재 Amazon SNS는 다음 AWS 지역에서 SMS 메시징을 지원합니다.

지역명	지역	엔드포인트	프로토콜
미국 동부(오하이오)	us-east-2	sns.us-east-2.amazonaws.com	HTTP 및 HTTPS
미국 동부(버지니아 북부)	us-east-1	sns.us-east-1.amazonaws.com	HTTP 및 HTTPS
미국 서부(캘리포니아 북부)	us-west-1	sns.us-west-1.amazonaws.com	HTTP 및 HTTPS
미국 서부(오레곤)	us-west-2	sns.us-west-2.amazonaws.com	HTTP 및 HTTPS
아프리카(케이프타운)	af-south-1	sns.af-south-1.amazonaws.com	HTTP 및 HTTPS
아시아 태평양(하이데라바드)	ap-south-2	sns.ap-south-2.amazonaws.com	HTTP 및 HTTPS
아시아 태평양(자카르타)	ap-southeast-3	sns.ap-southeast-3.amazonaws.com	HTTP 및 HTTPS

지역명	지역	엔드포인트	프로토콜
아시아 태평양(멜버른)	ap-southeast-4	sns.ap-southeast-4.amazonaws.com	HTTP 및 HTTPS
아시아 태평양(뭄바이)	ap-south-1	sns.ap-south-1.amazonaws.com	HTTP 및 HTTPS
아시아 태평양(오사카)	ap-northeast-3	sns.ap-northeast-3.amazonaws.com	HTTP 및 HTTPS
아시아 태평양(싱가포르)	ap-southeast-1	sns.ap-southeast-1.amazonaws.com	HTTP 및 HTTPS
아시아 태평양(시드니)	ap-southeast-2	sns.ap-southeast-2.amazonaws.com	HTTP 및 HTTPS
아시아 태평양(도쿄)	ap-northeast-1	sns.ap-northeast-1.amazonaws.com	HTTP 및 HTTPS
캐나다(중부)	ca-central-1	sns.ca-central-1.amazonaws.com	HTTP 및 HTTPS
유럽(프랑크푸르트)	eu-central-1	sns.eu-central-1.amazonaws.com	HTTP 및 HTTPS
유럽(아일랜드)	eu-west-1	sns.eu-west-1.amazonaws.com	HTTP 및 HTTPS
유럽(런던)	eu-west-2	sns.eu-west-2.amazonaws.com	HTTP 및 HTTPS
유럽(밀라노)	eu-south-1	sns.eu-south-1.amazonaws.com	HTTP 및 HTTPS
유럽(파리)	eu-west-3	sns.eu-west-3.amazonaws.com	HTTP 및 HTTPS
유럽(스페인)	eu-south-2	sns.eu-south-2.amazonaws.com	HTTP 및 HTTPS

지역명	지역	엔드포인트	프로토콜
유럽(스톡홀름)	eu-north-1	sns.eu-north-1.amazonaws.com	HTTP 및 HTTPS
유럽(취리히)	eu-central-2	sns.eu-central-2.amazonaws.com	HTTP 및 HTTPS
이스라엘(텔아비브)	il-central-1	sns.il-central-1.amazonaws.com	HTTP 및 HTTPS
중동(바레인)	me-south-1	sns.me-south-1.amazonaws.com	HTTP 및 HTTPS
중동(UAE)	me-central-1	sns.me-central-1.amazonaws.com	HTTP 및 HTTPS
남아메리카(상파울루)	sa-east-1	sns.sa-east-1.amazonaws.com	HTTP 및 HTTPS
AWS GovCloud (미국 동부)	us-gov-east-1	sns. us-gov-east-1.amazonaws.com	HTTP 및 HTTPS
AWS GovCloud (미국 서부)	us-gov-west-1	sns. us-gov-west-1.amazonaws.com	HTTP 및 HTTPS

Amazon SNS를 사용하여 다음 국가 및 리전으로 SMS 메시지를 전송할 수 있습니다.

> **Note**
>
> 지원되는 국가에서 발신자 ID를 사용하면 SMS 전송을 개선할 수 있습니다.

국가 또는 리전	ISO 코드	다이얼링 코드	단축 코드 지원	긴 코드 지원	발신자 ID 지원	양방향 SMS 지원
A						

국가 또는 리전	ISO 코드	다이얼링 코드	단축 코드 지원	긴 코드 지원	발신자 ID 지원	양방향 SMS 지원
아프가니스탄	AF	93	아니요	아니요	예	아니요
알바니아	AL	355	아니요	아니요	예	아니요
알제리	DZ	213	아니요	아니요	예	아니요
안도라	AD	376	아니요	아니요	예	아니요
앙귈라	AI	1-264	아니요	아니요	예	아니요
앤티가 바부다	AG	1-268	아니요	아니요	예	아니요
아르헨티나	AR	54	예	아니요	아니요	아니요
아르메니아	AM	374	아니요	아니요	예	아니요
아루바 섬	AW	297	아니요	아니요	예	아니요
호주	AU	61	아니요	예	등록 필요[1]	예
오스트리아	AT	43	예	예	예	예
아제르바이잔	AZ	994	아니요	아니요	예	아니요
B						
바하마	BS	1-242	아니요	아니요	아니요	아니요
바레인	BH	973	아니요	아니요	예	아니요
방글라데시	BD	880	아니요	아니요	예	아니요
바베이도스	BB	1-246	아니요	아니요	예	아니요
벨로루시	BY	375	아니요	아니요	등록 필요[1]	아니요

국가 또는 리전	ISO 코드	다이얼링 코드	단축 코드 지원	긴 코드 지원	발신자 ID 지원	양방향 SMS 지원
벨기에	BE	32	아니요	예	아니요	예
벨리즈	BZ	501	아니요	아니요	예	아니요
버뮤다	BM	1-441	아니요	아니요	예	아니요
부탄	BT	975	아니요	아니요	예	아니요
볼리비아	BO	591	아니요	아니요	예	아니요
보스니아 헤르체코비나	BA	387	아니요	아니요	예	아니요
보츠와나	BW	267	아니요	아니요	예	아니요
브라질	BR	55	예	아니요	아니요	예
브루나이	BN	673	아니요	아니요	예	아니요
불가리아	BG	359	예	아니요	예	예
부르키나 파소	BF	226	아니요	아니요	예	아니요
부룬디	BL	257	아니요	아니요	예	아니요
C						
캄보디아	KH	855	아니요	아니요	예	아니요
카메룬	CM	237	아니요	아니요	예	아니요
캐나다	CA	1	예	예	아니요	예
카보베르데	CV	238	아니요	아니요	예	아니요

국가 또는 리전	ISO 코드	다이얼링 코드	단축 코드 지원	긴 코드 지원	발신자 ID 지원	양방향 SMS 지원
케이맨 제도	KY	1-345	아니요	아니요	아니요	아니요
중앙아프리카 공화국	CF	236	아니요	아니요	예	아니요
차드	TD	235	아니요	아니요	예	아니요
칠레	CL	56	예	예	아니요	예
중국	CN	86	예	아니요	아니요[2]	예
콜롬비아	CO	57	예	예	아니요	예
코모로	KM	269	아니요	아니요	예	아니요
쿡 제도	CK	682	아니요	아니요	예	예
코스타리카	CR	506	아니요	아니요	아니요	아니요
크로아티아	HR	385	아니요	아니요	예	아니요
사이프러스	CY	357	아니요	아니요	예	아니요
체코(체코 공화국)	CZ	420	아니요	예	예	예
D						
콩고민주공화국	CD	243	아니요	아니요	예	아니요
덴마크	DK	45	예	예	예	예
지부티	DJ	253	아니요	아니요	예	아니요
도미니카	DM	1-767	아니요	아니요	예	아니요

국가 또는 리전	ISO 코드	다이얼링 코드	단축 코드 지원	긴 코드 지원	발신자 ID 지원	양방향 SMS 지원
도미니카 공화국	DO	1-809, 1-829, 1-849	예	아니요	아니요	예
E						
에콰도르	EC	593	예	아니요	아니요	예
이집트	EG	20	예	아니요	등록 필요[1]	예
엘살바도르	SV	503	아니요	아니요	아니요	아니요
적도 기니	GQ	240	아니요	아니요	예	아니요
에리트레아	ER	291	아니요	아니요	예	아니요
에스토니아	EE	372	아니요	예	예	예
에티오피아	ET	251	아니요	아니요	예	아니요
F						
페로 제도	FO	298	아니요	아니요	예	아니요
피지	FJ	679	아니요	아니요	예	아니요
핀란드	FI	358	예	예	예	예
프랑스	FR	33	예	아니요	예	예
프랑스령 기아나	GF	594	아니요	아니요	예	아니요
프랑스령 폴리네시아	PF	689	아니요	아니요	예	아니요
G						

국가 또는 리전	ISO 코드	다이얼링 코드	단축 코드 지원	긴 코드 지원	발신자 ID 지원	양방향 SMS 지원
가봉	GA	241	아니요	아니요	예	아니요
감비아	GM	220	아니요	아니요	예	아니요
그루지야	GE	995	아니요	아니요	예	아니요
독일	DE	49	예	예	예	예
가나	GH	233	아니요	아니요	예	아니요
지브롤터	GI	350	아니요	아니요	예	아니요
그리스	GR	30	아니요	아니요	예	아니요
그린란드	GL	299	아니요	아니요	예	아니요
그레나다	GD	1-473	아니요	아니요	예	아니요
과들루프	GP	590	아니요	아니요	예	아니요
괌	GU	1-671	아니요	아니요	아니요	예
과테말라	GT	502	아니요	아니요	아니요	아니요
건지	GG	44-1481	아니요	아니요	예	아니요
기니	GN	224	아니요	아니요	예	아니요
기니비사우	GW	245	아니요	아니요	예	N/A
가이아나	GY	592	아니요	아니요	예	아니요
H						
아이티	H	509	아니요	아니요	예	아니요
온두라스	HN	504	아니요	아니요	예	아니요
홍콩	HK	852	아니요	예	예	예

국가 또는 리전	ISO 코드	다이얼링 코드	단축 코드 지원	긴 코드 지원	발신자 ID 지원	양방향 SMS 지원
헝가리	HU	36	아니요	예	아니요	예
I						
아이슬란드	IS	354	아니요	아니요	예	아니요
인도	IN	91	예	예[4]	등록 필요[3]	예
인도네시아	ID	62	아니요	아니요	예	아니요
이라크	IQ	964	아니요	아니요	예	아니요
아일랜드	IE	353	아니요	예	예	예
맨 섬	IM	44-1624	아니요	아니요	예	아니요
이스라엘	IL	972	아니요	예	예	예
이탈리아	IT	39	예	예	예	예
아이버리 코스트	CI	225	아니요	아니요	예	아니요
J						
자메이카	JM	1-876	아니요	아니요	예	아니요
일본	JP	81	예	예	예	예
저지	JE	44-1534	아니요	예	예	예
요르단	JO	962	아니요	아니요	등록 필요[1]	아니요
K						
카자흐스탄	KZ	7	아니요	아니요	예	아니요
케냐	KE	254	아니요	아니요	예	아니요

국가 또는 리전	ISO 코드	다이얼링 코드	단축 코드 지원	긴 코드 지원	발신자 ID 지원	양방향 SMS 지원
코소보	XK	383	아니요	아니요	예	아니요
쿠웨이트	KW	965	아니요	아니요	등록 필요[1]	아니요
키르기스스탄	KG	996	아니요	아니요	예	아니요
L						
라오스	LA	856	아니요	아니요	예	아니요
라트비아	LV	371	아니요	아니요	예	아니요
레바논	LB	961	아니요	아니요	예	아니요
레소토	LS	266	아니요	아니요	예	아니요
라이베리아	LR	231	아니요	예	아니요	
리비아	LY	218	아니요	아니요	예	아니요
리히텐슈타인	LI	423	아니요	아니요	예	아니요
리투아니아	LT	370	아니요	예	예	예
룩셈부르크	LU	352	아니요	예	예	예
M						
마카오	MO	853	아니요	아니요	예	아니요
마케도니아	MK	389	아니요	아니요	예	아니요
마다가스카르	MG	261	아니요	아니요	예	아니요
말라위	MW	265	아니요	아니요	예	아니요

국가 또는 리전	ISO 코드	다이얼링 코드	단축 코드 지원	긴 코드 지원	발신자 ID 지원	양방향 SMS 지원
말레이시아	MY	60	예	아니요	아니요	예
몰디브	MV	960	아니요	아니요	예	아니요
말리	ML	223	아니요	아니요	예	아니요
몰타	MT	356	아니요	아니요	예	아니요
마셜 제도	MH	692	아니요	아니요	아니요	아니요
마르티니크	MQ	596	아니요	아니요	예	아니요
모리타니아	MR	222	아니요	아니요	예	아니요
모리셔스	MU	230	아니요	아니요	예	아니요
마요트	YT	262	아니요	아니요	예	아니요
멕시코	MX	52	예	아니요	아니요	예
미크로네시아(연방공화국)	FM	691	아니요	아니요	아니요	아니요
몰도바	MD	373	아니요	아니요	예	아니요
모나코	MC	377	아니요	아니요	아니요	아니요
몽골	MN	976	아니요	아니요	예	아니요
몬테네그로	ME	382	아니요	아니요	예	아니요
몬트세라트 섬	MS	1-664	아니요	아니요	예	아니요
모로코	MA	212	예	아니요	예	예
모잠비크	MZ	258	아니요	아니요	아니요	아니요

국가 또는 리전	ISO 코드	다이얼링 코드	단축 코드 지원	긴 코드 지원	발신자 ID 지원	양방향 SMS 지원
미얀마	MM	95	아니요	예	예	예
N						
나미비아	NA	264	아니요	아니요	예	아니요
네팔	NP	977	아니요	아니요	예	아니요
네덜란드	NL	31	예	예	예	예
네덜란드령 안틸 제도	AN	599	아니요	아니요	예	아니요
뉴칼레도니아	NC	687	아니요	아니요	예	아니요
뉴질랜드[6]	NZ	64	예	아니요	아니요	예
니카라과	NI	505	아니요	아니요	아니요	아니요
니제르	NE	227	아니요	아니요	예	아니요
나이지리아	NG	234	아니요	아니요	예	아니요
니우에	NU	683	아니요	아니요	예	아니요
노르웨이	NO	47	아니요	예	예	예
O						
오만	OM	968	아니요	아니요	아니요	N/A
P						
파키스탄	PK	92	아니요	아니요	예	N/A
팔레스타인	PS	970	아니요	아니요	예	아니요

국가 또는 리전	ISO 코드	다이얼링 코드	단축 코드 지원	긴 코드 지원	발신자 ID 지원	양방향 SMS 지원
파나마	PA	507	아니요	아니요	예	아니요
파푸아뉴기니	PG	675	아니요	아니요	예	아니요
파라과이	PY	595	아니요	아니요	아니요	아니요
페루	PE	51	예	아니요	아니요	예
필리핀	PH	63	아니요	예	등록 필요[1]	예
폴란드	PL	48	아니요	예	예	예
포르투갈	PT	351	아니요	예	예	예
푸에르토리코	PR	1-797, 1-939	아니요	아니요	아니요	예
Q						
카타르	QA	974	아니요	아니요	등록 필요[1]	아니요
R						
콩고	CG	242	아니요	아니요	아니요	아니요
레위니옹	RE	262	아니요	아니요	예	아니요
루마니아	RO	40	아니요	예	예	예
러시아	RU	7	예	아니요	등록 필요[1]	예
르완다	RW	250	아니요	아니요	예	아니요
S						

국가 또는 리전	ISO 코드	다이얼링 코드	단축 코드 지원	긴 코드 지원	발신자 ID 지원	양방향 SMS 지원
세인트 크리스토퍼 네비스	KN	1-869	아니요	아니요	아니요	아니요
세인트루시아	LC	1-758	아니요	아니요	아니요	아니요
사모아	WS	685	아니요	아니요	예	아니요
산마리노	SM	378	아니요	아니요	예	아니요
상투메 프린시페	ST	239	아니요	아니요	예	아니요
사우디아라비아	SA	966	아니요	예[4]	등록 필요[1]	아니요
세네갈	SN	221	아니요	아니요	예	아니요
세르비아	RS	381	아니요	아니요	예	아니요
세이셸	SC	248	아니요	아니요	예	아니요
시에라리온	SL	232	아니요	아니요	예	아니요
싱가포르	SG	65	예	예	예[5]	예
슬로바키아	SK	421	아니요	예	예	아니요
슬로베니아	SI	386	아니요	아니요	예	아니요
솔로몬 제도	SB	677	아니요	아니요	예	아니요
소말리아	SO	252	아니요	아니요	예	아니요

국가 또는 리전	ISO 코드	다이얼링 코드	단축 코드 지원	긴 코드 지원	발신자 ID 지원	양방향 SMS 지원
남아프리카 공화국	ZA	27	예	예	아니요	예
대한민국	KR	82	아니요	아니요	아니요	아니요
남수단	SS	211	아니요	아니요	예	아니요
스페인	ES	34	예	예	예	예
스리랑카	LK	94	아니요	아니요	등록 필요[1]	아니요
수리남	SR	597	아니요	아니요	예	아니요
스와질란드	SZ	268	아니요	아니요	예	아니요
스웨덴	SE	46	예	예	예	예
스위스	CH	41	아니요	예	예	예
T						
대만	TW	886	아니요	예	아니요	예
타지키스탄	TJ	992	아니요	아니요	예	아니요
탄자니아	TX	255	아니요	아니요	예	아니요
태국	TH	66	아니요	예	등록 필요[1]	예
동티모르	TL	670	아니요	아니요	예	아니요
토고	TG	228	아니요	아니요	예	아니요
통가	TO	676	아니요	아니요	예	아니요
트리니다드 토바고	TT	1-868	아니요	아니요	예	아니요

국가 또는 리전	ISO 코드	다이얼링 코드	단축 코드 지원	긴 코드 지원	발신자 ID 지원	양방향 SMS 지원
튀니지	TN	216	아니요	아니요	예	아니요
터키	TR	90	아니요	아니요	등록 필요[1]	아니요
투르크메니스탄	TM	993	아니요	아니요	아니요	아니요
터크스 카이코스 군도	TC	1-649	아니요	아니요	예	아니요
투발루	TC	688	아니요	아니요	예	아니요
U						
우간다	UG	256	아니요	아니요	예	아니요
우크라이나	UA	380	아니요	예	예	예
아랍 에미리트 연합국(UAE)	AE	971	예	예	등록 필요[1]	예
영국	GB	44	예	예	예	예
미국	US	1	예	예	아니요	예
우루과이	UY	598	예	아니요	아니요	예
우즈베키스탄	UZ	998	아니요	아니요	예	아니요
V						
바누아투	VU	678	아니요	아니요	예	아니요
베네수엘라	VE	58	아니요	아니요	아니요	아니요

국가 또는 리전	ISO 코드	다이얼링 코드	단축 코드 지원	긴 코드 지원	발신자 ID 지원	양방향 SMS 지원
베트남	VN	84	아니요	아니요	등록 필요[1]	아니요
영국령 버진 제도	VG	1-284	아니요	아니요	예	아니요
미국령 버진 제도, US	VI	1-340	아니요	아니요	아니요	예
W						
X						
Y						
예멘	YE	967	아니요	아니요	예	아니요
Z						
잠비아	ZM	260	아니요	아니요	예	아니요
짐바브웨	ZW	263	아니요	아니요	예	아니요

참고

1. 발신자는 사전 등록된 알파벳 발신자 ID를 사용해야 합니다. 발신자 ID를 AWS Support요청하려면을 참조하십시오. Amazon SNS로 SMS 메시징에 대한 발신자 ID 요청 일부 국가에서는 발신자가 승인을 받기 위해 특정 요구 사항을 충족하거나 특정 제한 사항을 준수해야 합니다. 이러한 경우 AWS Support 발신자 ID 요청을 제출한 후 추가 정보를 요청하기 위해 연락을 드릴 수 있습니다.
2. 발신자는 보내려는 각 유형의 메시지에 대해 사전 등록된 템플릿을 사용해야 합니다. 발신자가 이 요구 사항을 충족하지 않으면 메시지가 차단됩니다. 템플릿을 등록하려면 를 사용하여 Amazon SNS SMS 케이스를 여십시오 AWS Support. 사례를 생성할 때 발시자 ID를 요청하기 위해 제공하는 것과 동일한 정보를 제공합니다. 자세한 정보는 Amazon SNS로 SMS 메시징에 대한 발신자 ID

요청을 참조하세요. 일부 국가에서는 발신자가 승인을 받기 위해 특정 추가 요구 사항을 충족하거나 특정 제한 사항을 준수해야 합니다. 이러한 경우 추가 정보를 요청할 AWS Support 수 있습니다.

> **Note**
>
> 중국으로 메시지를 보내려면 먼저 승인을 AWS Support 위해 템플릿을 등록해야 합니다.

3. 발신자는 사전 등록된 알파벳 발신자 ID를 사용해야 합니다. 추가 등록 단계가 필요합니다. 자세한 정보는 인도 발신자 ID 등록 요구 사항을 참조하세요.

4. 이러한 국가의 긴 코드는 수신 메시지만 지원합니다. 즉, 이러한 긴 코드를 사용하여 수신자에게 메시지를 보낼 수 없지만 수신자로부터 메시지를 수신하는 데는 사용할 수 있습니다. 발신자 ID는 아웃바운드 메시지만 지원하므로 이러한 긴 코드는 알파벳 발신자 ID를 사용하여 메시지를 보내는 경우 수신자가 수신 거부를 허용하는 데 유용한 방법입니다.

5. Amazon SNS는 싱가포르의 정보통신 미디어 개발 기관(IMDA)에서 생성한 레지스트리인 싱가포르 SMS 발신자 레지스트리(SSIR)에 등록된 발신자 ID를 사용하여 싱가포르로 SMS 트래픽을 보낼 수 있습니다. 싱가포르 발신인 ID를 사용하기 위한 요구 사항에 대한 자세한 내용은 싱가포르 발신자 ID 등록 요구 사항에서 확인하세요.

   등록되지 않은 발신자 ID 또는 짧은 코드 또는 긴 코드와 같은 대체 발신 ID 유형을 사용하여 싱가포르에서 SMS 트래픽을 보낼 수도 있습니다.

6. 전용 단축 코드가 없는 경우에도 Amazon SNS는 공유 단축 코드 풀을 사용하여 뉴질랜드 수신자에게 메시지를 전송하려고 시도합니다. 공유 번호에 대한 현지 이동 통신사의 제한으로 인해 이러한 공유 번호를 통한 전송은 최선의 방식으로 이루어집니다. 따라서 Amazon SNS에서는 뉴질랜드로 전송되는 모든 트래픽에 대해 전용 단축 코드를 구매할 것을 적극 권장합니다. URL이 포함된 메시지는 전용 단축 코드 프로세스를 통해 허용 목록에 추가되어야 합니다. 단축 코드를 구매하는 방법에 대한 자세한 내용은 Amazon SNS에서 SMS 메시징에 대한 전용 단축 코드 요청 섹션을 참조하세요.

SMS 모범 사례

휴대폰 사용자는 원치 않는 SMS 메시지를 거의 허용하지 않습니다. 원치 않는 SMS 캠페인에 대한 응답 속도는 거의 언제나 느리므로 ROI(Return on Investment)가 낮습니다.

또한 무선통신 사업자가 대량 SMS 발신자를 지속적으로 감사하여, 원치 않는 메시지를 전송하는 것으로 확인된 번호에서 보내는 메시지를 제한하거나 차단합니다.

또한 원치 않는 콘텐츠를 전송하면 AWS 이용 정책을 위반하는 것입니다. Amazon SNS 팀은 SMS 캠페인을 정기적으로 감사하여 원치 않는 메시지를 전송하는 것처럼 보일 경우 메시지 전송 기능을 제한하거나 차단할 수 있습니다.

마지막으로 대부분의 국가, 리전 및 관할권에서는 원치 않는 SMS 메시지를 보낼 경우 심각한 위약금을 부과합니다. 예를 들어, 미국의 전화 소비자 보호법(TCPA)에 따르면 소비자는 원치 않는 메시지를 수신할 때마다 $500–$1,500의 손해 배상금(발신자가 지불)을 받을 수 있습니다.

이 단원에서는 고객 참여를 개선하고 높은 위약금을 방지하는 데 도움이 되는 다양한 모범 사례에 대해 설명합니다. 그러나 이 단원에서는 법률적인 조언은 다루지 않습니다. 항상 변호사에게 문의하여 법률 자문을 받으세요.

주제

- Amazon SNS의 SMS 문자 수 한도

법률, 규정 및 통신 사업자 요구 사항 준수

고객이 거주하는 지역의 법률 및 규정을 위반하면 상당한 벌금 및 위약금이 부과될 수 있습니다. 따라서 사업을 영위하는 각 국가 및 리전의 SMS 메시징 관련 법률을 파악하고 있어야 합니다.

다음 목록에는 전 세계 주요 시장에서 SMS 통신에 적용되는 주요 법률에 대한 링크가 나와 있습니다.

- 미국: 1991년에 제정된 전화 소비자보호법(TCPA라고도 함)은 특정 유형의 SMS 메시지에 적용됩니다. 자세한 정보는 미연방 통신 위원회 웹 사이트의 rules and regulations를 참조하세요.
- 영국: PECR이라고도 하는 Privacy and Electronic Communications(EC Directive) Regulations 2003이 특정 유형의 SMS 메시지에 적용됩니다. 자세한 정보는 영국 정보감독원(Information Commissioner's Office) 웹 사이트에 게시된 What are PECR?을 참조하세요.
- 유럽 연합: ePrivacy Directive라고도 하는 Privacy and Electronic Communications Directive 2002가 일부 유형의 SMS 메시지에 적용됩니다. 자세한 정보는 Europa.eu 웹 사이트에 게시된 법률 전문을 참조하세요.
- 캐나다: 주로 캐나다 스팸 방지법 또는 CASL이라고 하는 Fighting Internet and Wireless Spam Act가 특정 유형의 SMS 메시지에 적용됩니다. 자세한 정보는 캐나다 의회 웹 사이트에 게시된 법률 전문을 참조하세요.
- 일본: Act on Regulation of Transmission of Specific Electronic Mail이 특정 유형의 SMS 메시지에 적용됩니다. 자세한 내용은 일본 총무성 웹 사이트에 게시된 스팸에 대한 일본의 대응을 참조하십시오.

발신자로서 회사 또는 조직이 이러한 국가 중 하나에 기반을 두지 않는 경우에도 이러한 법률이 적용될 수 있습니다. 이 목록에 나열된 일부 법률은 원래 원치 않는 이메일 또는 전화 통화 문제를 해결하기 위해 발의되었으나 SMS 메시지에도 적용되도록 해석 또는 확장되었습니다. 그 외 국가 및 리전에도 SMS 메시지 전송과 관련된 법률이 있을 수 있습니다. 법률적인 자문을 얻으려면 고객이 거주하는 각 국가 또는 리전의 변호사에게 문의하세요.

많은 국가에서는 현지 통신사가 궁극적으로 네트워크를 통해 어떤 종류의 트래픽이 흐르는지 결정할 권한을 가지고 있습니다. 이는 통신사가 현지 법률의 최소 요구 사항을 초과하는 SMS 콘텐츠에 제한을 적용할 수 있음을 의미합니다.

권한 획득

보내려는 특정 유형의 메시지를 받도록 명시적으로 요청하지 않은 수신자에게는 메시지를 보내지 마세요. 같은 회사 내의 조직 간에도 옵트인 목록을 공유하지 마세요.

수신자가 온라인 양식을 사용하여 메시지 수신을 등록할 수 있는 경우 자동 스크립트를 통해 자신도 모르게 사람들을 구독하지 않게 하는 시스템을 추가하세요. 또한 사용자가 단일 세션에서 전화번호를 제출할 수 있는 횟수를 제한해야 합니다.

SMS 옵트인 요청을 수신한 경우 메시지 수신을 확인하도록 요청하는 메시지를 수신자에게 전송합니다. 수신자가 구독을 확인할 때까지 해당 고객에게 추가 메시지를 보내지 마십시오. 구독 확인 메시지는 다음 예와 비슷합니다.

```
Text YES to join ExampleCorp alerts. 2 msgs/month. Msg & data rates may
apply. Reply HELP for help, STOP to cancel.
```

각 옵트인 요청 및 확인에 대한 날짜, 시간 및 원본을 포함하는 기록을 유지합니다. 그러면 통신업자 또는 규제 기관에서 해당 기록을 요청할 때와 고객 목록에 대한 정기 감사를 수행할 때 유용할 수 있습니다.

옵트인 워크플로

일부 경우(예: 미국 수신자 부담 또는 단축 코드 등록) 이동통신사에서는 전체 옵트인 워크플로의 목업 또는 스크린샷을 제공하도록 요구합니다. 목업 또는 스크린샷은 수신자가 완료하는 옵트인 워크플로와 매우 유사해야 합니다.

목업 또는 스크린샷에는 최고 수준의 규정 준수를 유지하기 위해 아래 나열된 모든 필수 공개 사항이 포함되어야 합니다.

필수 공개 사항

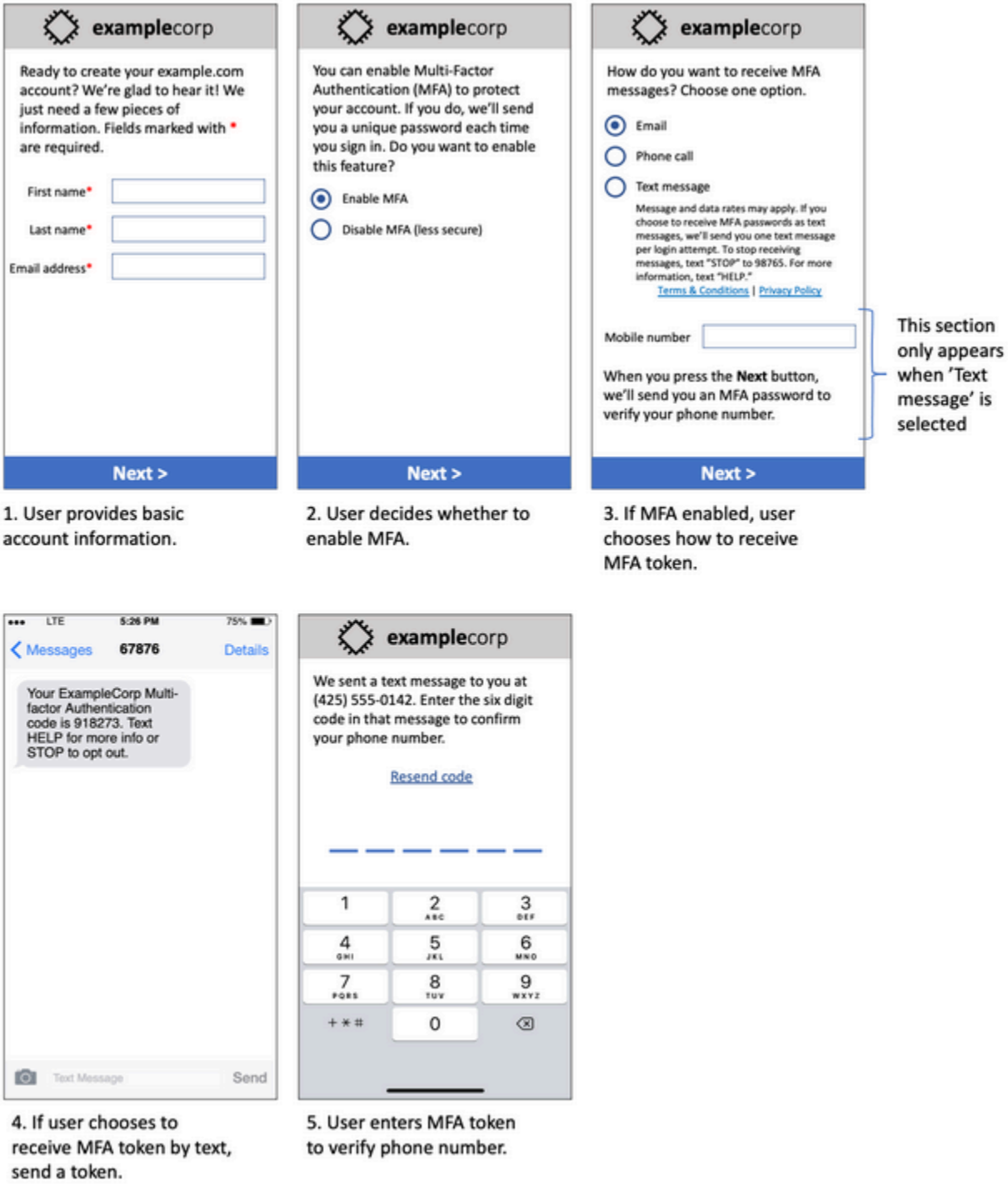
- 프로그램을 통해 보낼 메시징 사용 사례에 대한 설명
- "메시지 및 데이터 요금이 부과될 수 있습니다."라는 문구
- 수신자가 메시지를 받는 빈도 표시 예를 들어, 되풀이되는 메시징 프로그램은 '주당 메시지 1건'이라고 말할 수 있습니다. 일회성 암호 또는 다중 인증 사용 사례는 '다양한 메시지 빈도' 또는 '로그인 시도당 메시지 1건'이라고 말할 수 있습니다.
- 이용 약관 및 개인정보 처리방침 문서로 연결되는 링크

규정을 준수하지 않는 옵트인에 대한 일반적인 거부 이유

- 입력된 회사 이름이 목업이나 스크린샷에 표시된 이름과 일치하지 않는 경우. 명확하지 않은 관계는 옵트인 워크플로 설명에 설명해야 합니다.
- 메시지가 수신자에게 전송될 것으로 보이지만 그렇게 하기 전에 명시적으로 동의가 수집되지 않은 경우. 모든 메시징에는 명시적 동의가 필요합니다.

- 서비스에 가입하기 위해 문자 메시지 수신이 필요한 것으로 보이는 경우. 워크플로에서 옵트인 메시지를 받는 대신 사용할 수 있는 대안(예: 이메일이나 음성 통화 형식)을 제공하지 않는 경우에는 규정을 준수하지 않는 것입니다.
- 전체 옵트인 문구가 서비스 약관에만 표시되어 있는 경우. 공개 사항은 링크된 정책 문서 내에 포함하는 대신 항상 옵트인 시점에 수신자에게 제출해야 합니다.
- 고객이 귀하로부터 한 가지 유형의 메시지를 수신하는 데 동의하고 귀하가 다른 유형의 문자 메시지를 보내는 경우. 예를 들어 고객이 일회성 암호 수신에 동의했지만 투표 및 설문조사 메시지도 받았습니다.
- 필수 공개 사항(위에 나열됨)이 수신자에게 제공되지 않는 경우.

다음은 다중 인증 사용 사례에 대한 이동통신사의 요구 사항을 준수하는 예입니다.



다중 인증 사용 사례 목업

여기에는 최종 텍스트와 이미지가 포함되며 전체 옵트인 흐름과 주석이 함께 표시됩니다. 옵트인 흐름에서 고객은 문자 메시지 수신에 동의하고 필요한 모든 공개 내용을 포함하기 위해 명확하고 의도적인 조치를 취해야 합니다.

기타 옵트인 워크플로 유형

이동통신사는 위에 설명된 내용을 준수하는 경우 애플리케이션 및 웹사이트 외부의 옵트인 워크플로(예: 구두 또는 서면 옵트인)도 수락합니다. 규정을 준수하는 옵트인 워크플로와 구두 또는 서면 스크립트는 특정 메시지 유형을 수신하기 위해 수신자로부터 명시적인 동의를 수집합니다. 이러한 예로는 지원 에이전트가 서비스 데이터베이스에 기록하기 전에 동의를 수집하기 위해 사용하는 구두 스크립트나 판촉 전단지에 기재된 전화번호가 있습니다. 이러한 옵트인 워크플로 유형의 목업을 제공하려면 옵트인 스크립트, 마케팅 자료 또는 전화번호가 수집되는 데이터베이스의 스크린샷을 제공하면 됩니다. 옵트인이 명확하지 않거나 사용 사례가 특정 용량을 초과하는 경우 이동통신사에서 이러한 사용 사례에 대해 추가 질문을 할 수 있습니다.

이전 목록으로 보내지 않을 것

사람들은 전화번호를 자주 바꿉니다. 2년 전에 연락 동의를 수집한 전화번호는 현재 다른 사람의 번호일 수 있습니다. 새 메시징 프로그램에 이전 전화번호 목록을 사용하지 마세요. 이전 목록을 사용하면 해당 번호가 더 이상 사용되지 않아 일부 메시지가 전달되지 않을 수 있으며, 일부 수신자는 애초에 동의한 내용을 기억하지 못해 옵트아웃할 수도 있습니다.

고객 목록 감사

반복 SMS 캠페인을 전송할 경우 고객 목록을 정기적으로 감사하세요. 고객 목록 감사에서는 메시지를 수신하는 데 관심이 있는 고객에게만 메시지가 발송되는지 확인합니다.

목록을 감사할 때 각 옵트인 고객에게 구독 사실을 알리고 구독 해지 관련 정보를 제공하는 메시지를 전송합니다. 알림 메시지는 다음 예와 비슷합니다.

```
You're subscribed to ExampleCorp alerts. Msg & data rates may apply. Reply
HELP for help, STOP to unsubscribe.
```

기록 보관

각 고객이 SMS 메시지 수신을 요청한 시점과 각 고객에게 전송한 메시지가 무엇인지 알 수 있는 기록을 보관합니다. 전 세계 많은 국가 및 리전에서는 SMS 발신자에게 이러한 기록을 쉽게 검색할 수 있는 방식으로 유지 관리할 것을 요구합니다. 무선통신 사업자 역시 언제든지 이러한 정보를 요청할 수 있습니다. 이때 제공해야 하는 정확한 정보는 국가 또는 리전별로 다릅니다. 기록 보관 요건에 관한 자세한 정보는 고객이 위치한 각 국가 또는 리전의 상업용 SMS 메시징 관련 규정을 검토하세요.

이동통신사 또는 규제 기관에서 고객이 귀하의 메시지 수신을 선택했다는 증빙 자료를 요청하는 경우도 있습니다. 이러한 경우 AWS Support에서 귀하에게 연락하여 사업자 또는 기관이 요구하는 정보의

목록을 알려줄 수 있습니다. 필요한 정보를 제공할 수 없는 경우 추가적으로 SMS 메시지 전송 기능이 일시 중지될 수 있습니다.

메시지 내용은 명확, 정직, 간결하게 작성

SMS는 독특한 매체입니다. 메시지당 160자 제한은 메시지 내용이 간결해야 함을 의미합니다. 이메일과 같은 다른 커뮤니케이션 채널에서 사용할 수 있는 기술은 SMS 채널에 적용되지 않을 수 있으며 SMS 메시지와 함께 사용하면 부정직하거나 기만적이 내용처럼 보일 수도 있습니다. 메시지 내용이 모범 사례와 일치하지 않는 경우 수신자는 메시지를 무시할 수 있습니다. 최악의 경우 이동통신사에서 메시지를 스팸으로 식별하고 향후 귀하의 전화번호에서 보내는 메시지를 차단할 수 있습니다.

이 섹션에서는 효과적인 SMS 메시지 본문을 만들기 위한 몇 가지 팁과 아이디어를 제공합니다.

발신자 신원 식별

수신자는 귀하가 보낸 메시지임을 즉시 알 수 있어야 합니다. 이 모범 사례를 따르는 발신자는 각 메시지 시작 부분에 식별 이름('프로그램 이름')을 포함합니다.

금지 사항:

```
Your account has been accessed from a new device. Reply Y to confirm.
```

권장 사항:

```
ExampleCorp Financial Alerts: You have logged in to your account from a
new device. Reply Y to confirm, or STOP to opt-out.
```

메시지를 개인 간 메시지처럼 보이게 만들지 않을 것

일부 마케터는 메시지가 개인이 보낸 것처럼 보이게 하여 SMS 메시지에 개인적인 느낌을 더하고 싶어합니다. 하지만 이 기법을 사용하면 메시지가 피싱 시도처럼 보일 수 있습니다.

금지 사항:

```
Hi, this is Jane. Did you know that you can save up to 50% at
Example.com? Click here for more info: https://www.example.com.
```

권장 사항:

```
ExampleCorp Offers: Save 25-50% on sale items at Example.com. Click here
to browse the sale: https://www.example.com. Text STOP to opt-out.
```

금전적인 내용은 조심스럽게 다룰 것

사기꾼은 돈을 모으려는 사람들의 욕구를 이용합니다. 제안 내용이 믿을 수 없을 정도로 너무 좋게 보이게 하지 마세요. 돈을 미끼로 사람들을 현혹하지 마세요. 통화 기호를 사용하여 돈을 표시하지 마세요.

금지 사항:

```
Save big $$$ on your next car repair by going to https://
www.example.com.
```

권장 사항:

```
ExampleCorp Offers: Your ExampleCorp insurance policy gets you discounts
at 2300+ repair shops nationwide. More info at https://www.example.com.
Text STOP to opt-out.
```

필요한 문자만 작성

브랜드는 메시지에 ™ 또는 ®와 같은 상표 기호를 포함하여 상표를 보호하려는 경향이 있습니다. 그러나 이러한 기호는 160자 SMS 메시지에 포함될 수 있는 표준 문자 집합(GSM 알파벳이라고 함)의 일부가 아닙니다. 이러한 문자 중 하나가 포함된 메시지를 보내면 자동으로 메시지 본문당 70자만 지원하는 다른 문자 인코딩 시스템을 사용하여 메시지가 전송됩니다. 따라서 메시지가 여러 부분으로 나뉘어 질 수 있습니다. 보내는 부분 메시지 각각에 요금이 청구되므로 메시지 전체를 보내는 데 소요되는 비용이 예상보다 많을 수 있습니다. 또한 수신자는 메시지 1건이 아닌 여러 개의 메시지를 순차적으로 받을 수 있습니다. SMS 문자 인코딩에 대한 자세한 내용은 Amazon SNS의 SMS 문자 수 한도 섹션을 참조하세요.

금지 사항:

```
ExampleCorp Alerts: Save 20% when you buy a new ExampleCorp Widget® at
example.com and use the promo code WIDGET.
```

권장 사항:

```
ExampleCorp Alerts: Save 20% when you buy a new ExampleCorp Widget(R) at
example.com and use the promo code WIDGET.
```

> **Note**
>
> 앞의 두 가지 예는 거의 동일하지만 첫 번째 예에는 GSM 알파벳의 일부가 아닌 등록 상표 기호(®)가 포함되어 있습니다. 따라서 첫 번째 예는 두 건의 부분 메시지로 전송되고 두 번째 예는 하나의 메시지로 전송됩니다.

유효하고 안전한 링크 사용

메시지에 링크를 포함하는 경우 링크를 두 번 확인하여 링크가 작동하는지 확인합니다. 회사 네트워크 외부의 디바이스에서 링크를 테스트하여 링크가 제대로 연결되는지 확인합니다. SMS 메시지는 160자로 제한되어 있기 때문에 매우 긴 URL을 여러 메시지로 분할할 수 있습니다. 리디렉션 도메인을 사용하여 단축된 URL을 사용할 수 있습니다. 그러나 tinyurl.com 또는 bitly.com과 같은 무료 링크 단축 서비스를 사용하면 안 됩니다. 이동통신사는 이러한 도메인의 링크를 포함하는 메시지를 필터링하는 경향이 있기 때문입니다. 하지만 링크가 회사 또는 조직의 독점 사용 전용 도메인을 가리키는 경우에 한해 유료 링크 단축 서비스를 사용할 수 있습니다.

금지 사항:

```
Go to https://tinyurl.com/4585y8mr today for a special offer!
```

권장 사항:

```
ExampleCorp Offers: Today only, get an exclusive deal on an ExampleCorp
Widget. See https://a.co/cFKmaRG for more info. Text STOP to opt-out.
```

사용하는 약어 수 제한

SMS 채널의 160자 제한으로 인해 일부 발신자는 메시지에 약어를 광범위하게 사용해야 한다고 생각합니다. 그러나 약어를 과도하게 사용하면 많은 독자에게 전문성이 없는 것처럼 보일 수 있으며 일부 사용자가 메시지를 스팸으로 신고할 수 있습니다. 과도한 수의 약어를 사용하지 않고도 메시지를 조리 있게 작성할 수 있습니다.

금지 사항:

```
Get a gr8 deal on ExampleCorp widgets when u buy a 4-pack 2day.
```

권장 사항:

```
ExampleCorp Alerts: Today only—an exclusive deal on ExampleCorp Widgets
at example.com. Text STOP to opt-out.
```

적절한 응답

수신자가 메시지에 응답할 때 유용한 정보로 응답하는지 확인합니다. 예를 들어, 고객이 메시지 중 하나에 "HELP" 키워드로 응답할 경우 구독된 프로그램, 매월 전송할 메시지 수, 추가 정보를 문의할 수 있는 방법 등에 대한 정보를 해당 고객에게 전송합니다. HELP 응답은 다음 예와 비슷합니다.

```
HELP: ExampleCorp alerts: email help@example.com or call 425-555-0199. 2
msgs/month. Msg & data rates may apply. Reply STOP to cancel.
```

고객이 "STOP" 키워드로 회신할 경우 더 이상 메시지가 수신되지 않는다고 알려줍니다. STOP 응답은 다음 예와 비슷합니다.

```
You're unsubscribed from ExampleCorp alerts. No more messages will be sent.
Reply HELP, email help@example.com, or call 425-555-0199 for more info.
```

참여를 기반으로 전송 조정

시간에 따라 고객의 우선 순위가 변경될 수 있습니다. 고객이 메시지가 더 이상 유용하지 않다고 판단할 경우 메시지를 완전히 옵트아웃하거나 원치 않는 메시지로 보고할 수 있습니다. 따라서 고객의 참여에 따라 전송 방법을 조정하는 것이 중요합니다.

메시지에 거의 반응하지 않는 고객에 대해서는 메시지 빈도를 조정해야 합니다. 예를 들어, 참여하는 고객에게 주간 메시지를 발송하는 경우 참여도가 낮은 고객을 위해 월간 다이제스트를 별도로 만들 수 있습니다.

마지막으로 전혀 참여하지 않는 고객을 고객 목록에서 제거합니다. 이 단계는 고객이 메시지에 대해 불만을 갖는 것을 방지할 수 있습니다. 또한 비용을 절감하고 발신자로서 평판을 보호할 수 있습니다.

적절한 시간에 전송

정상 업무 시간에만 메시지를 전송합니다. 저녁 시간이나 밤중에 메시지를 전송할 경우 고객이 방해를 받지 않기 위해 목록을 구독 해제할 수 있습니다. 또한 고객이 즉시 응답할 수 없는 시간에 SMS 메시지를 전송하는 것은 좋지 않습니다.

대규모 대상에게 캠페인이나 여정을 보내는 경우 발신자 번호의 처리량을 다시 확인합니다. 수신자 수를 처리량으로 나누어 모든 수신자에게 메시지를 보내는 데 걸리는 시간을 확인합니다.

크로스 채널 피로 방지

캠페인에서 여러 통신 채널(예: 이메일, SMS, 푸시 메시지)을 사용할 경우 동일한 메시지를 모든 채널에서 보내지 마십시오. 동일한 메시지를 여러 채널에서 동시에 보낼 경우 고객이 메시지가 도움이 된다고 생각하지 않고 귀찮게 느낄 수 있습니다.

전용 단축 코드 사용

단축 코드를 사용할 경우 브랜드와 메시지 유형별로 별도의 단축 코드를 유지합니다. 예를 들어, 회사에 두 개의 브랜드가 있는 경우 브랜드별로 별도의 단축 코드를 사용합니다. 마찬가지로 트랜잭션 메시지와 프로모션 메시지를 모두 전송할 경우 각 메시지 유형에 대해 별도의 단축 코드를 사용합니다. 단축 코드를 요청하는 방법에 대한 자세한 내용은 Amazon SNS에서 SMS 메시징에 대한 전용 단축 코드 요청에서 확인하세요.

대상 전화번호 확인

Amazon SNS를 통해 SMS 메시지를 보낼 때 보내는 각 메시지 부분에 대해 요금이 청구됩니다. 메시지 부분당 지불하는 요금은 수신자의 국가 또는 지역에 따라 다릅니다. SMS 요금에 대한 자세한 내용은 Amazon SMS 요금을 참조하세요.

Amazon SNS가 SMS 메시지 전송 요청을 수락하면(SendMessages API 호출의 결과 또는 캠페인 또는 여정이 시작된 결과) 해당 메시지 전송에 대한 요금이 부과됩니다. 이 사항은 의도된 수신자가 실제로 메시지를 수신하지 않더라도 적용됩니다. 예를 들어 수신자의 전화번호가 더 이상 사용되지 않거나 메시지를 보낸 번호가 유효한 휴대전화 번호가 아닌 경우에도 메시지 전송 요금이 청구됩니다.

Amazon SNS는 SMS 메시지 전송에 대한 유효한 요청을 수락하고 전송을 시도합니다. 따라서 메시지를 보내는 대상 전화번호가 유효한 휴대폰 번호인지 확인해야 합니다. Amazon SNS 전화번호 검증 서비스를 사용하여 전화번호가 유효한지와 전화번호 유형(예: 모바일, 유선 또는 VoIP)을 확인할 수 있습니다. 자세한 내용은 Amazon Pinpoint 개발자 안내서의 Validating phone numbers in Amazon Pinpoint(Amazon Pinpoint에서 전화번호 확인)를 참조하세요.

중복성을 염두에 두고 설계하기

미션 크리티컬 메시징 프로그램의 경우 둘 이상의 AWS 리전에서 Amazon SNS를 구성하는 것이 좋습니다 Amazon SNS는 여러 AWS 리전에서 사용할 수 있습니다. Amazon SNS를 사용할 수 있는 리전의 전체 목록은 AWS 일반 참조 섹션을 참조하세요.

SMS 메시지에 사용하는 전화번호(짧은 코드, 긴 코드, 무료 전화번호, 10DLC 번호 포함)는 AWS 리전 간에 복제할 수 없습니다. 따라서 여러 리전에서 Amazon SNS를 사용하려면 Amazon SNS를 사용하

려는 각 리전에서 별도의 전화번호를 요청해야 합니다. 예를 들어 단축 코드를 사용하여 미국의 수신자에게 문자 메시지를 보내는 경우 사용할 각 AWS 리전에서 별도의 단축 코드를 요청해야 합니다.

일부 국가에서는 중복성을 추가하기 위해 여러 유형의 전화번호를 사용하기도 합니다. 예를 들어 미국에서는 단축번호, 10DLC 번호, 수신자 부담 번호를 요청할 수 있습니다. 각 전화번호 유형은 서로 다른 경로를 통해 수신자에게 전달됩니다. 동일한 AWS 리전에 있거나 여러 AWS 리전에 분산되어 있는 여러 전화번호 유형을 사용하면 추가 중복 계층이 제공되어 복원력을 향상시킬 수 있습니다.

SMS 제한 및 제약 조건

SMS 제한 및 제약 조건에 대해서는 Amazon Pinpoint 사용 설명서에서 Amazon Pinpoint의 SMS 제한 및 제약 조건을 참조하세요.

옵트아웃 키워드 관리

SMS 수신자는 디바이스에서 키워드로 회신하여 메시지를 옵트아웃할 수 있습니다. 자세한 내용은 SMS 메시지 수신 옵트아웃 섹션을 참조하세요.

CreatePool

새 풀을 생성하고 지정된 발신 ID를 풀에 연결하려면 `CreatePool` API 작업을 사용합니다. 자세한 내용은 Amazon Pinpoint SMS 및 음성 API의 CreatePool을 참조하세요.

PutKeyword

발신 전화번호 또는 풀에 대한 키워드 구성을 생성하거나 업데이트하려면 `PutKeyword` API 작업을 사용합니다. 자세한 내용은 Amazon Pinpoint SMS 및 음성 API의 PutKeyword를 참조하세요.

번호 설정 관리

SMS and voice settings(SMS 및 음성 설정) 페이지의 Number settings(번호 설정) 섹션에 있는 옵션을 사용하여 AWS Support에서 요청하여 계정에 할당한 전용 단축 코드 및 긴 코드에 대한 설정을 관리할 수 있습니다. 자세한 내용은 Amazon Pinpoint 사용 설명서의 번호 설정 관리를 참조하세요.

Amazon SNS의 SMS 문자 수 한도

SMS 메시지 하나에는 최대 140바이트의 정보를 담을 수 있습니다. SMS 메시지 하나에 포함할 수 있는 문자의 개수는 메시지에 포함된 문자의 유형에 따라 달라집니다.

메시지에서 GSM 7비트 알파벳이라고도 하는 GSM 03.38 문자 세트에 속한 문자만 사용하는 경우 문자를 최대 160개까지 포함할 수 있습니다. 메시지에 GSM 03.38 문자 집합 이외의 문자가 포함되는 경

우에는 문자를 최대 70자까지 포함할 수 있습니다. SMS 메시지를 전송할 때 Amazon SNS에서는 사용하기에 가장 효율적인 인코딩을 자동으로 결정합니다.

메시지에 최대 문자 수보다 많은 문자가 포함된 경우 메시지가 여러 부분으로 분할됩니다. 메시지가 여러 부분으로 분할되면 각 부분에는 앞에 오는 메시지 부분에 대한 추가 정보가 포함됩니다. 수신자의 디바이스가 이러한 방식으로 분리된 메시지 부분을 수신하면 이 추가 정보를 사용하여 모든 메시지 부분이 올바른 순서로 표시되도록 합니다. 수신자의 이동 통신사 및 디바이스에 따라 여러 메시지가 단일 메시지 또는 별도의 메시지 시퀀스로 표시될 수 있습니다. 따라서 각 메시지 부분의 문자 수는 153자(GSM 03.38 문자만 포함하는 메시지의 경우) 또는 67자(기타 문자를 포함하는 메시지의 경우)로 줄어듭니다. SMS 길이 계산기 도구를 사용하여 메시지를 보내기 전에 메시지에 포함된 메시지 부분의 수를 예측할 수 있습니다. 그 중 일부는 온라인에서 사용할 수 있습니다. 모든 메시지의 지원되는 최대 크기는 GSM 문자 1,600자 또는 그 외의 문자 630자입니다. 처리량 및 메시지 크기에 대한 자세한 내용은 Amazon Pinpoint 사용 설명서에서 SMS character limits in Amazon Pinpoint(Amazon Pinpoint의 SMS 문자 수 한도)를 참조하세요.

보내는 각 메시지의 메시지 부분 수를 보려면 먼저 Event stream settings(이벤트 스트림 설정)를 활성화해야 합니다. 이렇게 하면 Amazon SNS에서는 메시지가 수신자의 모바일 공급자에게 전달될 때 `_SMS.SUCCESS` 이벤트를 생성합니다. `_SMS.SUCCESS` 이벤트 레코드에는 `attributes.number_of_message_parts`라는 속성이 포함되어 있습니다. 이 속성은 메시지에 포함된 메시지 부분의 수를 지정합니다.

> **Important**
>
> 메시지 부분이 두 개 이상 포함된 메시지를 보낼 경우 메시지에 포함된 메시지 부분 수에 대한 요금이 청구됩니다.

GSM 03.38 문자 세트

다음 표에는 GSM 03.38 문자 세트에 속한 모든 문자가 나열되어 있습니다. 다음 표에 있는 문자만 포함하는 메시지를 전송하는 경우에는 메시지에 문자를 최대 160개까지 포함할 수 있습니다.

GSM 03.38 문자 세트												
A	B	C	D	E	F	G	H	I	J	K	L	M
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
a	b	c	d	e	f	g	h	i	j	k	l	m

GSM 03.38 문자 세트												
n	o	p	q	r	s	t	u	v	w	x	y	z
à	Å	å	Ä	ä	Ç	É	é	è	ì	Ñ	ñ	ò
Ø	ø	Ö	ö	ù	Ü	ü	Æ	æ	ß	0	1	2
3	4	5	6	7	8	9	&	*	@	:	,	¤
$	=	!	>	#	-	¡	¿	(	<	%	.	+
£	?	"	)	§	;	'	/	_	¥	Δ	Φ	Γ
Λ	Ω	Π	Ψ	Σ	Θ	Ξ						

GSM 03.38 문자 세트에는 앞서 보여드린 표에 있는 문자 외에 몇 가지 기호가 포함되어 있습니다. 그러나 각 기호는 다음과 같이 보이지 않는 이스케이프 문자도 포함되어 있기 때문에 문자 두 개로 계수됩니다.

- ^
- {
- }
- \
- [
-]
- ~
- |
- €

끝으로 GSM 03.38 문자 세트에는 다음과 같은 비인쇄 문자도 포함되어 있습니다.

- 공백 문자
- 텍스트 한 줄의 끝과 다른 줄의 시작을 표시하는 줄 바꿈 제어 문자
- 텍스트 한 줄의 첫머리로 이동하는 캐리지 리턴 제어 문자(일반적으로 줄 바꿈 문자 뒤에 옴)
- 위 목록의 문자에 자동으로 추가되는 이스케이프 제어 문자

예제 메시지

이 섹션에는 몇 가지 예제 SMS 메시지가 포함되어 있습니다. 각 예에 대해 이 섹션에는 메시지의 메시지 부분 수와 총 문자 수가 표시됩니다.

예제 1: GSM 03.38 알파벳의 문자만 포함하는 긴 메시지

다음 메시지에는 GSM 03.38 알파벳에 있는 문자만 포함되어 있습니다.

```
Hello Carlos. Your Example Corp. bill of $100 is now available. Autopay is scheduled for next Thursday, April 9. To view the details of your bill, go to https://example.com/bill1.
```

앞의 메시지에는 180자가 포함되어 있으므로 여러 메시지 부분으로 분할되어야 합니다. 메시지가 여러 메시지 부분으로 분할되면 각 부분은 153 GSM 03.38 문자를 포함할 수 있습니다. 그 결과, 이 메시지는 두 개의 메시지 부분으로 전송됩니다.

예제 2: 멀티바이트 문자가 포함된 메시지

다음 메시지에는 여러 중국어 문자가 포함되어 있으며 모두 GSM 03.38 알파벳 이외의 문자입니다.

```
##################################################·####1994#7#########
```

앞의 메시지에는 71자가 포함되어 있습니다. 그러나 메시지의 거의 모든 문자가 GSM 03.38 알파벳 이외의 문자이기 때문에 두 개의 메시지 부분으로 전송됩니다. 이러한 각 메시지 부분은 최대 67자를 포함할 수 있습니다.

예제 3: 단일 비GSM 문자가 포함된 메시지

다음 메시지에는 GSM 03.38 알파벳의 일부가 아닌 단일 문자가 포함되어 있습니다. 이 예에서 문자는 일반 아포스트로피(')와 다른 문자인 닫는 작은따옴표(’)입니다. Microsoft Word와 같은 워드 프로세싱 애플리케이션에서는 주로 아포스트로피가 닫는 작은따옴표로 자동으로 바뀝니다. Microsoft Word에서 SMS 메시지 초안을 작성하여 Amazon SNS에 붙여넣는 경우, 이러한 특수 문자를 제거하고 아포스트로피로 바꿔야 합니다.

```
John: Your appointment with Dr. Salazar’s office is scheduled for next Thursday at 4:30pm. Reply YES to confirm, NO to reschedule.
```

앞의 메시지에는 130자가 포함되어 있습니다. 그러나 이 메시지에는 GSM 03.38 알파벳의 일부가 아닌 닫는 작은따옴표 문자가 포함되어 있기 때문에 두 개의 메시지 부분으로 전송됩니다.

이 메시지의 닫는 작은따옴표 문자를 아포스트로피(GSM 03.38 알파벳의 일부)로 바꾸면 메시지가 단일 메시지 부분으로 전송됩니다.

모바일 푸시 알림

Amazon SNS를 사용하면 모바일 디바이스의 앱으로 알림 메시지를 직접 보낼 수 있습니다. 모바일 엔드포인트에 전송된 푸시 알림 메시지는 모바일 앱에서 메시지 알림, 배지 업데이트 또는 사운드 알림으로 나타날 수 있습니다.

주제

사용자 알림 작동 방식

푸시 알림 메시지를 모바일 디바이스와 데스크톱 모두로 전송하려면 아래의 지원되는 푸시 알림 서비스 중 하나를 사용합니다.

- Amazon Device Messaging(ADM)
- iOS 및 Mac OS X용 Apple 푸시 알림 서비스(APNS)
- Baidu 클라우드 푸시(Baidu)
- Firebase Cloud Messaging(FCM)
- Windows Phone용 Microsoft 푸시 알림 서비스(MPNS)
- Windows 푸시 알림 서비스(WNS)

APNS 및 FCM과 같은 푸시 알림 서비스는 이러한 서비스를 사용하도록 등록된 해당 모바일 디바이스 및 각 앱과 연결을 유지합니다. 앱과 모바일 디바이스가 등록되면 푸시 알림 서비스는 디바이스 토큰

을 반환합니다. Amazon SNS는 디바이스 토큰을 사용하여 직접 푸시 알림 메시지를 보낼 수 있는 모바일 엔드포인트를 생성합니다. Amazon SNS가 다른 푸시 알림 서비스와 통신할 수 있도록 허용하기 위해 Amazon SNS에 푸시 알림 서비스 자격 증명을 제출합니다. 자세한 정보는 사용자 알림 프로세스 개요에서 확인하세요.

직접 푸시 알림 메시지를 전송하는 것 외에도 주제를 구독하는 모바일 엔드포인트에 Amazon SNS를 사용하여 메시지를 보낼 수도 있습니다. Amazon SNS란 무엇인가요?에 설명된 대로 Amazon SQS, HTTP/S, 이메일, SMS 등 다른 엔드포인트 유형에서 주제를 구독하는 것과 동일한 개념입니다. 차이점은 구독 모바일 디바이스가 주제에 전송된 푸시 알림 메시지를 받도록 하기 위해 Amazon SNS가 푸시 알림 서비스를 사용하여 통신한다는 것입니다.

사용자 알림 프로세스 개요

1. 지원하려는 모바일 플랫폼에 대한 보안 인증 정보 및 디바이스 토큰을 얻습니다.
2. 보안 인증 정보를 사용하여 Amazon SNS를 사용하는 플랫폼 애플리케이션 객체(`PlatformApplicationArn`)를 생성합니다. 자세한 정보는 플랫폼 애플리케이션 생성을 참조하세요.
3. 반환된 보안 인증 정보를 사용하여 푸시 알림 서비스로부터 모바일 앱 및 디바이스의 디바이스 토큰을 요청합니다. 받은 토큰은 해당 모바일 앱과 디바이스를 나타냅니다.
4. 디바이스 토큰 및 `PlatformApplicationArn`을 사용하여 Amazon SNS를 사용하는 플랫폼 엔드포인트 객체(`EndpointArn`)를 생성합니다. 자세한 정보는 플랫폼 엔드포인트 생성에서 확인하세요.
5. `EndpointArn`을 사용하여 모바일 디바이스의 앱에 메시지를 게시합니다. 자세한 내용은 모바일 디바이스에 게시 및 Amazon Simple Notification Service API 참조의 게시 API를 참조하세요.

모바일 앱 설정

이 섹션에서는 에서 설명한 정보와 AWS Management Console 함께 를 사용하여 모바일 애플리케이션을 설정하는 Amazon SNS 사용자 알림에 대한 사전 조건 방법을 설명합니다.

주제

- Amazon SNS 사용자 알림에 대한 사전 조건
- 플랫폼 애플리케이션 생성
- 플랫폼 엔드포인트 생성
- 디바이스 토큰 또는 등록 ID 추가
- Apple 인증 방법

- Firebase Cloud Messaging(FCM) 인증 방법
- Firebase 클라우드 메시징 (FCM) 엔드포인트 관리

Amazon SNS 사용자 알림에 대한 사전 조건

Amazon SNS 모바일 푸시 알림을 사용하려면 다음이 필요합니다.

- 지원되는 푸시 알림 서비스(ADM, APN, Baidu, FCM, MPNS 또는 WNS) 중 하나에 연결하기 위한 자격 증명 집합
- 모바일 앱 및 디바이스를 위한 디바이스 토큰 또는 등록 ID
- 모바일 엔드포인트에 알림 메시지를 보내도록 구성된 Amazon SNS
- 지원되는 푸시 알림 서비스 중 하나를 사용하도록 등록 및 구성된 모바일 앱

애플리케이션을 푸시 알림 서비스에 등록하려면 여러 단계를 수행해야 합니다. Amazon SNS가 모바일 엔드포인트로 직접 푸시 알림 메시지를 보내도록 하려면 푸시 알림 서비스에 몇 가지 정보를 제공해야 합니다. 일반적으로 푸시 알림 서비스에 연결하기 위한 자격 증명, 푸시 알림 서비스로부터 받은 디바이스 토큰 또는 등록 ID(해당 모바일 디바이스 및 모바일 앱의 ID) 및 푸시 알림 서비스에 등록된 모바일 앱이 필요합니다.

자격 증명이 취하는 정확한 형식은 모바일 플랫폼 간에 다르지만, 모든 경우에 플랫폼에 연결하는 동안에는 이 자격 증명을 제출해야 합니다. 모바일 앱마다 자격 증명 집합 하나가 발급되며, 이는 해당 앱의 인스턴스에 메시지를 보내는 데 사용해야 합니다.

구체적인 이름은 사용하는 푸시 알림 서비스에 따라 다릅니다. 예를 들어, APN을 푸시 알림 서비스로 사용할 경우 디바이스 토큰이 필요합니다. 또는 FCM을 사용할 때에는 디바이스 토큰에 해당하는 것을 등록 ID라고 합니다. 디바이스 토큰 또는 등록 ID는 모바일 디바이스의 운영 체제에서 애플리케이션에 보내는 문자열입니다. 특정 모바일 디바이스에서 실행되는 모바일 앱의 인스턴스를 고유하게 식별하며, 이 앱-디바이스 쌍의 고유한 식별자라고 생각하면 됩니다.

Amazon SNS는 플랫폼 애플리케이션 리소스로 다른 몇 개의 설정과 함께 자격 증명을 저장합니다. 역시 몇 가지 추가 설정이 있는 디바이스 토큰이 플랫폼 엔드포인트라는 객체로 표현됩니다. 각 플랫폼 엔드포인트는 한 개의 특정 플랫폼 애플리케이션에 속하며, 모든 플랫폼 엔드포인트는 해당 플랫폼 애플리케이션에 저장된 자격 증명을 사용해 통신할 수 있습니다.

다음 단원에서는 지원되는 각각의 푸시 알림 서비스에 대한 필수 조건을 설명합니다. 필수 정보를 모두 갖췄으면 AWS Management Console 또는 Amazon SNS 모바일 푸시 API를 사용하여 푸시 알림 메시지를 보낼 수 있습니다. 자세한 정보는 사용자 알림 프로세스 개요을 참조하십시오.

플랫폼 애플리케이션 생성

Amazon SNS가 모바일 엔드포인트에 알림 메시지를 보내려면 직접 전송인지 주제 구독을 통하는지와 관계없이 반드시 플랫폼 애플리케이션을 먼저 생성해야 합니다. 앱이 AWS에 등록된 후, 다음 단계는 앱과 모바일 디바이스의 엔드포인트를 만드는 것입니다. 그러면 Amazon SNS는 엔드포인트를 앱과 디바이스에 전송하는 데 사용됩니다.

플랫폼 애플리케이션을 생성하려면

1. Amazon SNS 콘솔에 로그인합니다.
2. 탐색 창에서 Mobile(모바일)을 선택한 다음, Push notifications(푸시 알림)를 선택합니다.
3. Platform applications(플랫폼 애플리케이션) 섹션에서 Create platform application(플랫폼 애플리케이션 생성)을 선택합니다.

   모바일 애플리케이션을 생성할 수 있는 AWS 리전 목록은 모바일 애플리케이션에 지원되는 리전에서 확인하세요.
4. Application name(애플리케이션 이름)에 앱을 나타내는 이름을 입력합니다.

   앱 이름은 대문자 및 소문자 ASCII 문자, 숫자, 밑줄, 하이픈, 마침표로만 이루어져야 합니다. 이름 역시 1~256자여야 합니다.
5. Push notification platform(푸시 알림 플랫폼)의 경우 앱이 등록되는 플랫폼을 선택한 다음, 적절한 보안 인증 정보를 입력합니다.

   > **Note**
   >
   > Apple 푸시 알림 서비스(APNS) 플랫폼 중 하나를 사용하고 있는 경우 토큰 또는 인증서 기반 인증 중에서 선택한 다음 Choose file(파일 선택)을 선택하여 .p8 또는 .p12 파일(키체인 접근에서 내보낸 파일)을 Amazon SNS에 업로드할 수 있습니다.
6. 플랫폼 애플리케이션 생성을 선택합니다.

   그러면 앱이 Amazon SNS에 등록됩니다. 이 서비스는 선택한 플랫폼에 대한 플랫폼 애플리케이션 객체를 만든 다음 해당 PlatformApplicationArn을 반환합니다.

플랫폼 엔드포인트 생성

앱과 모바일 디바이스가 푸시 알림 서비스에 등록되면 푸시 알림 서비스는 디바이스 토큰을 반환합니다. Amazon SNS는 디바이스 토큰을 사용하여 직접 푸시 알림 메시지를 보낼 수 있는 모바일 엔드포인

트를 생성합니다. 자세한 정보는 Amazon SNS 사용자 알림에 대한 사전 조건 및 사용자 알림 프로세스 개요에서 확인하세요.

이 섹션에서는 플랫폼 엔드포인트를 생성할 때 권장되는 방법에 대해 설명합니다.

주제

- 플랫폼 엔드포인트 생성
- 의사(Pseudo) 코드
- AWS SDK 예제
- 문제 해결

플랫폼 엔드포인트 생성

Amazon SNS로 앱에 알림을 푸시하려면 먼저 플랫폼 엔드포인트 생성 작업을 호출해 Amazon SNS에 해당 앱의 디바이스 토큰을 등록해야 합니다. 이 작업은 플랫폼 애플리케이션의 Amazon 리소스 이름(ARN)과 디바이스 토큰을 파라미터로 취하고 생성된 플랫폼 엔드포인트의 ARN을 반환합니다.

이 `CreatePlatformEndpoint`작업은 다음과 같은 작업을 수행합니다.

- 플랫폼 엔드포인트가 이미 있는 경우 다시 생성하지 마십시오. 기존 플랫폼 엔드포인트의 ARN 호출자로 돌아가십시오.
- 디바이스 토큰은 같지만 설정이 다른 플랫폼 엔드포인트가 이미 있는 경우 다시 생성하지 마십시오. 호출자에 예외를 발생시키십시오.
- 플랫폼 엔드포인트가 없으면 생성하세요. 새로 생성한 플랫폼 엔드포인트의 ARN 호출자로 돌아가십시오.

앱이 시작할 때마다 바로 플랫폼 엔드포인트 생성 작업을 호출해서는 안 됩니다. 이 방법이 항상 작동하는 엔드포인트를 제공하는 것은 아니기 때문입니다. 이러한 상황은 예를 들면 같은 디바이스에서 앱을 제거했다가 다시 설치하고, 이에 대한 엔드포인트가 이미 있지만 비활성화된 경우에 발생할 수 있습니다. 성공적으로 등록하면 다음과 같은 결과를 얻을 수 있어야 합니다.

1. 이 앱-디바이스 조합에 대해 플랫폼 엔드포인트가 있는지 확인합니다.
2. 플랫폼 엔드포인트의 디바이스 토큰이 유효한 최신 디바이스 토큰인지 확인합니다.
3. 플랫폼 엔드포인트가 활성화되어 있고 사용할 준비가 되어 있는지 확인합니다.

의사(Pseudo) 코드

다음 의사(pseudo) 코드는 다양한 시작 조건에서 작동하고 활성화된 현재 플랫폼 엔드포인트를 생성하는 데 권장되는 사례를 설명합니다. 이 방법은 앱을 처음 등록하는지 아닌지, 이 앱의 플랫폼 엔드포인트가 이미 있는지, 플랫폼 엔드포인트가 활성화되어 있는지, 올바른 디바이스 토큰이 있는지 여부에 상관없이 효과가 있습니다. 연이어 여러 번 호출해도 중복 플랫폼 엔드포인트가 생성되거나, 이미 최신 상태이고 활성화된 경우에 기존 플랫폼 엔드포인트가 변경되지 않으므로 무관합니다.

```
retrieve the latest device token from the mobile operating system
if (the platform endpoint ARN is not stored)
  # this is a first-time registration
  call create platform endpoint
  store the returned platform endpoint ARN
endif

call get endpoint attributes on the platform endpoint ARN

if (while getting the attributes a not-found exception is thrown)
  # the platform endpoint was deleted
  call create platform endpoint with the latest device token
  store the returned platform endpoint ARN
else
  if (the device token in the endpoint does not match the latest one) or
      (get endpoint attributes shows the endpoint as disabled)
    call set endpoint attributes to set the latest device token and then enable the
 platform endpoint
  endif
endif
```

이 방법은 앱에서 자체적으로 등록하거나 다시 등록하려 할 때마다 사용할 수 있습니다. Amazon SNS에 디바이스 토큰 변경을 알리는 데에도 사용됩니다. 이 경우 최신 디바이스 토큰 값으로 작업을 호출하면 됩니다. 이 방법에 대해 유의해야 할 몇 가지 사항은 다음과 같습니다.

- 플랫폼 엔드포인트 생성 작업을 호출할 수도 있는 두 가지 경우가 있습니다. 최초 등록 중에서와 같이 앱에서 자신의 고유한 플랫폼 엔드포인트 ARN을 인식하지 못하는 맨 처음에 호출할 수 있습니다. 애플리케이션에서 해당 엔드포인트 ARN을 인식하지만 삭제된 경우와 같이 초기 엔드포인트 속성 가져오기 작업 호출이 찾을 수 없음 예외와 함께 실패할 경우에도 호출될 수 있습니다.
- 방금 플랫폼 엔드포인트를 생성했더라도 플랫폼 엔드포인트의 상태를 확인하기 위해 엔드포인트 속성 가져오기 작업이 호출됩니다. 이러한 상황은 플랫폼 엔드포인트가 이미 있지만 비활성화되어 있는 경우에 발생합니다. 이 경우 플랫폼 엔드포인트 생성 작업이 성공하지만 플랫폼 엔드포인트가 활

성화되지 않기 때문에 성공을 반환하기 전에 플랫폼 엔드포인트의 상태를 다시 한 번 확인해야 합니다.

AWS SDK 예제

다음 코드는 SDK에서 제공하는 Amazon SNS 클라이언트를 사용하여 이전 유사 코드를 구현하는 방법을 보여줍니다. AWS

AWS SDK를 사용하려면 자격 증명으로 구성해야 합니다. 자세한 정보는 AWS SDK 및 도구 참조 가이드의 공유 구성 및 자격 증명 파일을 참조하세요.

CLI

AWS CLI

플랫폼 애플리케이션 엔드포인트를 생성하려면

다음 `create-platform-endpoint` 예제에서는 지정된 토큰을 사용하여 지정된 플랫폼 애플리케이션의 엔드포인트를 생성합니다.

```
aws sns create-platform-endpoint \
    --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/GCM/
MyApplication \
    --token EXAMPLE12345...
```

출력:

```
{
      "EndpointArn": "arn:aws:sns:us-west-2:1234567890:endpoint/GCM/
MyApplication/12345678-abcd-9012-efgh-345678901234"
}
```

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointRequest;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
 */

public class RegistrationExample {
    public static void main(String[] args) {
        final String usage = """

                Usage:     <token> <platformApplicationArn>

                Where:
                   token - The name of the FIFO topic.\s
                   platformApplicationArn - The ARN value of platform
 application. You can get this value from the AWS Management Console.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String token = args[0];
        String platformApplicationArn = args[1];
```

```
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createEndpoint(snsClient, token, platformApplicationArn);
    }

    public static void createEndpoint(SnsClient snsClient, String token, String
 platformApplicationArn) {
        System.out.println("Creating platform endpoint with token " + token);
        try {
            CreatePlatformEndpointRequest endpointRequest =
 CreatePlatformEndpointRequest.builder()
                    .token(token)
                    .platformApplicationArn(platformApplicationArn)
                    .build();

            CreatePlatformEndpointResponse response =
 snsClient.createPlatformEndpoint(endpointRequest);
            System.out.println("The ARN of the endpoint is " +
 response.endpointArn());
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

자세한 정보는 모바일 푸시 API 작업을 참조하세요.

문제 해결

오래된 디바이스 토큰으로 플랫폼 엔드포인트 생성 반복 호출

특히 FCM 엔드포인트의 경우 애플리케이션이 처음 발급된 기기 토큰을 저장한 다음 애플리케이션을 시작할 때마다 해당 기기 토큰으로 플랫폼 엔드포인트 생성을 호출하는 것이 최선이라고 생각할 수 있습니다. 이렇게 하면 앱에서 디바이스 토큰의 상태를 관리하지 않아도 되고 Amazon SNS에서 디바이스 토큰을 최신 값으로 자동 업데이트하기 때문에 올바르게 보일 수도 있습니다. 하지만 이 해결 방법에는 여러 가지 심각한 문제가 있습니다.

- Amazon SNS는 만료된 디바이스 토큰을 새 디바이스 토큰으로 업데이트하기 위해 FCM의 피드백에 의존합니다. FCM은 일정 기간 동안 이전 디바이스 토큰에 대한 정보를 유지하지만 무기한은 아닙니다. FCM에서 이전 디바이스 토큰과 새 디바이스 토큰 사이의 연결에 대해 잊어버리면 Amazon SNS가 플랫폼 엔드포인트에 저장된 디바이스 토큰을 올바른 값으로 더 이상 업데이트하지 않고, 대신 플랫폼 엔드포인트를 비활성화합니다.
- 플랫폼 애플리케이션에는 동일한 디바이스 토큰에 해당하는 여러 플랫폼 엔드포인트가 포함되어 있습니다.
- Amazon SNS는 동일한 디바이스 토큰으로 시작해서 플랫폼 엔드포인트를 몇 개나 생성할 수 있는지에 대한 할당량을 둡니다. 결국 유효하지 않은 파라미터 예외 및 "이 엔드포인트가 이미 다른 토큰으로 등록되었습니다." 오류 메시지와 함께 새 엔드포인트 생성이 실패합니다.

FCM 엔드포인트 관리에 대한 자세한 내용은 을 참조하십시오. Firebase 클라우드 메시징 (FCM) 엔드포인트 관리

유효하지 않은 디바이스 토큰과 연결된 플랫폼 엔드포인트 다시 활성화

모바일 플랫폼(예: APN 또는 FCM)에서 Amazon SNS에 게시 요청에 사용된 디바이스 토큰이 유효하지 않다고 알리면 Amazon SNS에서 해당 디바이스 토큰과 연결된 플랫폼 엔드포인트를 비활성화합니다. 그리고 나서 Amazon SNS는 해당 디바이스 토큰에 대한 후속 게시를 거부합니다. 플랫폼 엔드포인트를 다시 활성화하고 계속 게시하는 것이 최선이라고 생각할지도 모르겠지만, 대부분의 경우에서 이렇게 하면 아무 효과가 없습니다. 즉, 게시되는 메시지가 전송되지 않고 플랫폼 엔드포인트가 이후에 곧 다시 비활성화됩니다.

플랫폼 엔드포인트와 연결된 디바이스 토큰이 실제로 유효하지 않기 때문입니다. 더 이상 설치된 앱에 해당하지 않기 때문에 여기에 전송해도 실패하는 것입니다. 다음에 게시되면 모바일 플랫폼에서 다시 한 번 Amazon SNS에 디바이스 토큰이 유효하지 않다고 알려 주고, Amazon SNS가 또 다시 플랫폼 엔드포인트를 비활성화합니다.

비활성화된 플랫폼 엔드포인트를 다시 활성화하려면 설정된 엔드포인트 속성 작업 호출과 함께 유효한 디바이스 토큰과 연결한 후에 활성화해야 합니다. 그래야만 해당 플랫폼 엔드포인트에 전송해도 성공할 수 있습니다. 해당 장치 토큰을 업데이트하지 않고 플랫폼 엔드포인트를 다시 활성화해도 효과가 있는 유일한 경우는 해당 엔드포인트와 연결된 장치 토큰이 이전에는 유효하지 않았다가 다시 유효해진 경우입니다. 이러한 상황은 예를 들면 같은 모바일 디바이스에서 앱을 제거했다가 다시 설치하고, 같은 디바이스 토큰을 받을 경우에 발생할 수 있습니다. 위에 소개한 방법은 이와 같이 연결된 디바이스 토큰이 제공되는 가장 최신 디바이스 토큰임을 확인한 후에만 플랫폼 엔드포인트를 다시 활성화하도록 합니다.

디바이스 토큰 또는 등록 ID 추가

앱과 모바일 디바이스를 Apple 푸시 알림 서비스(APN) 및 Firebase Cloud Messaging(FCM)과 같은 알림 서비스에 처음 등록할 때 디바이스 토큰 또는 등록 ID가 알림 서비스에서 반환됩니다. 디바이스 토큰 또는 등록 ID를 Amazon SNS에 추가하면 이 디바이스 토큰 또는 등록 ID는 `PlatformApplicationArn` API에서 앱과 디바이스에 대한 엔드포인트를 만드는 데 사용됩니다. Amazon SNS에서 엔드포인트를 만들면 `EndpointArn`이 반환됩니다. `EndpointArn`은 Amazon SNS가 어떤 앱과 모바일 디바이스에 알림 메시지를 전송할지를 아는 방법입니다.

다음 방법을 사용하여 디바이스 토큰과 등록 ID를 Amazon SNS에 추가할 수 있습니다.

- AWS Management Console을 사용하여 단일 토큰을 AWS에 수동으로 추가
- `CreatePlatformEndpoint` API를 사용하여 여러 토큰을 업로드
- 향후 앱을 설치할 디바이스에서 토큰을 등록

디바이스 토큰 또는 등록 ID를 수동으로 추가하려면

1. Amazon SNS 콘솔에 로그인합니다.
2. 탐색 창에서 모바일을 선택한 다음 푸시 알림을 선택합니다.
3. 플랫폼 애플리케이션 섹션에서 애플리케이션을 선택한 다음 편집을 선택합니다. 플랫폼 애플리케이션을 생성하지 않은 경우 지금 생성합니다. 작업 방법에 대한 지침은 플랫폼 애플리케이션 생성을(를) 참조하세요.
4. 엔드포인트 추가를 선택합니다.
5. [Endpoint Token] 상자에서 알림 서비스에 따라 토큰 ID 또는 등록 ID를 입력합니다. 예를 들어, ADM 및 FCM의 경우 등록 ID를 입력합니다.
6. (선택 사항) 사용자 데이터 상자에 엔드포인트와 연결할 임의의 정보를 입력합니다. Amazon SNS는 이 데이터를 사용하지 않습니다. 데이터는 UTF-8 형식이어야 하며 2KB 미만이어야 합니다.
7. 마지막으로 Add Endpoints(엔드포인트 추가)를 선택합니다.

   이제 엔드포인트가 만들어졌으므로 모바일 디바이스에 메시지를 직접 전송하거나 주제를 구독하는 모바일 디바이스에 메시지를 전송할 수 있습니다.

CreatePlatformEndpoint API를 사용하여 여러 토큰을 업로드하려면

다음 단계에서는 AWS에서 제공하는 샘플 Java 앱(bulkupload 패키지)을 사용하여 여러 토큰(디바이스 토큰 또는 등록 ID)을 Amazon SNS에 업로드하는 방법을 보여 줍니다. 이 샘플 앱을 사용하여 기존 토큰 업로드를 시작할 수 있습니다.

> **Note**
>
> 다음 단계에서는 Eclipse Java IDE를 사용합니다. 이 단계는 AWS SDK for Java를 설치했고 AWS 계정에 대한 AWS 보안 자격 증명을 가지고 있다고 전제합니다. 자세한 내용은 AWS SDK for Java 섹션을 참조하세요. 자격 증명에 대한 자세한 정보는 AWS 일반 참조의 보안 자격 증명을 가져오려면 어떻게 해야 합니까?를 참조하세요.

1. snsmobilepush.zip 파일을 다운로드하여 압축을 풉니다.
2. Eclipse에서 새 Java 프로젝트를 생성합니다.
3. SNSSamples 폴더를 새로 만든 Java 프로젝트의 최상위 디렉터리로 가져옵니다. Eclipse에서 Java 프로젝트 이름을 마우스 오른쪽 버튼으로 선택한 후 가져오기를 선택하고 일반을 확장한 후 파일 시스템, 다음을 차례로 선택한 다음 SNSSamples 폴더로 이동하여 확인, 마침을 차례로 선택합니다.
4. OpenCSV 라이브러리의 사본을 다운로드하여 bulkupload 패키지의 빌드 경로에 추가합니다.
5. bulkupload 패키지에 있는 BulkUpload.properties 파일을 엽니다.
6. 다음을 BulkUpload.properties로 추가하십시오:

   - 엔드포인트를 추가하려는 ApplicationArn.
   - 토큰이 포함된 CSV 파일 위치의 절대 경로.
   - Amazon SNS에서 올바르게 구문 분석한 토큰과 실패한 토큰을 기록하기 위해 작성할 CSV 파일의 이름(예: goodTokens.csv 및 badTokens.csv).
   - (선택 사항) 토큰이 포함된 CSV 파일에서 구분 기호와 따옴표를 지정할 문자.
   - (선택 사항) 엔드포인트를 동시에 만들기 위해 사용할 스레드 수. 기본 값은 1 스레드입니다.

   완성된 BulkUpload.properties는 다음과 같습니다.

```
applicationarn:arn:aws:sns:us-west-2:111122223333:app/FCM/fcmpushapp
csvfilename:C:\\mytokendirectory\\mytokens.csv
goodfilename:C:\\mylogfiles\\goodtokens.csv
```

```
badfilename:C:\\mylogfiles\\badtokens.csv
delimiterchar:'
quotechar:"
numofthreads:5
```

7. BatchCreatePlatformEndpointSample.java 애플리케이션을 실행하여 토큰을 Amazon SNS에 업로드합니다.

   이 예에서 엔드포인트를 만들기 위해 Amazon SNS에 성공적으로 업로드된 토큰은 `goodTokens.csv`에 로그되고, 잘못된 형식의 토큰은 `badTokens.csv`에 로그됩니다. 또한 Eclipse 콘솔에 기록된 STD OUT 로그에는 다음과 같은 내용이 포함됩니다.

```
<1>[SUCCESS] The endpoint was created with Arn arn:aws:sns:us-
west-2:111122223333:app/FCM/fcmpushapp/165j2214-051z-3176-b586-138o3d420071
<2>[ERROR: MALFORMED CSV FILE] Null token found in /mytokendirectory/mytokens.csv
```

향후 앱을 설치할 디바이스에서 토큰을 등록하려면

다음 두 가지 옵션 중 하나를 사용할 수 있습니다.

- Amazon Cognito 서비스 사용: 모바일 앱에서 Amazon SNS 플랫폼 애플리케이션과 관련된 엔드포인트를 만들려면 자격 증명이 필요합니다. 일정 기간 이후 만료되는 임시 자격 증명을 사용하는 것이 좋습니다. 대부분의 시나리오에서 Amazon Cognito를 사용하여 임시 보안 자격 증명을 만들 것을 권장합니다. 자세한 정보는 Amazon Cognito 개발자 안내서를 참조하세요. 앱이 Amazon SNS에 등록될 때 알림을 수신하려는 경우 새 엔드포인트 ARN을 제공하는 Amazon SNS 이벤트를 수신하도록 등록할 수 있습니다. `ListEndpointByPlatformApplication` API를 사용하여 Amazon SNS에 등록된 전체 엔드포인트 목록을 얻을 수도 있습니다.
- 프록시 서버 사용: 모바일 앱이 각 설치에서 호출하고 등록하기 위한 애플리케이션 인프라가 이미 설정된 경우 이 설정을 계속 사용할 수 있습니다. 서버는 프록시로 작용하여 저장하려는 모든 사용자 데이터와 함께 디바이스 토큰을 Amazon SNS 모바일 푸시 알림에 전달합니다. 이렇게 하기 위해 프록시 서버를 AWS 자격 증명을 사용하여 Amazon SNS에 연결하고 `CreatePlatformEndpoint` API 호출을 사용하여 토큰 정보를 업로드합니다. 새로 만든 엔드포인트 ARN(Amazon 리소스 이름)이 반환되고, 서버는 Amazon SNS에 대한 후속 게시 호출을 위해 이 정보를 저장할 수 있습니다.

Apple 인증 방법

앱 개발자임을 식별하는 정보를 제공하여 Amazon SNS가 iOS나 macOS 앱에 푸시 알림을 보낼 수 있는 권한을 부여할 수 있습니다. 인증하려면 플랫폼 애플리케이션을 생성할 때 키 또는 인증서를 제공하십시오. 두 가지 모두 Apple 개발자 계정에서 얻을 수 있습니다.

토큰 서명 키

Amazon SNS가 Apple 푸시 알림 서비스(APNS) 인증 토큰에 서명하는 데 사용하는 프라이빗 서명 키입니다.

서명 키를 입력하면 Amazon SNS는 사용자가 푸시 알림을 보낼 때마다 토큰을 사용하여 APNS에 인증합니다. 이 서명 키로 APNS 프로덕션 환경 및 샌드박스 환경에 푸시 알림을 보낼 수 있습니다.

서명 키는 만료되지 않으며 여러 앱에 대해 동일한 서명 키를 사용할 수 있습니다. 자세한 내용은 Apple 웹 사이트의 개발자 계정 도움말 섹션의 인증 토큰을 사용하여 APNS와 커뮤니케이션하기를 참조하십시오.

Certificate

푸시 알림을 보낼 때 Amazon SNS가 APNS에 인증하기 위해 사용하는 TLS 인증서입니다. 이 인증서는 Apple 개발자 계정에서 얻을 수 있습니다.

인증서는 1년 후에 만료됩니다. 만료되는 경우, 새 인증서를 생성하고 이를 Amazon SNS에 제공해야 합니다. 자세한 내용은 Apple 개발자 웹 사이트에서 APNS에 대한 인증서 기반 연결 설정을 참조하십시오.

AWS 관리 콘솔을 사용하여 APNS 설정을 관리하려면

1. Amazon SNS 콘솔에 로그인합니다.
2. 모바일(Mobile)에서 푸시 알림(Push notifications)을 선택합니다.
3. APNS 설정을 편집할 애플리케이션(Application)을 선택한 다음 편집(Edit)을 선택합니다.
4. 편집(Edit) 페이지의 인증 유형(Authentication type)에서 토큰(Token) 또는 인증서(Certificate)를 선택합니다.
5. 인증서 또는 토큰 서명 키에 적합한 자격 증명을 로드합니다. 이 정보는 Apple 개발자 계정에서 얻을 수 있습니다.
6. 선택한 인증 유형에 따라 다음 중 하나를 수행합니다.
   - 토큰(Token)을 선택한 경우 Apple 개발자 계정에서 얻은 다음 정보를 입력합니다. Amazon SNS가 인증 토큰을 만들려면 이 정보가 필요합니다.

- 서명 키(Signing key)— .p8 파일로 다운로드하는 Apple 개발자 계정의 인증 토큰 서명 키입니다. Apple에서는 서명 키를 한 번만 다운로드할 수 있습니다.
- 서명 키 ID(Signing key ID) – 서명 키에 할당된 ID입니다. Amazon SNS가 인증 토큰을 만들려면 이 정보가 필요합니다. Apple 개발자 계정에서 이 값을 찾으려면 인증서, ID 및 프로파일(Certificates, IDs & Profiles)을 선택하고 키(Keys) 섹션에서 원하는 키를 선택합니다.
- 팀 식별자(Team identifier) – Apple 개발자 계정 팀에 할당된 ID입니다. 이 값은 멤버십(Membership) 페이지에서 확인할 수 있습니다.
- 번들 식별자(Bundle identifier) – iOS 앱에 할당된 ID입니다. 이 값을 찾으려면 인증서, ID 및 프로파일(Certificates, IDs & Profiles)을 선택하고 식별자(Identifiers) 섹션에서 앱 ID(App IDs)를 선택한 다음 원하는 앱을 선택합니다.

- 인증서(Certificate)를 선택한 경우 다음 정보를 제공합니다.
  - SSL 인증서(SSL certificate) – TLS 인증서용 .p12 파일입니다. Apple 개발자 계정에서 인증서를 다운로드하여 설치한 다음 키체인 접근에서 이 파일을 내보낼 수 있습니다.
  - 인증서 암호(Certificate password) – 인증서에 암호를 할당했으면 여기서 지정하십시오.

7. 변경 작업을 마치면 변경 사항 저장을 선택합니다.

Firebase Cloud Messaging(FCM) 인증 방법

이 주제에서는 API와 함께 사용할 필수 FCM API (HTTP v1) 자격 증명을 Google로부터 받는 방법과 AWS AWS CLI AWS Management Console

주제

- 전제 조건
- FCM 설정 관리(API)
- FCM 설정 관리(CLI)
- FCM 설정 관리(콘솔)

> **Important**
>
> 2023년 6월 20일 — 구글은 파이어베이스 클라우드 메시징 (FCM) 레거시 HTTP API를 지원 중단했습니다. Amazon SNS는 이제 FCM HTTP v1 API를 사용하여 모든 디바이스 유형으로의 전송을 지원합니다. 중단이 발생하지 않도록 기존 모바일 푸시 애플리케이션을 2024년 6월 1일 또는 그 이전에 최신 FCM HTTP v1 API로 마이그레이션하는 것이 좋습니다.

2024년 1월 18일 — Amazon SNS는 안드로이드 디바이스에 모바일 푸시 알림을 전송하기 위한 FCM HTTP v1 API에 대한 지원을 도입했습니다.
2024년 3월 26일 — Amazon SNS는 애플 디바이스 및 웹푸시 대상을 위한 FCM HTTP v1 API를 지원합니다. 애플리케이션 중단을 방지하려면 2024년 6월 1일 또는 그 이전에 기존 모바일 푸시 애플리케이션을 최신 FCM HTTP v1 API로 마이그레이션하는 것이 좋습니다.

앱 개발자임을 식별하는 정보를 제공하여 Amazon SNS가 애플리케이션에 푸시 알림을 보낼 수 있는 권한을 부여할 수 있습니다. 인증하려면 플랫폼 애플리케이션을 만들 때 API 키 또는 토큰을 제공하십시오. Firebase 애플리케이션 콘솔에서 다음 정보를 얻을 수 있습니다.

API 키

API 키는 Firebase의 레거시 API를 직접 호출할 때 사용되는 보안 인증 정보입니다. Google은 2024년 6월 20일에 FCM 레거시 API를 삭제할 예정입니다. 현재 API 키를 플랫폼 보안 인증 정보로 사용하고 있다면 토큰을 옵션으로 선택하고 Firebase 애플리케이션용 관련 JSON 파일을 업로드하여 플랫폼 보안 인증 정보를 업데이트할 수 있습니다.

토큰

HTTP v1 API를 호출할 때는 수명이 짧은 액세스 토큰이 사용됩니다. 이것은 푸시 알림 전송을 위한 Firebase의 추천 API입니다. 액세스 토큰을 생성하기 위해 Firebase는 개발자에게 프라이빗 키 파일(service.json 파일이라고도 함) 형태의 보안 인증 정보 세트를 제공합니다.

전제 조건

Amazon SNS에서 FCM 설정을 관리하기 시작하려면 먼저 FCM service.json 보안 인증 정보를 받아야 합니다. service.json 보안 인증 정보를 얻으려면 Google Firebase 설명서에서 레거시 FCM API에서 HTTP v1으로 마이그레이션 섹션을 참조하십시오.

FCM 설정 관리(API)

API를 사용하여 FCM 푸시 알림을 만들 수 있습니다. AWS AWS 계정 내 Amazon SNS 리소스의 수와 크기는 제한되어 있습니다. 자세한 내용은 안내서의 Amazon 단순 알림 서비스 엔드포인트 및 할당량을 참조하십시오.AWS 일반 참조

Amazon SNS 주제 (AWS API) 와 함께 FCM 푸시 알림을 만들려면

키 보안 인증을 사용하는 경우, `PlatformCredential`은 `API key`입니다. 토큰 보안 인증을 사용하는 경우, `PlatformCredential`은 JSON 형식의 프라이빗 키 파일입니다.

- `CreatePlatformApplication`

기존 Amazon SNS 주제 (AWS API) 에 대한 FCM 자격 증명 유형을 검색하려면

다음 보안 인증 유형 `"AuthenticationMethod": "Token"` 또는 `"AuthenticationMethod": "Key"`를 검색합니다.

- GetPlatformApplicationAttributes

기존 Amazon SNS 주제에 대한 FCM 속성 설정(AWS API)

FCM 속성을 설정합니다.

- SetPlatformApplicationAttributes

FCM 설정 관리(CLI)

AWS Command Line Interface (CLI) 를 사용하여 FCM 푸시 알림을 만들 수 있습니다. AWS 계정 내 Amazon SNS 리소스의 수와 크기는 제한되어 있습니다. 자세한 내용은 Amazon Simple Notification Service 엔드포인트 및 할당량을 참조하세요.

Amazon SNS 주제와 함께 FCM 푸시 알림 생성(AWS CLI)

키 보안 인증을 사용하는 경우, `PlatformCredential`은 `API key`입니다. 토큰 보안 인증을 사용하는 경우, `PlatformCredential`은 JSON 형식의 프라이빗 키 파일입니다. AWS CLI를 사용할 경우 파일은 문자열 형식이어야 하며 특수 문자는 무시해야 합니다. Amazon SNS는 파일 형식을 올바르게 지정하기 위해 다음 명령을 사용할 것을 권장합니다: `SERVICE_JSON=`jq @json <<< cat service.json``:

- create-platform-application

기존 Amazon SNS 주제에 대한 FCM 보안 인증 유형 검색(AWS CLI)

다음 보안 인증 유형 `"AuthenticationMethod": "Token"` 또는 `"AuthenticationMethod": "Key"`를 검색합니다.

- get-platform-application-attributes

기존 Amazon SNS 주제에 대한 FCM 속성 설정(AWS CLI)

FCM 속성을 설정합니다.

- set-platform-application-attributes

FCM 설정 관리(콘솔)

애플리케이션이 FCM에 연결하는 데 사용하는 보안 인증 정보를 입력하려면 다음 단계를 따르십시오.

1. Amazon SNS 콘솔에 로그인합니다.
2. 모바일(Mobile)에서 푸시 알림(Push notifications)을 선택합니다.
3. 기존 FCM 애플리케이션을 선택하고 편집을 선택합니다. 플랫폼 애플리케이션을 이미 생성하지 않은 경우에는 플랫폼 애플리케이션 생성 섹션을 참조하십시오.
4. 편집 페이지의 Firebase Cloud Messaging 보안 인증에서 토큰 또는 키를 선택합니다. Firebase 애플리케이션 콘솔에서 다음 정보를 가져올 수 있습니다.
   - 토큰을 선택한 경우 유효한 프라이빗 키 파일을 업로드하십시오. 이 파일의 내용은 알림을 보낼 때 수명이 짧은 액세스 토큰을 생성하는 데 사용됩니다.
   - 키를 선택한 경우 Google API 키를 입력합니다.
5. 변경 작업을 마치면 변경 사항 저장을 선택합니다.

관련 주제

- 아마존 SNS에서 구글 파이어베이스 클라우드 메시징 (FCM) v1 페이로드 사용

Firebase 클라우드 메시징 (FCM) 엔드포인트 관리

주제

- 디바이스 토큰 관리 및 유지 관리
- 유효하지 않은 토큰 감지
- 오래된 토큰 제거

디바이스 토큰 관리 및 유지 관리

다음 단계에 따라 모바일 애플리케이션의 푸시 알림 전송 가능성을 보장할 수 있습니다.

1. 모든 디바이스 토큰, 해당 Amazon SNS 엔드포인트 ARN, 타임스탬프를 애플리케이션 서버에 저장합니다.

2. 오래된 토큰을 모두 제거하고 해당하는 Amazon SNS 엔드포인트 ARN을 삭제합니다.

앱을 처음 시작하면 디바이스에 대한 디바이스 토큰 (등록 토큰이라고도 함) 을 받게 됩니다. 이 기기 토큰은 기기의 운영체제에서 발행되며 FCM 애플리케이션에 연결됩니다. 이 디바이스 토큰을 받으면 Amazon SNS에 플랫폼 엔드포인트로 등록할 수 있습니다. 디바이스 토큰, Amazon SNS 플랫폼 엔드포인트 ARN, 타임스탬프를 애플리케이션 서버나 다른 영구 스토어에 저장하여 저장하는 것이 좋습니다. 디바이스 토큰을 검색하고 저장하도록 FCM 애플리케이션을 설정하려면 Google Firebase 설명서의 등록 토큰 검색 및 저장을 참조하십시오.

토큰을 관리하는 것이 중요합니다. up-to-date 사용자의 기기 토큰은 다음과 같은 조건에서 변경될 수 있습니다.

1. 모바일 애플리케이션이 새 디바이스에 복원됩니다.
2. 사용자가 애플리케이션을 제거하거나 업데이트합니다.
3. 사용자가 애플리케이션 데이터를 지웁니다.

디바이스 토큰이 변경되면 해당 Amazon SNS 엔드포인트를 새 토큰으로 업데이트하는 것이 좋습니다. 이렇게 하면 Amazon SNS가 등록된 디바이스와 계속 통신할 수 있습니다. 모바일 애플리케이션 내에 다음 유사 코드를 구현하여 이 작업을 수행할 수 있습니다. 활성화된 플랫폼 엔드포인트를 만들고 유지 관리하기 위한 권장 방법을 설명합니다. 이 접근 방식은 모바일 애플리케이션이 시작될 때마다 실행하거나 백그라운드에서 예약된 작업으로 실행할 수 있습니다.

의사(Pseudo) 코드

다음 FCM 유사 코드를 사용하여 기기 토큰을 관리하고 유지하세요.

```
retrieve the latest token from the mobile OS
if (endpoint arn not stored)
    # first time registration
    call CreatePlatformEndpoint
    store returned endpoint arn
endif

call GetEndpointAttributes on the endpoint arn

if (getting attributes encountered NotFound exception)
    #endpoint was deleted
    call CreatePlatformEndpoint
    store returned endpoint arn
```

```
else
    if (token in endpoint does not match latest) or
        (GetEndpointAttributes shows endpoint as disabled)
        call SetEndpointAttributes to set the
                    latest token and enable the endpoint
    endif
endif
```

토큰 업데이트 요구사항에 대해 자세히 알아보려면 Google Firebase 문서에서 정기적으로 토큰 업데이트를 참조하세요.

유효하지 않은 토큰 감지

잘못된 디바이스 토큰으로 메시지가 FCM v1 엔드포인트로 발송되면 Amazon SNS는 다음 예외 중 하나를 수신합니다.

- UNREGISTERED(HTTP 404) — Amazon SNS에서 이 예외를 수신하면 엔드포인트와 연결된 플랫폼 토큰이 유효하지 않은 상태에서 전송 실패 이벤트를 받게 됩니다. `FailureType` `InvalidPlatformToken` `FailureMessage` Amazon SNS는 전송이 실패하고 이 예외가 발생하면 플랫폼 엔드포인트를 비활성화합니다.
- INVALID_ARGUMENT(HTTP 400) — Amazon SNS에서 이 예외를 수신하면 디바이스 토큰 또는 메시지 페이로드가 유효하지 않음을 의미합니다. 자세한 내용은 ErrorCodeGoogle의 Firebase 설명서를 참조하십시오.

두 경우 모두 반환될 `INVALID_ARGUMENT` 수 있으므로 Amazon SNS는 알림 본문의 중 하나를 반환하고 알림 본문은 유효하지 않습니다. `FailureType` `InvalidNotification` `FailureMessage` 이 오류가 발생하면 페이로드가 올바른지 확인하십시오. 올바르면 디바이스 토큰이 up-to-date 맞는지 확인하십시오. Amazon SNS는 전송이 실패하고 이 예외가 발생해도 플랫폼 엔드포인트를 비활성화하지 않습니다.

`InvalidPlatformToken`전송 실패 이벤트가 발생하는 또 다른 경우는 등록된 디바이스 토큰이 해당 메시지를 전송하려는 애플리케이션에 속하지 않는 경우입니다. 이 경우 구글은 SENDER_ID_MISMATCH 오류를 반환합니다. Amazon SNS는 전송이 실패하고 이 예외가 발생하면 플랫폼 엔드포인트를 비활성화합니다.

FCM v1 API에서 수신한 모든 관찰 오류 코드는 애플리케이션의 전송 상태 로깅을 설정할 CloudWatch 때 사용할 수 있습니다.

애플리케이션의 전송 이벤트를 수신하려면 을 참조하십시오. 사용 가능한 애플리케이션 이벤트

오래된 토큰 제거

엔드포인트 디바이스로의 메시지 전송이 실패하기 시작하면 토큰은 유효하지 않은 것으로 간주됩니다. Amazon SNS는 이러한 오래된 토큰을 플랫폼 애플리케이션의 비활성화된 엔드포인트로 설정합니다. 비활성화된 엔드포인트에 게시하면 Amazon SNS는 엔드포인트가 꺼지고 엔드포인트는 비활성화된 상태로 *EventDeliveryFailure* 이벤트를 반환합니다. `FailureType` `EndpointDisabled` `FailureMessage` 애플리케이션에 대한 전송 이벤트를 수신하려면 을 참조하십시오사용 가능한 애플리케이션 이벤트.

Amazon SNS에서 이 오류를 수신하면 플랫폼 애플리케이션에서 오래된 토큰을 제거하거나 업데이트해야 합니다.

모바일 푸시 알림 전송

이 섹션에서는 모바일 푸시 알림을 보내는 방법을 설명합니다.

주제

- 주제에 게시
- 모바일 디바이스에 게시
- 플랫폼별 페이로드를 사용한 게시

주제에 게시

Amazon SNS를 사용하여 주제를 구독하는 모바일 엔드포인트에 메시지를 전송할 수도 있습니다. Amazon SNS란 무엇인가요?에 설명된 대로 Amazon SQS, HTTP/S, 이메일, SMS 등 다른 엔드포인트 유형에서 주제를 구독하는 것과 동일한 개념입니다. 차이점은 Amazon SNS가 Apple 푸시 알림 서비스(APNS) 및 Google Firebase Cloud Messaging(FCM)과 같은 알림 서비스를 통해 통신한다는 것입니다. 알림 서비스를 통해 구독된 모바일 엔드포인트는 주제로 전송된 알림을 수신합니다.

모바일 디바이스에 게시

모바일 디바이스에서 애플리케이션을 나타내는 엔드포인트에 Amazon SNS 푸시 알림 메시지를 직접 전송할 수 있습니다.

직접 메시지를 전송하려면

1. Amazon SNS 콘솔에 로그인합니다.
2. 탐색 창에서 Push notifications(푸시 알림)를 선택합니다.

3. 모바일 푸시 알림 페이지의 플랫폼 애플리케이션 섹션에서 애플리케이션 이름 (예:) 을 선택합니다 ***MyApp***.
4. ***MyApp***페이지의 엔드포인트 섹션에서 엔드포인트를 선택한 다음 메시지 게시를 선택합니다.
5. 엔드포인트에 메시지 게시(Publish message to endpoint) 페이지에서, 모바일 디바이스에서 애플리케이션에 나타나는 메시지를 입력한 후 메시지 게시(Publish message)를 선택합니다.

   Amazon SNS는 알림 메시지를 플랫폼 알림 서비스로 전송하고, 해당 메시지가 애플리케이션에 전송되게 합니다.

플랫폼별 페이로드를 사용한 게시

AWS Management Console 또는 Amazon SNS API를 사용하여 플랫폼별 페이로드가 포함된 사용자 지정 메시지를 모바일 디바이스로 전송할 수 있습니다. Amazon SNS API 사용에 대한 자세한 내용은 모바일 푸시 API 작업 및 `snsmobilepush.zip`의 `SNSMobilePush.java` 파일을 참조하세요.

주제

- JSON 형식 메시지 전송 중
- 플랫폼별 메시지 전송 중
- 다중 플랫폼의 애플리케이션으로 메시지 전송 중
- 경보 또는 백그라운드 알림으로서 메시지를 APNS로 전송
- 아마존 SNS에서 구글 파이어베이스 클라우드 메시징 (FCM) v1 페이로드 사용

JSON 형식 메시지 전송 중

플랫폼별 페이로드를 전송할 때 데이터는 JSON 키-값 쌍 문자열 형식이어야 하며, 따옴표가 이스케이프되어 있어야 합니다.

다음 예제는 FCM 플랫폼에 대한 사용자 지정 메시지를 보여줍니다.

```
{
"GCM": "{\"fcmV1Message\": {\"message\": {\"notification\": {\"title\": \"Hello\",
 \"body\": \"This is a test.\"}, \"data\": {\"dataKey\": \"example\"}}}}"
}
```

플랫폼별 메시지 전송 중

사용자 지정 데이터를 키-값 쌍으로 전송할 수 있을 뿐 아니라, 플랫폼별 키-값 쌍을 전송할 수도 있습니다.

FCM `data` 파라미터에서 사용자 지정 데이터 키-값 쌍 뒤에 FCM 파라미터 `time_to_live` 및 `collapse_key`가 포함되어 있음을 보여줍니다.

```
{
"GCM": "{\"fcmV1Message\": {\"message\": {\"notification\": {\"title\": \"TitleTest\",
 \"body\": \"Sample message for Android or iOS endpoints.\"}, \"data\":{\"time_to_live
\": 3600,\"collapse_key\":\"deals\"}}}}"
}
```

Amazon SNS에서 지원하는 각 푸시 알림 서비스에서 지원되는 키-값 쌍 목록은 다음을 참조하세요.

> **Important**
>
> Amazon SNS는 이제 안드로이드 디바이스에 모바일 푸시 알림을 전송하기 위한 파이어베이스 클라우드 메시징 (FCM) HTTP v1 API를 지원합니다.
> 2024년 3월 26일 — Amazon SNS는 애플 디바이스 및 웹푸시 대상을 위한 FCM HTTP v1 API를 지원합니다. 애플리케이션 중단을 방지하려면 2024년 6월 1일 또는 그 이전에 기존 모바일 푸시 애플리케이션을 최신 FCM HTTP v1 API로 마이그레이션하는 것이 좋습니다.

- APNS 설명서의 페이로드 키 참조
- FCM 설명서의 Firebase Cloud Messaging HTTP 프로토콜
- ADM 설명서의 메시지 보내기

다중 플랫폼의 애플리케이션으로 메시지 전송 중

FCM 및 APNS와 같은 다중 플랫폼용 디바이스에 설치된 애플리케이션에 메시지를 전송하려면 먼저 모바일 엔드포인트에서 Amazon SNS의 주제를 구독한 다음 메시지를 주제에 게시해야 합니다.

다음 예제에서는 APNS, FCM 및 ADM에서 구독 중인 모바일 엔드포인트에 전송할 메시지를 보여 줍니다.

```
{
  "default": "This is the default message which must be present when publishing a
 message to a topic. The default message will only be used if a message is not present
 for
one of the notification platforms.",
  "APNS": "{\"aps\":{\"alert\": \"Check out these awesome deals!\",\"url\":
\"www.amazon.com\"} }",
```

```
  "GCM": "{\"data\":{\"message\":\"Check out these awesome deals!\",\"url\":
\"www.amazon.com\"}}",
  "ADM": "{\"data\":{\"message\":\"Check out these awesome deals!\",\"url\":
\"www.amazon.com\"}}"
}
```

경보 또는 백그라운드 알림으로서 메시지를 APNS로 전송

Amazon SNS는 `alert` 또는 `background` 알림으로 APNS에 메시지를 보낼 수 있습니다(자세한 내용은 APNS 설명서의 앱에 백그라운드 업데이트 푸시 참조).

- `alert` APNS 알림은 경보 메시지를 표시하거나 사운드를 재생하거나 애플리케이션 아이콘에 배지를 추가하여 사용자에게 알립니다.
- `background` APNS 알림은 사용자에게 알리지 않고 알림의 내용을 실행하도록 애플리케이션을 가동하거나 명령합니다.

사용자 지정 APNS 헤더 값 지정

Amazon SNS `Publish` API 작업, AWS SDK 또는 를 사용하여 `AWS.SNS.MOBILE.APNS.PUSH_TYPE` 예약된 메시지 속성에 사용자 지정 값을 지정하는 것이 좋습니다. AWS CLI다음 CLI 예제는 지정된 주제에 `content-available`을 1로 설정하고 `apns-push-type`을 `background`로 설정합니다.

```
aws sns publish \
--endpoint-url https://sns.us-east-1.amazonaws.com \
--target-arn arn:aws:sns:us-east-1:123456789012:endpoint/APNS_PLATFORM/MYAPP/1234a567-
bc89-012d-3e45-6fg7h890123i \
--message '{"APNS_PLATFORM":"{\"aps\":{\"content-available\":1}}"}' \
--message-attributes '{ \
  "AWS.SNS.MOBILE.APNS.TOPIC":
{"DataType":"String","StringValue":"com.amazon.mobile.messaging.myapp"}, \
  "AWS.SNS.MOBILE.APNS.PUSH_TYPE":{"DataType":"String","StringValue":"background"} \
  "AWS.SNS.MOBILE.APNS.PRIORITY":{"DataType":"String","StringValue":"5"}}', \
--message-structure json
```

페이로드에서 APNS 푸시 유형 헤더 추론

`apns-push-type` APNS 헤더를 설정하지 않으면 Amazon SNS에서 JSON 형식 APNS 페이로드 구성의 `aps` 사전에 있는 `content-available` 키에 따라 헤더를 `alert` 또는 `background`로 설정합니다.

> **Note**
>
> Amazon SNS는 `alert` 또는 `background` 헤더만 유추할 수 있지만 `apns-push-type` 헤더는 다른 값으로 설정할 수 있습니다.

- 다음의 경우 `apns-push-type`이 `alert`로 설정됩니다.
  - `aps` 사전에 `1`로 설정된 `content-available`과 사용자 상호 작용을 트리거하는 하나 이상의 키가 포함되어 있는 경우.
  - `aps` 사전에 `0`으로 설정된 `content-available`이 포함되어 있거나 `content-available` 키가 없는 경우.
  - `content-available` 키의 값이 정수 또는 부울이 아닌 경우.
- 다음의 경우 `apns-push-type`이 `background`로 설정됩니다.
  - `aps` 사전에 `1`로 설정된 `content-available`만 포함되어 있고 사용자 상호 작용을 트리거하는 다른 키가 없는 경우.

> **Important**
>
> Amazon SNS에서 APNS에 대한 원시 구성 객체를 백그라운드 전용 알림으로 전송하는 경우, `1`로 설정된 `content-available`을 `aps` 사전에 포함해야 합니다. 사용자 지정 키를 포함할 수 있지만 `aps` 사전에는 사용자 상호 작용을 트리거하는 키(예: 경보, 배지 또는 사운드)를 포함할 수 없습니다.

다음은 원시 구성 객체의 예입니다.

```
{
  "APNS": "{\"aps\":{\"content-available\":1},\"Foo1\":\"Bar\",\"Foo2\":123}"
}
```

이 예에서 Amazon SNS는 메시지에 대한 `apns-push-type` APNS 헤더를 `background`로 설정합니다. Amazon SNS는 `apn` 사전에 `1`로 설정된 `content-available` 키가 포함되어 있고 사용자 상호 작용을 트리거할 수 있는 다른 키가 포함되어 있지 않음을 감지하면 헤더를 `background`로 설정합니다.

아마존 SNS에서 구글 파이어베이스 클라우드 메시징 (FCM) v1 페이로드 사용

Amazon SNS는 FCM HTTP v1 API를 사용하여 안드로이드, iOS 및 웹푸시 대상으로 알림을 전송할 수 있도록 지원합니다. 이 주제에서는 CLI 또는 Amazon SNS API를 사용하여 모바일 푸시 알림을 게시할 때의 페이로드 구조의 예를 제공합니다.

FCM 알림을 보낼 때 페이로드에 다음 메시지 유형을 포함할 수 있습니다.

- 데이터 메시지 - 데이터 메시지는 클라이언트 앱에서 처리되며 맞춤 키-값 쌍을 포함합니다. 데이터 메시지를 구성할 때는 JSON 객체를 값으로 포함하는 `data` 키를 포함시킨 다음 사용자 지정 키-값 쌍을 입력해야 합니다.
- 알림 메시지 또는 디스플레이 메시지 — 알림 메시지에는 FCM SDK에서 처리하는 사전 정의된 키 세트가 포함됩니다. 이러한 키는 전달하려는 기기 유형에 따라 달라집니다. 플랫폼별 알림 키에 대한 자세한 내용은 다음을 참조하십시오.
  - Android 알림 키
  - APNS 알림 키
  - 웹푸시 알림 키

FCM 메시지 유형에 대한 자세한 내용은 Google Firebase 문서의 메시지 유형을 참조하세요.

목차

FCM v1 페이로드 구조를 사용하여 메시지 보내기

FCM 애플리케이션을 처음 만들거나 FCM v1 기능을 활용하려는 경우 FCM v1 형식의 페이로드를 전송하도록 옵트인할 수 있습니다. 이렇게 하려면 최상위 키를 포함해야 합니다. `fcmV1Message` FCM v1 페이로드를 구성하는 방법에 대한 자세한 내용은 Google Firebase 문서에서 기존 FCM API에서 HTTP v1으로 마이그레이션하기 및 플랫폼 간 메시지 맞춤설정을 참조하세요.

Amazon SNS로 전송된 FCM v1 예제 페이로드:

> **Note**
>
> Amazon SNS를 사용하여 알림을 게시할 때는 다음 예제에 사용된 GCM 키 값을 문자열로 인코딩해야 합니다.

```
{
  "GCM": "{
     \"fcmV1Message\": {
        \"validate_only\" : false,
        \"message\" :
            {
            \"notification\": {
                \"title\": \"string\",
                \"body\": \"string\"
            },
            \"data\": {
                \"dataGen\": \"priority message\",
            },
            \"android\": {
                \"priority\": \"high\",
                \"notification\": {
                    \"body_loc_args\": [
                    \"string\"
                    ],
                    \"title_loc_args\": [
                    \"string\"
                    ],
                    \"sound\": \"string\",
                    \"title_loc_key\": \"string\",
                    \"title\": \"string\",
                    \"body\": \"string\",
                    \"click_action\": \"clicky_clacky\",
                    \"body_loc_key\": \"string\"
                },
                \"data\": {
                    \"dataAndroid\": \"priority message\",
                },
                \"ttl\": \"10023.32s\"
            },
            \"apns\": {
                \"payload\": {
```

```
                    \"aps\": {
                        \"alert\": {
                            \"subtitle\": \"string\",
                            \"title-loc-args\": [
                            \"string\"
                            ],
                            \"title-loc-key\": \"string\",
                            \"loc-args\": [
                            \"string\"
                            ],
                            \"loc-key\": \"string\",
                            \"title\": \"string\",
                            \"body\": \"string\"
                        },
                        \"category\": \"Click\",
                        \"content-available\": 0,
                        \"sound\": \"string\",
                        \"badge\": 5
                    }
                }
            },
            \"webpush\": {
                \"notification\": {
                    \"badge\": \"5\",
                    \"title\": \"string\",
                    \"body\": \"string\"
                },
                \"data\": {
                    \"dataWeb\": \"priority message\",
                }
            }
        }
    }
  }"
}
```

JSON 페이로드를 전송할 때는 요청에 `message-structure` 속성을 포함하고 로 설정해야 합니다. `json`

CLI 예제:

```
aws sns publish --topic $TOPIC_ARN --message '{"GCM": "{\"fcmV1Message\": {\"message\":
{\"notification\":{\"title\":\"string\",\"body\":\"string\"},\"android\":{\"priority
```

```
\":\"high\",\"notification\":{\"title\":\"string\",\"body\":\"string\"},\"data\":
{\"customAndroidDataKey\":\"custom key value\"},\"ttl\":\"0s\"},\"apns\":{\"payload
\":{\"aps\":{\"alert\":{\"title\":\"string\", \"body\":\"string\"},\"content-
available\":1,\"badge\":5}}},\"webpush\":{\"notification\":{\"badge\":\"URL\",\"body
\":\"Test\"},\"data\":{\"customWebpushDataKey\":\"priority message\"}},\"data\":
{\"customGeneralDataKey\":\"priority message\"}}}}", "default": "{\"notification\":
 {\"title\": \"test\"}"}' --region $REGION --message-structure json
```

FCM v1 형식의 페이로드를 전송하는 방법에 대한 자세한 내용은 Google Firebase 문서의 다음 내용을 참조하세요.

- 기존 FCM API에서 HTTP v1로 마이그레이션하세요.
- FCM 메시지에 대한 정보
- REST 리소스: projects.messages

기존 페이로드 구조를 사용하여 FCM v1 API로 메시지 보내기

FCM v1로 마이그레이션할 때 기존 자격 증명에 사용하던 페이로드 구조를 변경할 필요가 없습니다. Amazon SNS는 페이로드를 새로운 FCM v1 페이로드 구조로 변환하여 Google에 전송합니다.

입력 메시지 페이로드 형식:

```
{
  "GCM": "{\"notification\": {\"title\": \"string\", \"body\": \"string\",
 \"android_channel_id\": \"string\", \"body_loc_args\": [\"string\"], \"body_loc_key\":
 \"string\", \"click_action\": \"string\", \"color\": \"string\", \"icon\": \"string
\", \"sound\": \"string\", \"tag\": \"string\", \"title_loc_args\": [\"string\"],
 \"title_loc_key\": \"string\"}, \"data\": {\"message\": \"priority message\"}}"
}
```

Google로 전송된 메시지:

```
{
  "message": {
    "token": "***",
    "notification": {
      "title": "string",
      "body": "string"
    },
    "android": {
      "priority": "high",
```

```
      "notification": {
        "body_loc_args": [
          "string"
        ],
        "title_loc_args": [
          "string"
        ],
        "color": "string",
        "sound": "string",
        "icon": "string",
        "tag": "string",
        "title_loc_key": "string",
        "title": "string",
        "body": "string",
        "click_action": "string",
        "channel_id": "string",
        "body_loc_key": "string"
      },
      "data": {
        "message": "priority message"
      }
    },
    "apns": {
      "payload": {
        "aps": {
          "alert": {
            "title-loc-args": [
              "string"
            ],
            "title-loc-key": "string",
            "loc-args": [
              "string"
            ],
            "loc-key": "string",
            "title": "string",
            "body": "string"
          },
          "category": "string",
          "sound": "string"
        }
      }
    },
    "webpush": {
      "notification": {
```

```
        "icon": "string",
        "tag": "string",
        "body": "string",
        "title": "string"
      },
      "data": {
        "message": "priority message"
      }
    },
    "data": {
      "message": "priority message"
    }
  }
}
```

잠재적 위험

- 레거시에서 v1으로의 매핑은 Apple 푸시 알림 서비스 (APNS) `headers` 또는 키를 지원하지 않습니다. `fcm_options` 이 필드를 사용하려면 FCM v1 페이로드를 보내세요.
- FCM v1에서는 APNs 기기로 자동 알림을 보낼 때 메시지 헤더가 필요한 경우도 있습니다. 현재 APNs 기기로 자동 알림을 보내고 있는 경우 기존 접근 방식에서는 작동하지 않습니다. 대신 FCM v1 페이로드를 사용하여 예상치 못한 문제를 방지하는 것이 좋습니다. APN 헤더 목록과 용도를 찾으려면 Apple 개발자 안내서의 APN과의 통신을 참조하십시오.
- 알림을 보낼 때 TTL Amazon SNS 속성을 사용하는 경우 `android` 현장에서만 업데이트됩니다. TTLAPNS 속성을 설정하려면 FCM v1 페이로드를 사용하세요.
- `android`,`apns`, 및 `webpush` 키가 매핑되고 제공된 모든 관련 키로 채워집니다. 예를 들어 세 플랫폼 모두에서 공유하는 키인 키를 제공하는 `title` 경우 FCM v1 매핑은 세 플랫폼 모두에 제공한 제목을 채웁니다.
- 플랫폼 간에 공유되는 일부 키에는 다른 값 유형이 필요합니다. 예를 들어, `apns` 전달된 `badge` 키에는 정수 값이 필요하고 `webpush` 전달된 `badge` 키에는 문자열 값이 필요합니다. `badge`키를 제공하는 경우 FCM v1 매핑은 유효한 값을 제공한 키만 채웁니다.

FCM 전송 실패 이벤트

다음 표에는 FCM v1 알림 요청에 대해 Google에서 받은 오류/상태 코드에 해당하는 Amazon SNS 실패 유형이 나와 있습니다. FCM v1 API에서 수신한 모든 관찰 오류 코드는 애플리케이션의 전송 상태 로깅을 설정할 CloudWatch 때 확인할 수 있습니다.

FCM 오류/상태 코드	Amazon SNS 장애 유형	오류 메시지	원인 및 완화
UNREGISTERED	InvalidPlatformToken	엔드포인트와 연결된 플랫폼 토큰이 유효하지 않습니다.	엔드포인트에 연결된 디바이스 토큰이 오래되었거나 유효하지 않습니다. Amazon SNS가 엔드포인트를 비활성화했습니다. Amazon SNS 엔드포인트를 최신 디바이스 토큰으로 업데이트합니다.
INVALID_ARGUMENT	InvalidNotification	알림 본문이 유효하지 않습니다.	디바이스 토큰 또는 메시지 페이로드가 유효하지 않을 수 있습니다. 메시지 페이로드가 유효한지 확인하세요. 메시지 페이로드가 유효하면 Amazon SNS 엔드포인트를 최신 디바이스 토큰으로 업데이트하십시오.
SENDER_ID_MISMATCH	InvalidPlatformToken	엔드포인트와 연결된 플랫폼 토큰이 유효하지 않습니다.	디바이스 토큰과 연결된 플랫폼 애플리케이션에는 디바이스 토큰으로 전송할 권한이 없습니다. Amazon SNS 플랫폼 애플리케이션에서 올바른 FCM 자격 증명을 사용하고 있는지 확인하십시오.
UNAVAILABLE	DependencyUnavailable	종속성은 확인할 수 없습니다.	FCM이 요청을 제때 처리하지 못했습니다.

FCM 오류/상태 코드	Amazon SNS 장애 유형	오류 메시지	원인 및 완화
			Amazon SNS에서 실행한 모든 재시도가 실패했습니다. 이러한 메시지를 DLQ (데드레터 큐) 에 저장했다가 나중에 다시 전송할 수 있습니다.
INTERNAL	UnexpectedFailure	예상치 못한 오류가 발생했습니다. Amazon에 문의하십시오. 실패 문구 [내부 오류].	요청을 처리하는 중 FCM 서버에서 오류가 발생했습니다. Amazon SNS에서 실행한 모든 재시도가 실패했습니다. 이러한 메시지를 DLQ (데드레터 큐) 에 저장했다가 나중에 다시 전송할 수 있습니다.
THIRD_PARTY_AUTH_ERROR	InvalidCredentials	플랫폼 애플리케이션 자격 증명이 유효하지 않습니다.	iOS 장치 또는 Webpush 장치를 대상으로 하는 메시지를 보낼 수 없습니다. 개발 및 프로덕션 자격 증명이 유효한지 확인하십시오.

FCM 오류/상태 코드	Amazon SNS 장애 유형	오류 메시지	원인 및 완화
QUOTA_EXCEEDED	Throttled	[gcm] 에 의해 요청이 제한되었습니다.	메시지 속도 할당량, 기기 메시지 속도 할당량 또는 주제 메시지 속도 할당량을 초과했습니다. 이 문제를 해결하는 방법에 대한 자세한 ErrorCode내용은 Google Firebase 설명서를 참조하십시오.
PERMISSION_DENIED	InvalidNotification	알림 본문이 유효하지 않습니다.	PERMISSION_DENIED 예외의 경우 호출자 (FCM 애플리케이션) 는 페이로드에서 지정된 작업을 실행할 권한이 없습니다. FCM 콘솔로 이동하여 자격 증명에 필수 API 작업이 활성화되어 있는지 확인하세요.

모바일 앱 속성

Amazon Simple Notification Service(Amazon SNS)는 푸시 알림 메시지 전송 상태를 로그하기 위한 지원을 제공합니다. 애플리케이션 속성을 구성하면 Amazon SNS에서 모바일 엔드포인트로 전송된 메시지에 대한 로그 항목이 CloudWatch Logs로 전송됩니다. 메시지 전송 상태를 로깅하면 다음과 같이 더욱 확장된 운영 이해를 제공할 수 있습니다.

- 푸시 알림 메시지가 Amazon SNS로부터 푸시 알림 서비스로 전송되었는지 확인합니다.
- 푸시 알림 서비스로부터 Amazon SNS로 전송되는 응답을 식별합니다.
- 메시지 유지 시간(게시 타임스탬프 시간부터 푸시 알림 서비스에 넘겨주기 직전까지의 시간)을 결정합니다.

메시지 전송 상태에 대한 애플리케이션 속성을 구성하려면 AWS Management Console, AWS 소프트웨어 개발 키트(SDK) 또는 쿼리 API를 사용하면 됩니다.

주제

AWS Management Console을 사용한 메시지 전송 상태 속성 구성

1. Amazon SNS 콘솔에 로그인합니다.
2. 탐색 창에서 모바일을 가리킨 다음 푸시 알림을 선택합니다.
3. 플랫폼 애플리케이션 섹션에서 CloudWatch Logs를 수신하려는 엔드포인트가 포함된 애플리케이션을 선택합니다.
4. Application Actions(애플리케이션 작업)를 선택한 다음 Delivery Status(전송 상태)를 선택합니다.
5. Delivery Status(전송 상태) 대화 상자에서 Create IAM Roles(IAM 역할 생성)를 선택합니다.

   이제 IAM 콘솔로 리디렉션됩니다.
6. Amazon SNS에 사용자 대신 CloudWatch Logs를 사용할 수 있는 쓰기 액세스 권한을 부여하려면 허용을 선택합니다.
7. 이제 전송 상태 대화 상자로 돌아가 CloudWatch Logs를 수신하려는 성공적인 메시지의 백분율을 샘플 성공 백분율(0-100) 필드에 입력합니다.

   > **Note**
   >
   > 메시지 전송 상태를 위해 애플리케이션 속성을 구성하고 나면 메시지의 전송 실패가 CloudWatch Logs를 생성합니다.
8. 마지막으로, Save Configuration(구성 저장)을 선택합니다. 이제 CloudWatch Logs가 포함된 메시지 전송 상태를 보고 파싱할 수 있습니다. CloudWatch 사용에 대한 자세한 정보는 CloudWatch 설명서를 참조하세요.

Amazon SNS 메시지 전송 상태 CloudWatch 로그 예제

애플리케이션 엔드포인트를 위해 메시지 전송 상태 속성을 구성하고 나면 CloudWatch Logs가 생성됩니다. JSON 형식의 예제 로그는 다음과 같이 나타납니다.

성공


```
{
  "status": "SUCCESS",
  "notification": {
    "timestamp": "2015-01-26 23:07:39.54",
    "messageId": "9655abe4-6ed6-5734-89f7-e6a6a42de02a"
  },
  "delivery": {
    "statusCode": 200,
    "dwellTimeMs": 65,
    "token": "Examplei7fFachkJ1xjlqT64RaBkcGHochmf1VQAr9k-
IBJtKjp7fedYPzEwT_Pq3Tu0lroqro1cwWJUvgkcPPYcaXCpPWmG3Bqn-
wiqIEzp5zZ7y_jsM0PKPxKhddCzx6paEsyay9Zn3D4wNUJb8m6HXrBf9dqaEw",
    "attempts": 1,
    "providerResponse": "{\"multicast_id\":5138139752481671853,\"success
\":1,\"failure\":0,\"canonical_ids\":0,\"results\":[{\"message_id\":
\"0:1422313659698010%d6ba8edff9fd7ecd\"}]}",
    "destination": "arn:aws:sns:us-east-2:111122223333:endpoint/FCM/FCMPushApp/
c23e42de-3699-3639-84dd-65f84474629d"
  }
}
```


실패


```
{
  "status": "FAILURE",
  "notification": {
    "timestamp": "2015-01-26 23:29:35.678",
    "messageId": "c3ad79b0-8996-550a-8bfa-24f05989898f"
  },
  "delivery": {
    "statusCode": 8,
    "dwellTimeMs": 1451,
    "token": "examp1e29z6j5c4df46f80189c4c83fjcgf7f6257e98542d2jt3395kj73",
    "attempts": 1,
    "providerResponse": "NotificationErrorResponse(command=8, status=InvalidToken,
 id=1, cause=null)",
```

```
    "destination": "arn:aws:sns:us-east-2:111122223333:endpoint/APNS_SANDBOX/
APNSPushApp/986cb8a1-4f6b-34b1-9a1b-d9e9cb553944"
  }
}
```

푸시 알림 서비스 응답 코드를 확인하려면 플랫폼 응답 코드에서 확인하세요.

AWS SDK를 사용한 메시지 전송 상태 속성 구성

AWS SDK는 Amazon SNS에서 메시지 전송 상태 속성을 사용하기 위해 여러 언어로 API를 제공합니다.

다음 Java 예제에서는 `SetPlatformApplicationAttributes` API를 이용해 푸시 알림 메시지 전송 상태를 위한 애플리케이션 속성을 구성하는 방법을 보여줍니다. 메시지 전송 상태를 위해 `SuccessFeedbackRoleArn`, `FailureFeedbackRoleArn`, `SuccessFeedbackSampleRate` 속성을 사용할 수 있습니다. `SuccessFeedbackRoleArn` 및 `FailureFeedbackRoleArn` 속성은 사용자 대신 CloudWatch Logs를 사용할 수 있는 쓰기 액세스 권한을 Amazon SNS에 부여하는 데 사용됩니다. `SuccessFeedbackSampleRate` 속성은 성공적으로 전송된 메시지의 샘플 비율(0-100)을 지정할 때 사용됩니다. `FailureFeedbackRoleArn` 속성을 구성하고 나면, 메시지의 전송 실패가 CloudWatch Logs를 생성합니다.

```
SetPlatformApplicationAttributesRequest setPlatformApplicationAttributesRequest = new
 SetPlatformApplicationAttributesRequest();
Map<String, String> attributes = new HashMap<>();
attributes.put("SuccessFeedbackRoleArn", "arn:aws:iam::111122223333:role/SNS_CWlogs");
attributes.put("FailureFeedbackRoleArn", "arn:aws:iam::111122223333:role/SNS_CWlogs");
attributes.put("SuccessFeedbackSampleRate", "5");
setPlatformApplicationAttributesRequest.withAttributes(attributes);
setPlatformApplicationAttributesRequest.setPlatformApplicationArn("arn:aws:sns:us-
west-2:111122223333:app/FCM/FCMPushApp");
sns.setPlatformApplicationAttributes(setPlatformApplicationAttributesRequest);
```

SDK for Java에 대한 자세한 정보는 AWS SDK for Java 시작하기를 참조하세요.

플랫폼 응답 코드

다음은 푸시 알림 서비스 응답 코드를 위한 링크 목록입니다.

푸시 알림 서비스	응답 코드
Amazon Device Messaging(ADM)	ADM 설명서의 응답 형식을 참조하세요.

푸시 알림 서비스	응답 코드
Apple 푸시 알림 서비스(APN)	로컬 및 원격 알림 프로그래밍 가이드의 APN과 통신에서 APN의 HTTP/2 응답을 참조하세요.
Firebase Cloud Messaging(FCM)	Firebase Cloud Messaging 설명서의 Downstream Message Error Response Codes를 참조하세요.
Windows Phone용 Microsoft 푸시 알림 서비스(MPNS)	Windows 8 개발 설명서의 Push Notification Service Response Codes for Windows Phone 8을 참조하세요.
Windows 푸시 알림 서비스(WNS)	Windows 8 개발 설명서의 Push Notification Service Request and Response Headers(Windows Runtime Apps)에서 "응답 코드"를 참조하세요.

모바일 앱 이벤트

Amazon SNS는 특정 애플리케이션 이벤트가 발생할 때 알림을 트리거하도록 지원합니다. 그런 다음에 해당 이벤트에 대해 프로그래밍 방식의 작업을 수행할 수 있습니다. 애플리케이션에는 Apple 푸시 알림 서비스(APN), Firebase Cloud Messaging(FCM) 및 Windows 푸시 알림 서비스(WNS)와 같은 푸시 알림 서비스에 대한 지원이 포함되어야 합니다. Amazon SNS 콘솔 또는 AWS SDK를 사용하여 애플리케이션 이벤트 알림을 설정합니다. AWS CLI

주제

- 사용 가능한 애플리케이션 이벤트
- 모바일 푸시 알림 전송

사용 가능한 애플리케이션 이벤트

애플리케이션 이벤트 알림은 개별 플랫폼 엔드포인트가 전송 실패뿐 아니라 생성, 삭제 및 업데이트되는 시기도 추적합니다. 다음은 애플리케이션 이벤트의 속성 이름입니다.

속성 이름	알림 트리거
`EventEndpointCreated`	새 플랫폼 엔드포인트가 애플리케이션에 추가됩니다.
`EventEndpointDeleted`	애플리케이션과 연결된 플랫폼 엔드포인트가 삭제됩니다.
`EventEndpointUpdated`	애플리케이션과 연결된 플랫폼 엔드포인트의 속성이 변경됩니다.
`EventDeliveryFailure`	애플리케이션과 연결된 플랫폼 엔드포인트로 전송할 때 영구 실패가 발생합니다. **Note** 플랫폼 애플리케이션 쪽에서 전송 실패를 추적하려면 애플리케이션의 메시지 전송 상태 이벤트를 구독하세요. 자세한 정보는 메시지 전송 상태를 위한 Amazon SNS 애플리케이션 속성 사용을 참조하세요.

이러한 이벤트 알림을 받을 수 있는 애플리케이션에 속성을 연결할 수 있습니다.

모바일 푸시 알림 전송

애플리케이션 이벤트 알림을 전송하려면 각 유형의 이벤트에 대한 알림을 수신할 주제를 지정합니다. Amazon SNS가 알림을 전송할 때 주제는 프로그래밍 방식의 작업을 수행하는 엔드포인트로 알림을 라우팅할 수 있습니다.

Important

대용량 애플리케이션은 많은 수(예: 수만 개)의 애플리케이션 이벤트 알림을 생성합니다. 이렇게 많은 알림은 이메일 주소, 전화번호 및 모바일 애플리케이션과 같이 인간이 사용하도록 설계된 엔드포인트에 부담이 될 수 있습니다. 애플리케이션 이벤트 알림을 주제에 전송할 때 다음 지침을 고려하세요.

- 알림을 수신하는 각 주제에는 HTTP 또는 HTTPS 엔드포인트, Amazon SQS 대기열 또는 함수와 같은 프로그래밍 방식 엔드포인트에 대한 구독만 포함되어야 합니다. AWS Lambda

- 알림을 통해 트리거되는 처리량을 줄이려면 각 주제의 구독을 작은 수(예: 5개 이하)로 제한합니다.

Amazon SNS 콘솔, AWS Command Line Interface (AWS CLI) 또는 AWS SDK를 사용하여 애플리케이션 이벤트 알림을 보낼 수 있습니다.

AWS Management Console

1. Amazon SNS 콘솔에 로그인합니다.
2. 탐색 창에서 모바일(Mobile), 푸시 알림(Push notifications)을 선택합니다.
3. 모바일 푸시 알림 페이지의 플랫폼 애플리케이션 섹션에서 애플리케이션을 선택한 다음 편집을 선택합니다.
4. Event notifications(이벤트 알림) 섹션을 확장합니다.
5. 작업, 이벤트 구성을 선택합니다.
6. 다음 이벤트에 사용할 주제의 ARN을 입력합니다.
   - 엔드포인트 생성 완료
   - 엔드포인트 삭제 완료
   - 엔드포인트 업데이트 완료
   - 전송 실패
7. 변경 사항 저장를 선택합니다.

AWS CLI

set-platform-application-attributes 명령을 실행합니다.

다음 예는 4가지 애플리케이션 이벤트 모두에 대해 동일한 Amazon SNS 주제를 설정합니다.

```
aws sns set-platform-application-attributes
--platform-application-arn arn:aws:sns:us-east-1:12345EXAMPLE:app/FCM/
MyFCMPlatformApplication
--attributes EventEndpointCreated="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents",
EventEndpointDeleted="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents",
EventEndpointUpdated="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents",
```

```
EventDeliveryFailure="arn:aws:sns:us-
east-1:12345EXAMPLE:MyFCMPlatformApplicationEvents"
```

AWS SDK

AWS SDK를 사용하여 Amazon SNS API로 SetPlatformApplicationAttributes 요청을 제출하여 애플리케이션 이벤트 알림을 설정합니다.

시작 도움말 및 이전 버전에 대한 정보를 포함한 AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은을 참조하십시오. AWS SDK와 함께 Amazon SNS 사용

모바일 푸시 API 작업

Amazon SNS 모바일 푸시 API를 사용하려면 먼저 Apple 푸시 알림 서비스(APN) 및 Firebase Cloud Messaging(FCM)과 같은 푸시 알림 서비스의 사전 조건을 충족해야 합니다. 필수 조건에 대한 자세한 정보는 Amazon SNS 사용자 알림에 대한 사전 조건에서 확인하세요.

API를 사용하여 모바일 앱 및 디바이스에 푸시 알림 메시지를 보내려면 먼저 CreatePlatformApplication 작업을 사용해야 합니다. 이 작업은 PlatformApplicationArn 속성을 반환합니다. 그런 다음 PlatformApplicationArn에서 CreatePlatformEndpoint 속성을 사용하며, 그 결과 EndpointArn 속성을 반환합니다. 이제 EndpointArn 속성을 Publish 작업과 함께 사용하면서 모바일 앱과 디바이스에 알림 메시지를 보낼 수 있습니다. 또는 EndpointArn 속성을 Subscribe 작업과 함께 사용하여 주제를 구독할 수 있습니다. 자세한 정보는 사용자 알림 프로세스 개요에서 확인하세요.

Amazon SNS 모바일 푸시 API는 다음과 같습니다.

CreatePlatformApplication

디바이스 및 모바일 앱이 등록될 수 있는 지원되는 푸시 알림 서비스(예: APN 및 FCM) 중 하나를 위해 플랫폼 애플리케이션 객체를 생성합니다. PlatformApplicationArn 속성을 반환합니다. 이 속성은 CreatePlatformEndpoint 작업에서 사용합니다.

CreatePlatformEndpoint

지원되는 푸시 알림 서비스 중 하나에 디바이스 및 모바일 앱을 위한 엔드포인트를 생성합니다. CreatePlatformEndpoint에서는 PlatformApplicationArn 작업에서 반환한 CreatePlatformApplication 속성을 사용합니다. 그런 다음 EndpointArn를 사용하여 반환된 CreatePlatformEndpoint 속성을 Publish 작업과 함께 사용하면서 모바일 앱과 디바이스에 알림 메시지를 보냅니다.

CreateTopic

메시지를 게시할 수 있는 주제를 생성합니다.

DeleteEndpoint

지원되는 푸시 알림 서비스 중 하나에서 디바이스 및 모바일 앱을 위한 엔드포인트를 삭제합니다.

DeletePlatformApplication

플랫폼 애플리케이션 객체를 삭제합니다.

DeleteTopic

주제 및 해당 주제의 모든 구독을 삭제합니다.

GetEndpointAttributes

디바이스 및 모바일 앱을 위한 엔드포인트 속성을 검색합니다.

GetPlatformApplicationAttributes

플랫폼 애플리케이션 객체의 속성을 검색합니다.

ListEndpointsByPlatformApplication

지원되는 푸시 알림 서비스의 디바이스 및 모바일 앱을 위한 엔드포인트 및 엔드포인트 속성을 나열합니다.

ListPlatformApplications

지원되는 푸시 알림 서비스에 대해 플랫폼 애플리케이션 객체를 나열합니다.

Publish

주제를 구독하는 모든 엔드포인트에 알림 메시지를 전송합니다.

SetEndpointAttributes

디바이스 및 모바일 앱을 위한 엔드포인트의 속성을 설정합니다.

SetPlatformApplicationAttributes

플랫폼 애플리케이션 객체의 속성을 설정합니다.

Subscribe

엔드포인트에 확인 메시지를 전송하여 엔드포인트에서 구독할 수 있도록 준비합니다. 구독을 실제로 생성하려면 엔드포인트 오너가 확인 메시지에서 토큰을 사용하여 ConfirmSubscription 작업을 호출해야 합니다.

`Unsubscribe`

구독을 삭제합니다.

모바일 푸시 API 오류

다음 표에는 모바일 푸시에 대해 Amazon SNS API에서 반환하는 오류가 나와 있습니다. 모바일 푸시용 Amazon SNS API에 대한 자세한 정보는 모바일 푸시 API 작업에서 확인하세요.

오류	설명	HTTPS 상태 코드	API 작업
Application Name is null string	필수 애플리케이션 이름이 null로 설정되었습니다.	400	`CreatePlatformApplication`
Platform Name is null string	필수 플랫폼 이름이 null로 설정되었습니다.	400	`CreatePlatformApplication`
Platform Name is invalid	플랫폼 이름으로 잘못된 값 또는 범위를 벗어나는 값이 제공되었습니다.	400	`CreatePlatformApplication`
APNs — Principal is not a valid certificate	APN 보안 주체, 즉 "SSL 인증서"에 대해 잘못된 인증서가 제공되었습니다. 자세한 정보는 Amazon Simple Notification Service API 참조의 CreatePlatformApplication을 참조하세요.	400	`CreatePlatformApplication`
APNs — Principal is a valid cert but not in a .pem format	APN 보안 주체, 즉 "SSL 인증서"에 대해 유효하지만 .pem 형식	400	`CreatePlatformApplication`

오류	설명	HTTPS 상태 코드	API 작업
	이 아닌 인증서가 제공되었습니다.		
APNs — Principal is an expired certificate	APN 보안 주체, 즉 "SSL 인증서"에 대해 만료된 인증서가 제공되었습니다.	400	`CreatePlatformApplication`
APNs — Principal is not an Apple issued certificate	APN 보안 주체, 즉 "SSL 인증서"에 대해 Apple에서 발행하지 않은 인증서가 제공되었습니다.	400	`CreatePlatformApplication`
APNs — Principal is not provided	APN 보안 주체, 즉 "SSL 인증서"가 제공되지 않았습니다.	400	`CreatePlatformApplication`
APNs — Credential is not provided	APN 자격 증명, 즉 "개인 키"가 제공되지 않았습니다. 자세한 정보는 Amazon Simple Notification Service API 참조의 CreatePlatformApplication을 참조하세요.	400	`CreatePlatformApplication`
APNs — Credential are not in a valid .pem format	APN 자격 증명, 즉 "개인 키"가 유효한 .pem 형식이 아닙니다.	400	`CreatePlatformApplication`

오류	설명	HTTPS 상태 코드	API 작업
FCM — serverAPIKey is not provided	FCM 자격 증명, 즉 “API 키”가 제공되지 않았습니다. 자세한 정보는 Amazon Simple Notification Service API 참조의 CreatePlatformApplication을 참조하세요.	400	`CreatePlatformApplication`
FCM — serverAPIKey is empty	FCM 자격 증명, 즉 “API 키”가 비어 있습니다.	400	`CreatePlatformApplication`
FCM — serverAPIKey is a null string	FCM 자격 증명, 즉 “API 키”가 null입니다.	400	`CreatePlatformApplication`
FCM — serverAPIKey is invalid	FCM 자격 증명, 즉 “API 키”가 잘못되었습니다.	400	`CreatePlatformApplication`
ADM — clientsecret is not provided	필수 클라이언트 비밀번호가 제공되지 않았습니다.	400	`CreatePlatformApplication`
ADM — clientsecret is a null string	클라이언트 비밀번호에 대한 필수 문자열이 null입니다.	400	`CreatePlatformApplication`
ADM — client_secret is empty string	클라이언트 비밀번호에 대한 필수 문자열이 비어 있습니다.	400	`CreatePlatformApplication`
ADM — client_secret is not valid	클라이언트 비밀번호에 대한 필수 문자열이 유효하지 않습니다.	400	`CreatePlatformApplication`

오류	설명	HTTPS 상태 코드	API 작업
ADM — client_id is empty string	클라이언트 ID에 대한 필수 문자열이 비어 있습니다.	400	CreatePlatformApplication
ADM — clientId is not provided	클라이언트 ID에 대한 필수 문자열이 제공되지 않았습니다.	400	CreatePlatformApplication
ADM — clientid is a null string	클라이언트 ID에 대한 필수 문자열이 null입니다.	400	CreatePlatformApplication
ADM — client_id is not valid	클라이언트 ID에 대한 필수 문자열이 유효하지 않습니다.	400	CreatePlatformApplication
EventEndpointCreated has invalid ARN format	EventEndpointCreated에 유효하지 않은 ARN 형식이 있습니다.	400	CreatePlatformApplication
EventEndpointDeleted has invalid ARN format	EventEndpointDeleted에 유효하지 않은 ARN 형식이 있습니다.	400	CreatePlatformApplication
EventEndpointUpdated has invalid ARN format	EventEndpointUpdated에 유효하지 않은 ARN 형식이 있습니다.	400	CreatePlatformApplication
EventDeliveryAttemptFailure has invalid ARN format	EventDeliveryAttemptFailure에 유효하지 않은 ARN 형식이 있습니다.	400	CreatePlatformApplication
EventDeliveryFailure has invalid ARN format	EventDeliveryFailure에 유효하지 않은 ARN 형식이 있습니다.	400	CreatePlatformApplication

오류	설명	HTTPS 상태 코드	API 작업
EventEndpointCreated is not an existing Topic	EventEndpointCreated가 기존 주제가 아닙니다.	400	CreatePlatformApplication
EventEndpointDeleted is not an existing Topic	EventEndpointDeleted가 기존 주제가 아닙니다.	400	CreatePlatformApplication
EventEndpointUpdated is not an existing Topic	EventEndpointUpdated가 기존 주제가 아닙니다.	400	CreatePlatformApplication
EventDeliveryAttemptFailure is not an existing Topic	EventDeliveryAttemptFailure가 기존 주제가 아닙니다.	400	CreatePlatformApplication
EventDeliveryFailure is not an existing Topic	EventDeliveryFailure가 기존 주제가 아닙니다.	400	CreatePlatformApplication
Platform ARN is invalid	플랫폼 ARN이 유효하지 않습니다.	400	SetPlatformAttributes
Platform ARN is valid but does not belong to the user	플랫폼 ARN이 유효하지만 해당 사용자에게 속한 것이 아닙니다.	400	SetPlatformAttributes
APNs — Principal is not a valid certificate	APN 보안 주체, 즉 "SSL 인증서"에 대해 잘못된 인증서가 제공되었습니다. 자세한 정보는 Amazon Simple Notification Service API 참조의 CreatePlatformApplication을 참조하세요.	400	SetPlatformAttributes

오류	설명	HTTPS 상태 코드	API 작업
APNs — Principal is a valid cert but not in a .pem format	APN 보안 주체, 즉 "SSL 인증서"에 대해 유효하지만 .pem 형식이 아닌 인증서가 제공되었습니다.	400	SetPlatformAttributes
APNs — Principal is an expired certificate	APN 보안 주체, 즉 "SSL 인증서"에 대해 만료된 인증서가 제공되었습니다.	400	SetPlatformAttributes
APNs — Principal is not an Apple issued certificate	APN 보안 주체, 즉 "SSL 인증서"에 대해 Apple에서 발행하지 않은 인증서가 제공되었습니다.	400	SetPlatformAttributes
APNs — Principal is not provided	APN 보안 주체, 즉 "SSL 인증서"가 제공되지 않았습니다.	400	SetPlatformAttributes
APNs — Credential is not provided	APN 자격 증명, 즉 "개인 키"가 제공되지 않았습니다. 자세한 정보는 Amazon Simple Notification Service API 참조의 CreatePlatformApplication을 참조하세요.	400	SetPlatformAttributes
APNs — Credential are not in a valid .pem format	APN 자격 증명, 즉 "개인 키"가 유효한 .pem 형식이 아닙니다.	400	SetPlatformAttributes

오류	설명	HTTPS 상태 코드	API 작업
FCM — serverAPIKey is not provided	FCM 자격 증명, 즉 “API 키”가 제공되지 않았습니다. 자세한 정보는 Amazon Simple Notification Service API 참조의 CreatePlatformApplication을 참조하세요.	400	SetPlatformAttributes
FCM — serverAPIKey is a null string	FCM 자격 증명, 즉 “API 키”가 null입니다.	400	SetPlatformAttributes
ADM — clientId is not provided	클라이언트 ID에 대한 필수 문자열이 제공되지 않았습니다.	400	SetPlatformAttributes
ADM — clientid is a null string	클라이언트 ID에 대한 필수 문자열이 null입니다.	400	SetPlatformAttributes
ADM — clientsecret is not provided	필수 클라이언트 비밀번호가 제공되지 않았습니다.	400	SetPlatformAttributes
ADM — clientsecret is a null string	클라이언트 비밀번호에 대한 필수 문자열이 null입니다.	400	SetPlatformAttributes
EventEndpointUpdated has invalid ARN format	EventEndpointUpdated에 유효하지 않은 ARN 형식이 있습니다.	400	SetPlatformAttributes
EventEndpointDeleted has invalid ARN format	EventEndpointDeleted에 유효하지 않은 ARN 형식이 있습니다.	400	SetPlatformAttributes

오류	설명	HTTPS 상태 코드	API 작업
EventEndpointUpdated has invalid ARN format	EventEndpointUpdated에 유효하지 않은 ARN 형식이 있습니다.	400	SetPlatformAttributes
EventDeliveryAttemptFailure has invalid ARN format	EventDeliveryAttemptFailure에 유효하지 않은 ARN 형식이 있습니다.	400	SetPlatformAttributes
EventDeliveryFailure has invalid ARN format	EventDeliveryFailure에 유효하지 않은 ARN 형식이 있습니다.	400	SetPlatformAttributes
EventEndpointCreated is not an existing Topic	EventEndpointCreated가 기존 주제가 아닙니다.	400	SetPlatformAttributes
EventEndpointDeleted is not an existing Topic	EventEndpointDeleted가 기존 주제가 아닙니다.	400	SetPlatformAttributes
EventEndpointUpdated is not an existing Topic	EventEndpointUpdated가 기존 주제가 아닙니다.	400	SetPlatformAttributes
EventDeliveryAttemptFailure is not an existing Topic	EventDeliveryAttemptFailure가 기존 주제가 아닙니다.	400	SetPlatformAttributes
EventDeliveryFailure is not an existing Topic	EventDeliveryFailure가 기존 주제가 아닙니다.	400	SetPlatformAttributes
Platform ARN is invalid	플랫폼 ARN이 유효하지 않습니다.	400	GetPlatformApplicationAttributes

오류	설명	HTTPS 상태 코드	API 작업
Platform ARN is valid but does not belong to the user	플랫폼 ARN이 유효하지만 해당 사용자에게 속한 것이 아닙니다.	403	GetPlatformApplicationAttributes
Token specified is invalid	지정된 토큰이 유효하지 않습니다.	400	ListPlatformApplications
Platform ARN is invalid	플랫폼 ARN이 유효하지 않습니다.	400	ListEndpointsByPlatformApplication
Platform ARN is valid but does not belong to the user	플랫폼 ARN이 유효하지만 해당 사용자에게 속한 것이 아닙니다.	404	ListEndpointsByPlatformApplication
Token specified is invalid	지정된 토큰이 유효하지 않습니다.	400	ListEndpointsByPlatformApplication
Platform ARN is invalid	플랫폼 ARN이 유효하지 않습니다.	400	DeletePlatformApplication
Platform ARN is valid but does not belong to the user	플랫폼 ARN이 유효하지만 해당 사용자에게 속한 것이 아닙니다.	403	DeletePlatformApplication
Platform ARN is invalid	플랫폼 ARN이 유효하지 않습니다.	400	CreatePlatformEndpoint

오류	설명	HTTPS 상태 코드	API 작업
Platform ARN is valid but does not belong to the user	플랫폼 ARN이 유효하지만 해당 사용자에게 속한 것이 아닙니다.	404	CreatePlatformEndpoint
Token is not specified	토큰이 지정되지 않았습니다.	400	CreatePlatformEndpoint
Token is not of correct length	토큰이 올바른 길이가 아닙니다.	400	CreatePlatformEndpoint
Customer User data is too large	고객 사용자 데이터는 UTF-8 인코딩 형식에서 2048바이트를 넘을 수 없습니다.	400	CreatePlatformEndpoint
Endpoint ARN is invalid	엔드포인트 ARN이 유효하지 않습니다.	400	DeleteEndpoint
Endpoint ARN is valid but does not belong to the user	엔드포인트 ARN이 유효하지만 해당 사용자에게 속한 것이 아닙니다.	403	DeleteEndpoint
Endpoint ARN is invalid	엔드포인트 ARN이 유효하지 않습니다.	400	SetEndpointAttributes
Endpoint ARN is valid but does not belong to the user	엔드포인트 ARN이 유효하지만 해당 사용자에게 속한 것이 아닙니다.	403	SetEndpointAttributes
Token is not specified	토큰이 지정되지 않았습니다.	400	SetEndpointAttributes
Token is not of correct length	토큰이 올바른 길이가 아닙니다.	400	SetEndpointAttributes

오류	설명	HTTPS 상태 코드	API 작업
Customer User data is too large	고객 사용자 데이터는 UTF-8 인코딩 형식에서 2048바이트를 넘을 수 없습니다.	400	SetEndpointAttributes
Endpoint ARN is invalid	엔드포인트 ARN이 유효하지 않습니다.	400	GetEndpointAttributes
Endpoint ARN is valid but does not belong to the user	엔드포인트 ARN이 유효하지만 해당 사용자에게 속한 것이 아닙니다.	403	GetEndpointAttributes
Target ARN is invalid	대상 ARN이 유효하지 않습니다.	400	Publish
Target ARN is valid but does not belong to the user	대상 ARN이 유효하지만 해당 사용자에게 속한 것이 아닙니다.	403	Publish
Message format is invalid	메시지 형식이 잘못되었습니다.	400	Publish
Message size is larger than supported by protocol/end-service	메시지 크기가 protocol/end-service에서 지원하는 크기보다 큽니다.	400	Publish

모바일 푸시 알림에 Amazon SNS 유지 시간(TTL) 메시지 속성 사용

Amazon Simple Notification Service(Amazon SNS)는 모바일 푸시 알림 메시지에 대한 유지 시간(TTL) 메시지 속성 설정을 지원합니다. 이는 Android로 전송할 때 Amazon 디바이스 메시징 (ADM) 및 Firebase 클라우드 메시징 (FCM) 과 같이 이를 지원하는 모바일 푸시 알림 서비스를 위해 Amazon SNS 메시지 본문 내에서 TTL을 설정하는 기존 기능에 추가됩니다.

TTL 메시지 속성은 메시지에 대한 만료 메타 데이터를 지정하는 데 사용됩니다. 이를 통해 Apple 푸시 알림 서비스(APN) 또는 FCM과 같은 푸시 알림 서비스가 메시지를 엔드포인트에 전달해야 하는 시간을 지정할 수 있습니다. 모바일 디바이스가 꺼져 있는 등의 이유로 인해 지정한 TTL 내에 메시지를 전송할 수 없으면 메시지가 삭제되고 더 이상 전송 시도가 이루어지지 않습니다. 메시지 속성 내에 TTL을 지정하려면 AWS Management Console, AWS 소프트웨어 개발 키트 (SDK) 또는 쿼리 API를 사용할 수 있습니다.

주제

- 푸시 알림 서비스에 대한 TTL 메시지 속성
- TTL 정의를 위한 우선 순위
- 를 사용하여 TTL을 지정합니다. AWS Management Console

푸시 알림 서비스에 대한 TTL 메시지 속성

다음은 AWS SDK 또는 쿼리 API를 사용할 때 설정하는 데 사용할 수 있는 푸시 알림 서비스의 TTL 메시지 속성 목록입니다.

푸시 알림 서비스	TTL 메시지 속성
Amazon Device Messaging(ADM)	`AWS.SNS.MOBILE.ADM.TTL`
Apple 푸시 알림 서비스(APN)	`AWS.SNS.MOBILE.APNS.TTL`
Apple 푸시 알림 서비스 샌드박스(APNS_SANDBOX)	`AWS.SNS.MOBILE.APNS_SANDBOX.TTL`
Baidu 클라우드 푸시(Baidu)	`AWS.SNS.MOBILE.BAIDU.TTL`
Firebase 클라우드 메시징 (Android로 전송하는 경우 FCM)	`AWS.SNS.MOBILE.FCM.TTL`
Windows 푸시 알림 서비스(WNS)	`AWS.SNS.MOBILE.WNS.TTL`

각 푸시 알림 서비스는 TTL을 다르게 처리합니다. Amazon SNS는 모든 푸시 알림 서비스에 대한 TTL의 추상 보기를 제공하므로 TTL을 더 쉽게 지정할 수 있습니다. 를 사용하여 TTL (초) 을 지정하는 경우 TTL 값을 한 번만 입력하면 됩니다. 그러면 Amazon SNS가 메시지를 게시할 때 선택한 각 푸시 알림 서비스에 대한 TTL을 계산합니다. AWS Management Console

TTL은 게시 시간과 상관관계가 있습니다. 푸시 알림 메시지를 특정 푸시 알림 서비스에 넘겨주기 전에 Amazon SNS는 푸시 알림에 대한 유지 시간(게시 타임스탬프 시간부터 푸시 알림 서비스에 넘겨주기 직전까지의 시간)을 계산하여 남은 TTL을 특정 푸시 알림 서비스에 전달합니다. TTL이 유지 시간보다 짧을 경우 Amazon SNS는 게시를 시도하지 않습니다.

푸시 알림 메시지에 TTL을 지정하는 경우 TTL 값은 양의 정수여야 합니다. 단, 의 `0` 값이 푸시 알림 서비스 (예: APN 및 FCM (Android로 보내는 경우) 에 대한 특정 의미를 갖는 경우를 제외하면 TTL 값은 양의 정수여야 합니다. TTL 값이 `0`으로 설정된 경우 해당 푸시 알림 서비스에서 `0`이 특별한 의미가 없으면 Amazon SNS는 메시지를 삭제합니다. APN을 사용할 때 `0`으로 설정된 TTL 파라미터에 대한 자세한 정보는 이진 공급자 API 설명서의 표 A-3 원격 알림에 대한 항목 식별자를 참조하세요.

TTL 정의를 위한 우선 순위

Amazon SNS가 푸시 알림 메시지의 TTL을 정의하기 위해 사용하는 우선순위는 다음 순서를 기반으로 하며, 여기서 번호가 낮을수록 우선순위가 높은 것입니다.

1. 메시지 속성 TTL
2. 메시지 본문 TTL
3. 푸시 알림 서비스 기본 TTL(서비스별로 다름)
4. Amazon SNS 기본 TTL(4주)

동일한 메시지에 대해 다른 TTL 값을 설정하면(메시지 속성과 메시지 본문에 대해 서로 다른 값) Amazon SNS가 메시지 본문의 TTL을 수정하여 메시지 속성에 지정된 TTL에 맞춥니다.

를 사용하여 TTL을 지정합니다. AWS Management Console

1. Amazon SNS 콘솔에 로그인합니다.
2. 탐색 창에서 모바일(Mobile), 푸시 알림(Push notifications)을 선택합니다.
3. Mobile push notifications(모바일 푸시 알림) 페이지의 플랫폼 애플리케이션 섹션에서 애플리케이션을 선택합니다.
4. ***MyApplication***페이지의 엔드포인트 섹션에서 애플리케이션 엔드포인트를 선택한 다음 메시지 게시를 선택합니다.
5. 메시지 세부 정보 섹션에서 TTL을 입력합니다. 푸시 알림 서비스가 엔드포인트에 메시지를 전송할 시간(초)입니다.
6. 메시지 게시를 선택합니다.

모바일 애플리케이션에 지원되는 리전

현재 다음 리전에서 모바일 애플리케이션을 생성할 수 있습니다.

- 미국 동부(오하이오)
- 미국 동부(버지니아 북부)
- 미국 서부(캘리포니아 북부)
- 미국 서부(오리건)
- 아프리카(케이프타운)
- 아시아 태평양(홍콩)
- 아시아 태평양(자카르타)
- 아시아 태평양(뭄바이)
- 아시아 태평양(오사카)
- 아시아 태평양(서울)
- 아시아 태평양(싱가포르)
- 아시아 태평양(시드니)
- 아시아 태평양(도쿄)
- 캐나다(중부)
- 유럽(프랑크푸르트)
- 유럽(아일랜드)
- 유럽(런던)
- 유럽(밀라노)
- 유럽(파리)
- 유럽(스톡홀름)
- 중동(바레인)
- 중동(UAE)
- 남아메리카(상파울루)
- AWS GovCloud(미국 서부)

모바일 푸시 알림 모범 사례

이 단원에서는 고객 참여를 개선하는 데 도움이 되는 모범 사례에 대해 설명합니다.

엔드포인트 관리

디바이스에서 사용자의 작업(예: 앱이 디바이스에 다시 설치됨)으로 인해 디바이스 토큰이 변경되거나 특정 iOS 버전에서 실행되는 디바이스에 영향을 미치는 인증서 업데이트로 인해 전송 문제가 발생할 수 있습니다. 앱이 시작될 때마다 APN에 등록하는 것이 Apple에서 권장하는 모범 사례입니다.

사용자가 앱을 열 때마다 디바이스 토큰이 변경되지 않으므로 멱등 `CreatePlatformEndpoint` API를 사용할 수 있습니다. 그러나 토큰 자체가 유효하지 않거나 엔드포인트가 유효하지만 비활성화된 경우 (예: 프로덕션 환경과 샌드박스 환경이 일치하지 않는 경우) 동일한 디바이스에 중복이 발생할 수 있습니다.

의사(Pseudo) 코드에 있는 것과 같은 디바이스 토큰 관리 메커니즘을 사용할 수 있습니다.

FCM v1 기기 토큰 관리 및 유지 관리에 대한 자세한 내용은 을 참조하십시오. Firebase 클라우드 메시징 (FCM) 엔드포인트 관리

전송 상태 로깅

푸시 알림 전송 상태를 모니터링하려면 Amazon SNS 플랫폼 애플리케이션에 대한 전송 상태 로깅을 사용 설정하는 것이 좋습니다. 로그에는 푸시 플랫폼 서비스에서 반환된 공급자 응답 코드가 포함되어 있으므로 전송 실패 문제를 해결하는 데 도움이 됩니다. 전송 상태 로깅 사용 설정에 대한 자세한 내용은 푸시 알림에 대한 Amazon SNS 주제 전송 로그에 액세스하려면 어떻게 해야 합니까?를 참조하세요.

이벤트 알림

이벤트 기반 방식으로 엔드포인트를 관리하기 위해 이벤트 알림 기능을 사용할 수 있습니다. 이렇게 하면 구성된 Amazon SNS 주제가 엔드포인트 생성, 삭제, 업데이트 및 전송 실패와 같은 플랫폼 애플리케이션 이벤트에 대해 Lambda 함수와 같은 구독자에게 이벤트를 팬아웃할 수 있습니다.

이메일 알림

이 페이지에서는 AWS Management Console AWS SDK for Java, 또는 를 사용하여 Amazon SNS 주제에 이메일 주소를 구독하는 방법을 설명합니다 AWS SDK for .NET.

> **참고**
>
> - 이메일 메시지의 본문을 사용자 지정할 수 없습니다. 이메일 전송 기능은 마케팅 메시지가 아닌 내부 시스템 알림을 제공하기 위한 것입니다.

- 이메일 엔드포인트의 직접 구독은 표준 주제에만 지원됩니다.
- 이메일 전송 처리량은 Amazon SNS 할당량에 따라 제한됩니다.

Important

메일 그룹 수신자가 모든 수신자에 대해 Amazon SNS 주제 이메일의 구독을 취소하는 것을 방지하려면 AWS Support에서 인증을 해야만 취소할 수 있는 이메일 구독 설정을 참조하세요.

를 사용하여 Amazon SNS 주제에 이메일 주소를 구독하려면 AWS Management Console

1. Amazon SNS 콘솔에 로그인합니다.
2. 왼쪽의 탐색 창에서 구독을 선택합니다.
3. 구독 페이지에서 구독 생성을 선택합니다.
4. 구독 생성 페이지의 세부 정보 섹션에서 다음을 수행합니다.
   a. 주제 ARN에서 주제의 Amazon 리소스 이름(ARN)을 선택합니다.
   b. 프로토콜에서 이메일을 선택합니다.
   c. 엔드포인트에 이메일 주소를 입력합니다.
   d. (선택 사항) 필터 정책을 구성하려면 구독 필터 정책 섹션을 확장합니다. 자세한 정보는 Amazon SNS 구독 필터 정책을 참조하세요.
   e. (선택 사항) 페이로드 기반 필터링을 활성화하려면 `Filter Policy Scope`를 `MessageBody`로 구성하십시오. 자세한 정보는 Amazon SNS 구독 필터 정책 범위을 참조하세요.
   f. (선택 사항) 구독에 대한 배달 못한 편지 대기열을 구성하려면 리드라이브 정책(배달 못한 편지 대기열) 섹션을 확장합니다. 자세한 정보는 Amazon SNS 배달 못한 편지 대기열(DLQ)을 참조하세요.
   g. 구독 생성을 선택합니다.

   콘솔에서 구독을 만들고 구독의 세부 정보 페이지를 엽니다.

수신자가 구독을 확인해야만 이 이메일 주소가 메시지를 수신할 수 있습니다.

구독을 확인하려면

1. 이메일 받은 편지함을 확인하고 Amazon SNS의 이메일에서 구독 확인을 선택합니다.
2. Amazon SNS가 웹 브라우저를 열고 구독 ID와 함께 구독 확인을 표시합니다.

AWS SDK를 사용하여 Amazon SNS 주제에 이메일 주소를 구독하려면

AWS SDK를 사용하려면 사용자 인증 정보로 구성해야 합니다. 자세한 정보는 AWS SDK 및 도구 참조 가이드의 공유 구성 및 자격 증명 파일을 참조하세요.

다음 코드 예제는 사용 `Subscribe` 방법을 보여줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 내용이 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

주제에 대한 이메일 주소를 구독하세요.

```
        /// <summary>
        /// Creates a new subscription to a topic.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object, used
        /// to create an Amazon SNS subscription.</param>
        /// <param name="topicArn">The ARN of the topic to subscribe to.</param>
        /// <returns>A SubscribeResponse object which includes the subscription
        /// ARN for the new subscription.</returns>
        public static async Task<SubscribeResponse> TopicSubscribeAsync(
            IAmazonSimpleNotificationService client,
            string topicArn)
        {
            SubscribeRequest request = new SubscribeRequest()
            {
                TopicArn = topicArn,
```

```
                ReturnSubscriptionArn = true,
                Protocol = "email",
                Endpoint = "recipient@example.com",
            };

            var response = await client.SubscribeAsync(request);

            return response;
        }
```

선택적 필터를 사용하여 주제 대기열을 구독하세요.

```
    /// <summary>
    /// Subscribe a queue to a topic with optional filters.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="useFifoTopic">The optional filtering policy for the
subscription.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <returns>The ARN of the new subscription.</returns>
    public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
filterPolicy, string queueArn)
    {
        var subscribeRequest = new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "sqs",
            Endpoint = queueArn
        };

        if (!string.IsNullOrEmpty(filterPolicy))
        {
            subscribeRequest.Attributes = new Dictionary<string, string>
{ { "FilterPolicy", filterPolicy } };
        }

        var subscribeResponse = await
_amazonSNSClient.SubscribeAsync(subscribeRequest);
        return subscribeResponse.SubscriptionArn;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 Subscribe를 참조하십시오.

C++

SDK for C++

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

주제에 대한 이메일 주소를 구독하세요.

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to an email address.
/*!
  \param topicARN: An SNS topic Amazon Resource Name (ARN).
  \param emailAddress: An email address.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::subscribeEmail(const Aws::String &topicARN,
                                 const Aws::String &emailAddress,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("email");
    request.SetEndpoint(emailAddress);

    const Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
```

```
    else {
        std::cerr << "Error while subscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

모바일 애플리케이션을 구독하여 주제를 구독하십시오.

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to a mobile app.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param endpointARN: The ARN for a mobile app or device endpoint.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::subscribeApp(const Aws::String &topicARN,
                          const Aws::String &endpointARN,
                          const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("application");
    request.SetEndpoint(endpointARN);

    const Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
```

```
        std::cerr << "Error while subscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

Lambda 함수를 구독하여 주제를 등록하십시오.

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to an AWS Lambda function.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param lambdaFunctionARN: The ARN for an AWS Lambda function.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::subscribeLambda(const Aws::String &topicARN,
                                  const Aws::String &lambdaFunctionARN,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("lambda");
    request.SetEndpoint(lambdaFunctionARN);

    const Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " <<
 outcome.GetError().GetMessage()
```

```
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

SQS 대기열에서 주제를 구독하십시오.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);

            Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
 outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN <<
 "."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
```

```
                        sqsClient);

                return false;
            }
```

필터를 사용하여 주제를 구독하십시오.

```
        static const Aws::String TONE_ATTRIBUTE("tone");
        static const Aws::Vector<Aws::String> TONES = {"cheerful", "funny",
 "serious",
                                                       "sincere"};

        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);
            if (isFifoTopic) {
                if (first) {
                    std::cout << "Subscriptions to a FIFO topic can have
 filters."
                              << std::endl;
                    std::cout
                            << "If you add a filter to this subscription, then
 only the filtered messages "
                            << "will be received in the queue." << std::endl;
                    std::cout << "For information about message filtering, "
                              << "see https://docs.aws.amazon.com/sns/latest/dg/
sns-message-filtering.html"
                              << std::endl;
                    std::cout << "For this example, you can filter messages by a
 \""
                              << TONE_ATTRIBUTE << "\" attribute." << std::endl;
                }

                std::ostringstream ostringstream;
                ostringstream << "Filter messages for \"" << queueName
```

```
                            << "\"'s subscription to the topic \""
                            << topicName << "\"?  (y/n)";

                // Add filter if user answers yes.
                if (askYesNoQuestion(ostringstream.str())) {
                    Aws::String jsonPolicy = getFilterPolicyFromUser();
                    if (!jsonPolicy.empty()) {
                        filteringMessages = true;

                        std::cout << "This is the filter policy for this
subscription."
                                  << std::endl;
                        std::cout << jsonPolicy << std::endl;

                        request.AddAttributes("FilterPolicy", jsonPolicy);
                    }
                    else {
                        std::cout
                                << "Because you did not select any attributes, no
filter "
                                << "will be added to this subscription." <<
std::endl;
                    }
                }
            }  // if (isFifoTopic)
            Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN <<
"."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
```

```
                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }

//! Routine that lets the user select attributes for a subscription filter
 policy.
/*!
 \sa getFilterPolicyFromUser()
 \return Aws::String: The filter policy as JSON.
 */
Aws::String AwsDoc::TopicsAndQueues::getFilterPolicyFromUser() {
    std::cout
            << "You can filter messages by one or more of the following \""
            << TONE_ATTRIBUTE << "\" attributes." << std::endl;

    std::vector<Aws::String> filterSelections;
    int selection;
    do {
        for (size_t j = 0; j < TONES.size(); ++j) {
            std::cout << "  " << (j + 1) << ". " << TONES[j]
                      << std::endl;
        }
        selection = askQuestionForIntRange(
                "Enter a number (or enter zero to stop adding more). ",
                0, static_cast<int>(TONES.size()));

        if (selection != 0) {
            const Aws::String &selectedTone(TONES[selection - 1]);
            // Add the tone to the selection if it is not already added.
            if (std::find(filterSelections.begin(),
                          filterSelections.end(),
                          selectedTone)
                == filterSelections.end()) {
                filterSelections.push_back(selectedTone);
            }
        }
    } while (selection != 0);

    Aws::String result;
```

```
    if (!filterSelections.empty()) {
        std::ostringstream jsonPolicyStream;
        jsonPolicyStream << "{ \"" << TONE_ATTRIBUTE << "\": [";


        for (size_t j = 0; j < filterSelections.size(); ++j) {
            jsonPolicyStream << "\"" << filterSelections[j] << "\"";
            if (j < filterSelections.size() - 1) {
                jsonPolicyStream << ",";
            }
        }
        jsonPolicyStream << "] }";

        result = jsonPolicyStream.str();
    }

    return result;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 Subscribe를 참조하십시오.

CLI

AWS CLI

주제를 구독하려면

다음 subscribe 명령은 이메일 주소로 지정된 주제를 구독합니다.

```
aws sns subscribe \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --protocol email \
    --notification-endpoint my-email@example.com
```

출력:

```
{
    "SubscriptionArn": "pending confirmation"
}
```

- API 세부 정보는 AWS CLI 명령 참조의 Subscribe를 참조하세요.

Go

SDK for Go V2

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

선택적 필터를 사용하여 주제 대기열을 구독하세요.

```
// 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(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(context.TODO(), &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
```

```
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 Subscribe를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

주제에 대한 이메일 주소를 구독하세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeEmail {
    public static void main(String[] args) {
        final String usage = """
                Usage:     <topicArn> <email>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   email - The email address to use.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String email = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subEmail(snsClient, topicArn, email);
        snsClient.close();
    }

    public static void subEmail(SnsClient snsClient, String topicArn, String
 email) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("email")
                    .endpoint(email)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() +
 "\n\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

주제에 대한 HTTP 엔드포인트를 구독하십시오.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeHTTPS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn> <url>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   url - The HTTPS endpoint that you want to receive
 notifications.
                """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String url = args[1];
```

```
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subHTTPS(snsClient, topicArn, url);
        snsClient.close();
    }

    public static void subHTTPS(SnsClient snsClient, String topicArn, String url)
 {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("https")
                    .endpoint(url)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN is " + result.subscriptionArn()
 + "\n\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Lambda 함수를 구독하여 주제를 등록하십시오.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeLambda {

    public static void main(String[] args) {

        final String usage = """

                Usage:    <topicArn> <lambdaArn>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   lambdaArn - The ARN of an AWS Lambda function.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String lambdaArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnValue = subLambda(snsClient, topicArn, lambdaArn);
        System.out.println("Subscription ARN: " + arnValue);
        snsClient.close();
    }

    public static String subLambda(SnsClient snsClient, String topicArn, String
 lambdaArn) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("lambda")
                    .endpoint(lambdaArn)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();
```

```
            SubscribeResponse result = snsClient.subscribe(request);
            return result.subscriptionArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Subscribe를 참조하십시오.

JavaScript

(v3) 용 JavaScript SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

별도의 모듈에서 클라이언트를 생성하고 내보냅니다.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
```

```
 * @param {string} topicArn - The ARN of the topic for which you wish to confirm
 a subscription.
 * @param {string} emailAddress - The email address that is subscribed to the
 topic.
 */
export const subscribeEmail = async (
  topicArn = "TOPIC_ARN",
  emailAddress = "usern@me.com",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "email",
      TopicArn: topicArn,
      Endpoint: emailAddress,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
};
```

모바일 애플리케이션에서 주제를 구독하세요.

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic the subscriber is subscribing
 to.
 * @param {string} endpoint - The Endpoint ARN of an application. This endpoint
 is created
 *                            when an application registers for notifications.
 */
```

```
export const subscribeApp = async (
  topicArn = "TOPIC_ARN",
  endpoint = "ENDPOINT",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "application",
      TopicArn: topicArn,
      Endpoint: endpoint,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
  return response;
};
```

Lambda 함수를 구독하여 주제를 등록하십시오.

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic the subscriber is subscribing
 to.
 * @param {string} endpoint - The Endpoint ARN of and AWS Lambda function.
 */
export const subscribeLambda = async (
  topicArn = "TOPIC_ARN",
  endpoint = "ENDPOINT",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
```

```
      Protocol: "lambda",
      TopicArn: topicArn,
      Endpoint: endpoint,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
  return response;
};
```

SQS 대기열에서 주제를 구독하십시오.

```
import { SubscribeCommand, SNSClient } from "@aws-sdk/client-sns";

const client = new SNSClient({});

export const subscribeQueue = async (
  topicArn = "TOPIC_ARN",
  queueArn = "QUEUE_ARN",
) => {
  const command = new SubscribeCommand({
    TopicArn: topicArn,
    Protocol: "sqs",
    Endpoint: queueArn,
  });

  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '931e13d9-5e2b-543f-8781-4e9e494c5ff2',
```

```
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:subscribe-queue-
test-430895:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

필터를 사용하여 주제를 구독하십시오.

```
import { SubscribeCommand, SNSClient } from "@aws-sdk/client-sns";

const client = new SNSClient({});

export const subscribeQueueFiltered = async (
  topicArn = "TOPIC_ARN",
  queueArn = "QUEUE_ARN",
) => {
  const command = new SubscribeCommand({
    TopicArn: topicArn,
    Protocol: "sqs",
    Endpoint: queueArn,
    Attributes: {
      // This subscription will only receive messages with the 'event' attribute
 set to 'order_placed'.
      FilterPolicyScope: "MessageAttributes",
      FilterPolicy: JSON.stringify({
        event: ["order_placed"],
      }),
    },
  });

  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '931e13d9-5e2b-543f-8781-4e9e494c5ff2',
  //     extendedRequestId: undefined,
```

```
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:subscribe-queue-
test-430895:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 Subscribe를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

주제에 대한 이메일 주소를 구독하세요.

```
suspend fun subEmail(topicArnVal: String, email: String): String {

    val request = SubscribeRequest {
        protocol = "email"
        endpoint = email
        returnSubscriptionArn = true
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        return result.subscriptionArn.toString()
    }
}
```

Lambda 함수를 구독하여 주제를 등록하십시오.

```
suspend fun subLambda(topicArnVal: String?, lambdaArn: String?) {

    val request = SubscribeRequest {
        protocol = "lambda"
        endpoint = lambdaArn
        returnSubscriptionArn = true
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" The subscription Arn is ${result.subscriptionArn}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 Subscribe를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> 더 많은 정보가 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

주제에 대한 이메일 주소를 구독하세요.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation
 message.
```

```
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'email';
$endpoint = 'sample@example.com';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

주제에 대한 HTTP 엔드포인트를 구독하십시오.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation
 message.
 *
```

```
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'https';
$endpoint = 'https://';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 Subscribe를 참조하십시오.

Python

SDK for Python(Boto3)

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

주제에 대한 이메일 주소를 구독하세요.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def subscribe(topic, protocol, endpoint):
        """
        Subscribes an endpoint to the topic. Some endpoint types, such as email,
        must be confirmed before their subscriptions are active. When a
 subscription
        is not confirmed, its Amazon Resource Number (ARN) is set to
        'PendingConfirmation'.

        :param topic: The topic to subscribe to.
        :param protocol: The protocol of the endpoint, such as 'sms' or 'email'.
        :param endpoint: The endpoint that receives messages, such as a phone
 number
                         (in E.164 format) for SMS messages, or an email address
 for
                         email messages.
        :return: The newly added subscription.
        """
        try:
            subscription = topic.subscribe(
                Protocol=protocol, Endpoint=endpoint, ReturnSubscriptionArn=True
            )
            logger.info("Subscribed %s %s to topic %s.", protocol, endpoint,
 topic.arn)
        except ClientError:
            logger.exception(
                "Couldn't subscribe %s %s to topic %s.", protocol, endpoint,
 topic.arn
            )
            raise
        else:
            return subscription
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 Subscribe를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

주제에 대한 이메일 주소를 구독하세요.

```
require "aws-sdk-sns"
require "logger"

# Represents a service for creating subscriptions in Amazon Simple Notification
 Service (SNS)
class SubscriptionService
  # Initializes the SubscriptionService with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Attempts to create a subscription to a topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param protocol [String] The subscription protocol (e.g., email)
  # @param endpoint [String] The endpoint that receives the notifications (email
 address)
  # @return [Boolean] true if subscription was successfully created, false
 otherwise
  def create_subscription(topic_arn, protocol, endpoint)
    @sns_client.subscribe(topic_arn: topic_arn, protocol: protocol, endpoint:
 endpoint)
```

```
    @logger.info("Subscription created successfully.")
    true
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error while creating the subscription: #{e.message}")
    false
  end
end

# Main execution if the script is run directly
if $PROGRAM_NAME == __FILE__
  protocol = "email"
  endpoint = "EMAIL_ADDRESS" # Should be replaced with a real email address
  topic_arn = "TOPIC_ARN"    # Should be replaced with a real topic ARN

  sns_client = Aws::SNS::Client.new
  subscription_service = SubscriptionService.new(sns_client)

  @logger.info("Creating the subscription.")
  unless subscription_service.create_subscription(topic_arn, protocol, endpoint)
    @logger.error("Subscription creation failed. Stopping program.")
    exit 1
  end
end
```

- 자세한 정보는 AWS SDK for Ruby 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for Ruby API 참조의 Subscribe를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

주제에 대한 이메일 주소를 구독하세요.

```
async fn subscribe_and_publish(
```

```
    client: &Client,
    topic_arn: &str,
    email_address: &str,
) -> Result<(), Error> {
    println!("Receiving on topic with ARN: `{}`", topic_arn);

    let rsp = client
        .subscribe()
        .topic_arn(topic_arn)
        .protocol("email")
        .endpoint(email_address)
        .send()
        .await?;

    println!("Added a subscription: {:?}", rsp);

    let rsp = client
        .publish()
        .topic_arn(topic_arn)
        .message("hello sns!")
        .send()
        .await?;

    println!("Published message: {:?}", rsp);

    Ok(())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 Subscribe을 참조하십시오.

SAP ABAP

SAP ABAP용 SDK

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

주제에 대한 이메일 주소를 구독하세요.

```
   TRY.
       oo_result = lo_sns->subscribe(                       "oo_result is
returned for testing purposes."
               iv_topicarn = iv_topic_arn
               iv_protocol = 'email'
               iv_endpoint = iv_email_address
               iv_returnsubscriptionarn = abap_true
           ).
       MESSAGE 'Email address subscribed to SNS topic.' TYPE 'I'.
     CATCH /aws1/cx_snsnotfoundexception.
       MESSAGE 'Topic does not exist.' TYPE 'E'.
     CATCH /aws1/cx_snssubscriptionlmte00.
       MESSAGE 'Unable to create subscriptions. You have reached the maximum
number of subscriptions allowed.' TYPE 'E'.
   ENDTRY.
```

- API에 대한 세부 정보는 SAP ABAP용AWS SDK API 참조의 Subscribe를 참조하세요.

AWS SDK를 사용하는 Amazon SNS의 코드 예제

다음 코드 예제는 AWS 소프트웨어 개발 키트 (SDK) 와 함께 Amazon SNS를 사용하는 방법을 보여줍니다.

작업은 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 작업은 개별 서비스 함수를 호출하는 방법을 보여 주며 관련 시나리오와 교차 서비스 예시에서 컨텍스트에 맞는 작업을 볼 수 있습니다.

시나리오는 동일한 서비스 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여주는 코드 예시입니다.

교차 서비스 예시는 여러 AWS 서비스전반에서 작동하는 샘플 애플리케이션입니다.

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 을 참조하십시오. AWS SDK와 함께 Amazon SNS 사용 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

시작하기

Hello Amazon SNS

다음 코드 예시는 Amazon SNS 사용을 시작하는 방법을 보여줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> 자세한 내용은 에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
using Amazon.SimpleNotificationService;
using Amazon.SimpleNotificationService.Model;

namespace SNSActions;

public static class HelloSNS
```

```
{
    static async Task Main(string[] args)
    {
        var snsClient = new AmazonSimpleNotificationServiceClient();

        Console.WriteLine($"Hello Amazon SNS! Following are some of your
 topics:");
        Console.WriteLine();

        // You can use await and any of the async methods to get a response.
        // Let's get a list of topics.
        var response = await snsClient.ListTopicsAsync(
            new ListTopicsRequest());

        foreach (var topic in response.Topics)
        {
            Console.WriteLine($"\tTopic ARN: {topic.TopicArn}");
            Console.WriteLine();
        }
    }
}
```

- API 세부 정보는 AWS SDK for .NET API ListTopics참조를 참조하십시오.

C++

SDK for C++

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

C MakeLists .txt CMake 파일의 코드입니다.

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS sns)
```

```
# Set this project's name.
project("hello_sns")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_sns.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

hello_sns.cpp 소스 파일의 코드입니다.

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

```
#include <aws/sns/SNSClient.h>
#include <aws/sns/model/ListTopicsRequest.h>
#include <iostream>

/*
 *  A "Hello SNS" starter application which initializes an Amazon Simple
 Notification
 *  Service (Amazon SNS) client and lists the SNS topics in the current account.
 *
 *  main function
 *
 *  Usage: 'hello_sns'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::SNS::SNSClient snsClient(clientConfig);

        Aws::Vector<Aws::SNS::Model::Topic> allTopics;
        Aws::String nextToken; // Next token is used to handle a paginated
 response.
        do {
            Aws::SNS::Model::ListTopicsRequest request;

            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }

            const Aws::SNS::Model::ListTopicsOutcome outcome =
 snsClient.ListTopics(
                    request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::SNS::Model::Topic> &paginatedTopics =
                        outcome.GetResult().GetTopics();
```

```
                if (!paginatedTopics.empty()) {
                    allTopics.insert(allTopics.cend(), paginatedTopics.cbegin(),
                                     paginatedTopics.cend());
                }
            }
            else {
                std::cerr << "Error listing topics " <<
 outcome.GetError().GetMessage()
                          << std::endl;
                return 1;
            }

            nextToken = outcome.GetResult().GetNextToken();
        } while (!nextToken.empty());

        std::cout << "Hello Amazon SNS! You have " << allTopics.size() << "
 topic"
                  << (allTopics.size() == 1 ? "" : "s") << " in your account."
                  << std::endl;

        if (!allTopics.empty()) {
            std::cout << "Here are your topic ARNs." << std::endl;
            for (const Aws::SNS::Model::Topic &topic: allTopics) {
                std::cout << "  * " << topic.GetTopicArn() << std::endl;
            }
        }
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- API에 대한 자세한 내용은 API 레퍼런스를 참조하십시오 ListTopics.AWS SDK for C++

Go

SDK for Go V2

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 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() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 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(context.TODO())
```

```
  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 ListTopics참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
package com.example.sns;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.paginators.ListTopicsIterable;

public class HelloSNS {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
```

```
        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsIterable listTopics = snsClient.listTopicsPaginator();
            listTopics.stream()
                    .flatMap(r -> r.topics().stream())
                    .forEach(content -> System.out.println(" Topic ARN: " +
 content.topicArn()));

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API ListTopics참조를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

SNS 클라이언트를 초기화하고 계정의 주제를 나열하세요.

```
import { SNSClient, paginateListTopics } from "@aws-sdk/client-sns";

export const helloSns = async () => {
  // The configuration object (`{}`) is required. If the region and credentials
  // are omitted, the SDK uses your local configuration if it exists.
  const client = new SNSClient({});
```

```
  // You can also use `ListTopicsCommand`, but to use that command you must
  // handle the pagination yourself. You can do that by sending the
 `ListTopicsCommand`
  // with the `NextToken` parameter from the previous request.
  const paginatedTopics = paginateListTopics({ client }, {});
  const topics = [];

  for await (const page of paginatedTopics) {
    if (page.Topics?.length) {
      topics.push(...page.Topics);
    }
  }

  const suffix = topics.length === 1 ? "" : "s";

  console.log(
    `Hello, Amazon SNS! You have ${topics.length} topic${suffix} in your
 account.`,
  );
  console.log(topics.map((t) => `  * ${t.TopicArn}`).join("\n"));
};
```

- API 세부 정보는 AWS SDK for JavaScript API ListTopics참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import aws.sdk.kotlin.services.sns.SnsClient
import aws.sdk.kotlin.services.sns.model.ListTopicsRequest
import aws.sdk.kotlin.services.sns.paginators.listTopicsPaginated
import kotlinx.coroutines.flow.transform

/**
Before running this Kotlin code example, set up your development environment,
```

```
including your credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
 */
suspend fun main() {
    listTopicsPag()
}

suspend fun listTopicsPag() {
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.listTopicsPaginated(ListTopicsRequest { })
            .transform { it.topics?.forEach { topic -> emit(topic) } }
            .collect { topic ->
                println("The topic ARN is ${topic.topicArn}")
            }
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 ListTopics.

코드 예시

- AWS SDK를 사용하는 Amazon SNS의 작업
  - AWS SDK 또는 CheckIfPhoneNumberIsOptedOut CLI와 함께 사용
  - AWS SDK 또는 ConfirmSubscription CLI와 함께 사용
  - AWS SDK 또는 CreateTopic CLI와 함께 사용
  - AWS SDK 또는 DeleteTopic CLI와 함께 사용
  - AWS SDK 또는 GetSMSAttributes CLI와 함께 사용
  - AWS SDK 또는 GetTopicAttributes CLI와 함께 사용
  - AWS SDK 또는 ListPhoneNumbersOptedOut CLI와 함께 사용
  - AWS SDK 또는 ListSubscriptions CLI와 함께 사용
  - AWS SDK 또는 ListTopics CLI와 함께 사용
  - AWS SDK 또는 Publish CLI와 함께 사용
  - AWS SDK 또는 SetSMSAttributes CLI와 함께 사용
  - AWS SDK 또는 SetSubscriptionAttributes CLI와 함께 사용
  - AWS SDK 또는 SetSubscriptionAttributesRedrivePolicy CLI와 함께 사용

- AWS SDK 또는 SetTopicAttributes CLI와 함께 사용
- AWS SDK 또는 Subscribe CLI와 함께 사용
- AWS SDK 또는 TagResource CLI와 함께 사용
- AWS SDK 또는 Unsubscribe CLI와 함께 사용

- AWS SDK를 사용하는 Amazon SNS의 시나리오
  - AWS SDK를 사용하여 Amazon SNS 푸시 알림을 위한 플랫폼 엔드포인트를 생성합니다.
  - SDK를 사용하여 FIFO Amazon SNS 주제를 만들고 게시하십시오. AWS
  - AWS SDK를 사용하여 Amazon SNS 주제에 SMS 메시지를 게시합니다.
  - AWS SDK를 사용하여 Amazon S3를 사용하여 Amazon SNS에 대용량 메시지를 게시합니다.
  - AWS SDK를 사용하여 Amazon SNS SMS 문자 메시지를 게시합니다.
  - SDK를 사용하여 Amazon SNS 메시지를 Amazon SQS 대기열에 게시합니다. AWS
- SDK를 사용하는 AWS Amazon SNS의 서버리스 예제
  - Amazon SNS 트리거를 사용하여 Lambda 함수 호출
- SDK를 사용하는 AWS Amazon SNS의 크로스 서비스 예제
  - DynamoDB 테이블에 데이터를 제출하기 위한 애플리케이션 구축
  - 메시지를 번역하는 게시 및 구독 애플리케이션 구축
  - 사용자가 레이블을 사용하여 사진을 관리할 수 있는 사진 자산 관리 애플리케이션 만들기
  - Amazon Textract 탐색기 애플리케이션 생성
  - Amazon Rekognition에서 SDK를 사용하여 동영상 속 사람과 물체를 감지합니다. AWS
  - SDK를 사용하여 Amazon SNS 메시지를 Amazon SQS 대기열에 게시합니다. AWS
  - API Gateway를 사용하여 Lambda 함수 호출
  - 예약된 이벤트를 사용하여 Lambda 함수 호출

AWS SDK를 사용하는 Amazon SNS의 작업

다음 코드 예제는 AWS SDK를 사용하여 개별 Amazon SNS 작업을 수행하는 방법을 보여줍니다. 이들 발췌문은 Amazon SNS API를 직접적으로 호출하며, 컨텍스트에서 실행되어야 하는 더 큰 프로그램에서 발췌한 코드입니다. 각 예제에는 코드 설정 및 실행 지침을 찾을 수 있는 링크가 포함되어 있습니다. GitHub

다음 예제에는 가장 일반적으로 사용되는 작업만 포함되어 있습니다. 전체 목록은 Amazon Simple Notification Service(SNS) API 참조를 참조하세요.

예제

- AWS SDK 또는 CheckIfPhoneNumberIsOptedOut CLI와 함께 사용
- AWS SDK 또는 ConfirmSubscription CLI와 함께 사용
- AWS SDK 또는 CreateTopic CLI와 함께 사용
- AWS SDK 또는 DeleteTopic CLI와 함께 사용
- AWS SDK 또는 GetSMSAttributes CLI와 함께 사용
- AWS SDK 또는 GetTopicAttributes CLI와 함께 사용
- AWS SDK 또는 ListPhoneNumbersOptedOut CLI와 함께 사용
- AWS SDK 또는 ListSubscriptions CLI와 함께 사용
- AWS SDK 또는 ListTopics CLI와 함께 사용
- AWS SDK 또는 Publish CLI와 함께 사용
- AWS SDK 또는 SetSMSAttributes CLI와 함께 사용
- AWS SDK 또는 SetSubscriptionAttributes CLI와 함께 사용
- AWS SDK 또는 SetSubscriptionAttributesRedrivePolicy CLI와 함께 사용
- AWS SDK 또는 SetTopicAttributes CLI와 함께 사용
- AWS SDK 또는 Subscribe CLI와 함께 사용
- AWS SDK 또는 TagResource CLI와 함께 사용
- AWS SDK 또는 Unsubscribe CLI와 함께 사용

AWS SDK 또는 **CheckIfPhoneNumberIsOptedOut** CLI와 함께 사용

다음 코드 예제는 `CheckIfPhoneNumberIsOptedOut`의 사용 방법을 보여줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
using System;
```

```
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example shows how to use the Amazon Simple Notification Service
    /// (Amazon SNS) to check whether a phone number has been opted out.
    /// </summary>
    public class IsPhoneNumOptedOut
    {
        public static async Task Main()
        {
            string phoneNumber = "+15551112222";

            IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

            await CheckIfOptedOutAsync(client, phoneNumber);
        }

        /// <summary>
        /// Checks to see if the supplied phone number has been opted out.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS Client object used
        /// to check if the phone number has been opted out.</param>
        /// <param name="phoneNumber">A string representing the phone number
        /// to check.</param>
        public static async Task
CheckIfOptedOutAsync(IAmazonSimpleNotificationService client, string
phoneNumber)
        {
            var request = new CheckIfPhoneNumberIsOptedOutRequest
            {
                PhoneNumber = phoneNumber,
            };

            try
            {
                var response = await
client.CheckIfPhoneNumberIsOptedOutAsync(request);

                if (response.HttpStatusCode == System.Net.HttpStatusCode.OK)
                {
```

```
                    string optOutStatus = response.IsOptedOut ? "opted out" :
"not opted out.";
                    Console.WriteLine($"The phone number: {phoneNumber} is
{optOutStatus}");
                }
            }
            catch (AuthorizationErrorException ex)
            {
                Console.WriteLine($"{ex.Message}");
            }
        }
    }
```

- API 세부 정보는 AWS SDK for .NET API CheckIfPhoneNumberIsOptedOut참조를 참조하십시오.

CLI

AWS CLI

전화번호의 SMS 메시지 옵트아웃을 확인하려면

다음 `check-if-phone-number-is-opted-out` 예시는 지정된 전화번호가 현재 AWS 계정으로부터 SMS 메시지 수신을 거부했는지 여부를 확인합니다.

```
aws sns check-if-phone-number-is-opted-out \
    --phone-number +1555550100
```

출력:

```
{
    "isOptedOut": false
}
```

- API 세부 정보는 AWS CLI 명령 CheckIfPhoneNumberIsOptedOut참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
 software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutRequest;
import
 software.amazon.awssdk.services.sns.model.CheckIfPhoneNumberIsOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CheckOptOut {
    public static void main(String[] args) {

        final String usage = """

                Usage:    <phoneNumber>

                Where:
                   phoneNumber - The mobile phone number to look up (for example,
 +1XXX5550100).

                """;

        if (args.length != 1) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String phoneNumber = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        checkPhone(snsClient, phoneNumber);
        snsClient.close();
    }

    public static void checkPhone(SnsClient snsClient, String phoneNumber) {
        try {
            CheckIfPhoneNumberIsOptedOutRequest request =
 CheckIfPhoneNumberIsOptedOutRequest.builder()
                    .phoneNumber(phoneNumber)
                    .build();

            CheckIfPhoneNumberIsOptedOutResponse result =
 snsClient.checkIfPhoneNumberIsOptedOut(request);
            System.out.println(
                    result.isOptedOut() + "Phone Number " + phoneNumber + " has
 Opted Out of receiving sns messages." +
                            "\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API CheckIfPhoneNumberIsOptedOut참조를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

별도의 모듈에서 클라이언트를 생성하고 내보냅니다.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
import { CheckIfPhoneNumberIsOptedOutCommand } from "@aws-sdk/client-sns";

import { snsClient } from "../libs/snsClient.js";

export const checkIfPhoneNumberIsOptedOut = async (
  phoneNumber = "5555555555",
) => {
  const command = new CheckIfPhoneNumberIsOptedOutCommand({
    phoneNumber,
  });

  const response = await snsClient.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '3341c28a-cdc8-5b39-a3ee-9fb0ee125732',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
```

```
  //     totalRetryDelay: 0
  //   },
  //   isOptedOut: false
  // }
  return response;
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API CheckIfPhoneNumberIsOptedOut참조를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Indicates whether the phone number owner has opted out of receiving SMS
 messages from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
```

```
]);

$phone = '+1XXX5550100';

try {
    $result = $SnSclient->checkIfPhoneNumberIsOptedOut([
        'phoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 자세한 정보는 AWS SDK for PHP 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for PHP API CheckIfPhoneNumberIsOptedOut참조를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오AWS SDK와 함께 Amazon SNS 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **ConfirmSubscription** CLI와 함께 사용

다음 코드 예제는 ConfirmSubscription의 사용 방법을 보여줍니다.

CLI

AWS CLI

구독을 확인하려면

다음 confirm-subscription 명령은 my-topic라는 SNS 주제를 구독할 때 시작된 확인 프로세스를 완료합니다. --token 파라미터는 구독 호출에 지정된 알림 엔드포인트로 전송된 확인 메시지에서 제공됩니다.

```
aws sns confirm-subscription \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
```

```
    --token
 2336412f37fb687f5d51e6e241d7700ae02f7124d8268910b858cb4db727ceeb2474bb937929d3bdd7ce5d0c
```

출력:

```
{
    "SubscriptionArn": "arn:aws:sns:us-west-2:123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f"
}
```

- API 세부 정보는 AWS CLI 명령 ConfirmSubscription참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ConfirmSubscriptionRequest;
import software.amazon.awssdk.services.sns.model.ConfirmSubscriptionResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ConfirmSubscription {
    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:    <subscriptionToken> <topicArn>

                Where:
                   subscriptionToken - A short-lived token sent to an endpoint
 during the Subscribe action.
                   topicArn - The ARN of the topic.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionToken = args[0];
        String topicArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        confirmSub(snsClient, subscriptionToken, topicArn);
        snsClient.close();
    }

    public static void confirmSub(SnsClient snsClient, String subscriptionToken,
 String topicArn) {
        try {
            ConfirmSubscriptionRequest request =
 ConfirmSubscriptionRequest.builder()
                    .token(subscriptionToken)
                    .topicArn(topicArn)
                    .build();

            ConfirmSubscriptionResponse result =
 snsClient.confirmSubscription(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode() + "\n\nSubscription Arn: \n\n"
                    + result.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API ConfirmSubscription참조를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

별도의 모듈에서 클라이언트를 생성하고 내보냅니다.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
import { ConfirmSubscriptionCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} token - This token is sent the subscriber. Only subscribers
 *                         that are not AWS services (HTTP/S, email) need to be
 confirmed.
 * @param {string} topicArn - The ARN of the topic for which you wish to confirm
 a subscription.
 */
export const confirmSubscription = async (
  token = "TOKEN",
  topicArn = "TOPIC_ARN",
) => {
  const response = await snsClient.send(
```

```
    // A subscription only needs to be confirmed if the endpoint type is
    // HTTP/S, email, or in another AWS account.
    new ConfirmSubscriptionCommand({
      Token: token,
      TopicArn: topicArn,
      // If this is true, the subscriber cannot unsubscribe while
 unauthenticated.
      AuthenticateOnUnsubscribe: "false",
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '4bb5bce9-805a-5517-8333-e1d2cface90b',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:TOPIC_NAME:xxxxxxxx-
xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API ConfirmSubscription참조를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'vendor/autoload.php';
```

```
use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Verifies an endpoint owner's intent to receive messages by
 * validating the token sent to the endpoint by an earlier Subscribe action.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription_token = 'arn:aws:sns:us-east-1:111122223333:MyTopic:123456-
abcd-12ab-1234-12ba3dc1234a';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->confirmSubscription([
        'Token' => $subscription_token,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- API 세부 정보는 AWS SDK for PHP API ConfirmSubscription참조를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오AWS SDK와 함께 Amazon SNS 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **CreateTopic** CLI와 함께 사용

다음 코드 예제는 CreateTopic의 사용 방법을 보여줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- FIFO 주제에 생성 및 게시
- 대기열에 메시지 게시

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

특정 이름으로 주제를 생성합니다.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example shows how to use Amazon Simple Notification Service
    /// (Amazon SNS) to add a new Amazon SNS topic.
    /// </summary>
    public class CreateSNSTopic
    {
        public static async Task Main()
        {
            string topicName = "ExampleSNSTopic";

            IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

            var topicArn = await CreateSNSTopicAsync(client, topicName);
```

```
            Console.WriteLine($"New topic ARN: {topicArn}");
        }

        /// <summary>
        /// Creates a new SNS topic using the supplied topic name.
        /// </summary>
        /// <param name="client">The initialized SNS client object used to
        /// create the new topic.</param>
        /// <param name="topicName">A string representing the topic name.</param>
        /// <returns>The Amazon Resource Name (ARN) of the created topic.</
returns>
        public static async Task<string>
 CreateSNSTopicAsync(IAmazonSimpleNotificationService client, string topicName)
        {
            var request = new CreateTopicRequest
            {
                Name = topicName,
            };

            var response = await client.CreateTopicAsync(request);

            return response.TopicArn;
        }
    }
```

이름과 특정 FIFO 및 중복 제거 속성을 사용하여 새 주제를 생성합니다.

```
    /// <summary>
    /// Create a new topic with a name and specific FIFO and de-duplication
 attributes.
    /// </summary>
    /// <param name="topicName">The name for the topic.</param>
    /// <param name="useFifoTopic">True to use a FIFO topic.</param>
    /// <param name="useContentBasedDeduplication">True to use content-based de-
duplication.</param>
    /// <returns>The ARN of the new topic.</returns>
    public async Task<string> CreateTopicWithName(string topicName, bool
 useFifoTopic, bool useContentBasedDeduplication)
    {
        var createTopicRequest = new CreateTopicRequest()
        {
```

```
            Name = topicName,
        };

        if (useFifoTopic)
        {
            // Update the name if it is not correct for a FIFO topic.
            if (!topicName.EndsWith(".fifo"))
            {
                createTopicRequest.Name = topicName + ".fifo";
            }

            // Add the attributes from the method parameters.
            createTopicRequest.Attributes = new Dictionary<string, string>
            {
                { "FifoTopic", "true" }
            };
            if (useContentBasedDeduplication)
            {
                createTopicRequest.Attributes.Add("ContentBasedDeduplication",
"true");
            }
        }

        var createResponse = await
_amazonSNSClient.CreateTopicAsync(createTopicRequest);
        return createResponse.TopicArn;
    }
```

- API 세부 정보는 AWS SDK for .NET API CreateTopic참조를 참조하십시오.

C++

SDK for C++

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
//! Create an Amazon Simple Notification Service (Amazon SNS) topic.
```

```
/*!
  \param topicName: An Amazon SNS topic name.
  \param topicARNResult: String to return the Amazon Resource Name (ARN) for the
 topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::createTopic(const Aws::String &topicName,
                              Aws::String &topicARNResult,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::CreateTopicRequest request;
    request.SetName(topicName);

    const Aws::SNS::Model::CreateTopicOutcome outcome =
 snsClient.CreateTopic(request);

    if (outcome.IsSuccess()) {
        topicARNResult = outcome.GetResult().GetTopicArn();
        std::cout << "Successfully created an Amazon SNS topic " << topicName
                  << " with topic ARN '" << topicARNResult
                  << "'." << std::endl;

    }
    else {
        std::cerr << "Error creating topic " << topicName << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
        topicARNResult.clear();
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API CreateTopic참조를 참조하십시오.

CLI

AWS CLI

SNS 주제를 생성하려면

다음 create-topic 예제에서는 my-topic이라는 SNS 주제를 생성합니다.

```
aws sns create-topic \
    --name my-topic
```

출력:

```
{
    "ResponseMetadata": {
        "RequestId": "1469e8d7-1642-564e-b85d-a19b4b341f83"
    },
    "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
}
```

자세한 내용은 AWS 명령줄 인터페이스 사용 설명서의 Amazon SQS 및 Amazon SNS에서 AWS 명령줄 인터페이스 사용을 참조하십시오.

- API 세부 정보는 AWS CLI 명령 CreateTopic참조를 참조하십시오.

Go

SDK for Go V2

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// 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(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(context.TODO(), &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 CreateTopic참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
```

```
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicName>

                Where:
                   topicName - The name of the topic to create (for example,
 mytopic).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
```

```
            CreateTopicRequest request = CreateTopicRequest.builder()
                    .name(topicName)
                    .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API CreateTopic참조를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

별도의 모듈에서 클라이언트를 생성하고 내보냅니다.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
import { CreateTopicCommand } from "@aws-sdk/client-sns";
```

```
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicName - The name of the topic to create.
 */
export const createTopic = async (topicName = "TOPIC_NAME") => {
  const response = await snsClient.send(
    new CreateTopicCommand({ Name: topicName }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '087b8ad2-4593-50c4-a496-d7e90b82cf3e',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:TOPIC_NAME'
  // }
  return response;
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API CreateTopic참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun createSNSTopic(topicName: String): String {

    val request = CreateTopicRequest {
```

```
        name = topicName
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.createTopic(request)
        return result.topicArn.toString()
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 CreateTopic.

PHP

SDK for PHP

> **Note**
>
> 자세한 내용은 여기에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Create a Simple Notification Service topics in your AWS account at the
 requested region.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
```

```
$topicname = 'myTopic';

try {
    $result = $SnSclient->createTopic([
        'Name' => $topicname,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 자세한 정보는 AWS SDK for PHP 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for PHP API CreateTopic참조를 참조하십시오.

Python

SDK for Python(Boto3)

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def create_topic(self, name):
        """
        Creates a notification topic.
```

```
        :param name: The name of the topic to create.
        :return: The newly created topic.
        """
        try:
            topic = self.sns_resource.create_topic(Name=name)
            logger.info("Created topic %s with ARN %s.", name, topic.arn)
        except ClientError:
            logger.exception("Couldn't create topic %s.", name)
            raise
        else:
            return topic
```

- API에 대한 자세한 내용은 파이썬용AWS SDK (Boto3) API 레퍼런스를 참조하십시오 CreateTopic.

Ruby

SDK for Ruby

> **Note**
>
> 자세한 내용은 여기에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# This class demonstrates how to create an Amazon Simple Notification Service
 (SNS) topic.
class SNSTopicCreator
  # Initializes an SNS client.
  #
  # Utilizes the default AWS configuration for region and credentials.
  def initialize
    @sns_client = Aws::SNS::Client.new
  end

  # Attempts to create an SNS topic with the specified name.
  #
  # @param topic_name [String] The name of the SNS topic to create.
```

```
  # @return [Boolean] true if the topic was successfully created, false
 otherwise.
  def create_topic(topic_name)
    @sns_client.create_topic(name: topic_name)
    puts "The topic '#{topic_name}' was successfully created."
    true
  rescue Aws::SNS::Errors::ServiceError => e
    # Handles SNS service errors gracefully.
    puts "Error while creating the topic named '#{topic_name}': #{e.message}"
    false
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_name = "YourTopicName" # Replace with your topic name
  sns_topic_creator = SNSTopicCreator.new

  puts "Creating the topic '#{topic_name}'..."
  unless sns_topic_creator.create_topic(topic_name)
    puts "The topic was not created. Stopping program."
    exit 1
  end
end
```

- 자세한 정보는 AWS SDK for Ruby 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for Ruby API CreateTopic참조를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
async fn make_topic(client: &Client, topic_name: &str) -> Result<(), Error> {
    let resp = client.create_topic().name(topic_name).send().await?;
```

```
    println!(
        "Created topic with ARN: {}",
        resp.topic_arn().unwrap_or_default()
    );

    Ok(())
}
```

- API에 대한 자세한 내용은 Rust용AWS SDK API 레퍼런스를 참조하십시오 CreateTopic.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 자세한 내용은 GitHub 다음과 같습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   TRY.
       oo_result = lo_sns->createtopic( iv_name = iv_topic_name ). " oo_result
is returned for testing purposes. "
       MESSAGE 'SNS topic created' TYPE 'I'.
     CATCH /aws1/cx_snstopiclimitexcdex.
       MESSAGE 'Unable to create more topics. You have reached the maximum
number of topics allowed.' TYPE 'E'.
   ENDTRY.
```

- API 세부 정보는 SAP용AWS SDK ABAP API 참조를 참조하십시오 CreateTopic.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. AWS SDK와 함께 Amazon SNS 사용 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **DeleteTopic** CLI와 함께 사용

다음 코드 예제는 DeleteTopic의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 대기열에 메시지 게시

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

주제 ARN으로 주제를 삭제합니다.

```
/// <summary>
/// Delete a topic by its topic ARN.
/// </summary>
/// <param name="topicArn">The ARN of the topic.</param>
/// <returns>True if successful.</returns>
public async Task<bool> DeleteTopicByArn(string topicArn)
{
    var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
        new DeleteTopicRequest()
        {
            TopicArn = topicArn
        });
    return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
}
```

- API 세부 정보는 AWS SDK for .NET API DeleteTopic참조를 참조하십시오.

C++

SDK for C++

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
//! Delete an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::deleteTopic(const Aws::String &topicARN,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::DeleteTopicRequest request;
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::DeleteTopicOutcome outcome =
 snsClient.DeleteTopic(request);

    if (outcome.IsSuccess()) {
        std::cout << "Successfully deleted the Amazon SNS topic " << topicARN <<
 std::endl;
    }
    else {
        std::cerr << "Error deleting topic " << topicARN << ":" <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API DeleteTopic참조를 참조하십시오.

CLI

AWS CLI

SNS 주제를 삭제하려면

다음 delete-topic 예제에서는 지정된 SNS 주제를 삭제합니다.

```
aws sns delete-topic \
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic"
```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 DeleteTopic참조를 참조하십시오.

Go

SDK for Go V2

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// 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(topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(context.TODO(), &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 DeleteTopic참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:     <topicArn>

                Where:
                   topicArn - The ARN of the topic to delete.
                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        System.out.println("Deleting a topic with name: " + topicArn);
        deleteSNSTopic(snsClient, topicArn);
        snsClient.close();
    }

    public static void deleteSNSTopic(SnsClient snsClient, String topicArn) {
        try {
            DeleteTopicRequest request = DeleteTopicRequest.builder()
                    .topicArn(topicArn)
                    .build();

            DeleteTopicResponse result = snsClient.deleteTopic(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API DeleteTopic참조를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

별도의 모듈에서 클라이언트를 생성하고 내보냅니다.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
import { DeleteTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic to delete.
 */
export const deleteTopic = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new DeleteTopicCommand({ TopicArn: topicArn }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'a10e2886-5a8f-5114-af36-75bd39498332',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
```

```
  // }
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API DeleteTopic참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun deleteSNSTopic(topicArnVal: String) {

    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.deleteTopic(request)
        println("$topicArnVal was successfully deleted.")
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 DeleteTopic.

PHP

SDK for PHP

> **Note**
>
> 자세한 내용은 여기에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Deletes an SNS topic and all its subscriptions.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->deleteTopic([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- API 세부 정보는 AWS SDK for PHP API DeleteTopic참조를 참조하십시오.

Python

SDK for Python(Boto3)

> **Note**
>
> 자세한 내용은 다음과 같습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def delete_topic(topic):
        """
        Deletes a topic. All subscriptions to the topic are also deleted.
        """
        try:
            topic.delete()
            logger.info("Deleted topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't delete topic %s.", topic.arn)
            raise
```

- API에 대한 자세한 내용은 파이썬용AWS SDK (Boto3) API 레퍼런스를 참조하십시오 DeleteTopic.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 자세한 내용은 여기에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
TRY.
    lo_sns->deletetopic( iv_topicarn = iv_topic_arn ).
    MESSAGE 'SNS topic deleted.' TYPE 'I'.
  CATCH /aws1/cx_snsnotfoundexception.
    MESSAGE 'Topic does not exist.' TYPE 'E'.
ENDTRY.
```

- API 세부 정보는 SAP용AWS SDK ABAP API 참조를 참조하십시오 DeleteTopic.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. AWS SDK와 함께 Amazon SNS 사용 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **GetSMSAttributes** CLI와 함께 사용

다음 코드 예제는 `GetSMSAttributes`의 사용 방법을 보여줍니다.

C++

SDK for C++

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
//! Retrieve the default settings for sending SMS messages from your AWS account
 by using
```

```
//! Amazon Simple Notification Service (Amazon SNS).
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::getSMSType(const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::GetSMSAttributesRequest request;
    //Set the request to only retrieve the DefaultSMSType setting.
    //Without the following line, GetSMSAttributes would retrieve all settings.
    request.AddAttributes("DefaultSMSType");

    const Aws::SNS::Model::GetSMSAttributesOutcome outcome =
 snsClient.GetSMSAttributes(
            request);

    if (outcome.IsSuccess()) {
        const Aws::Map<Aws::String, Aws::String> attributes =
                outcome.GetResult().GetAttributes();
        if (!attributes.empty()) {
            for (auto const &att: attributes) {
                std::cout << att.first << ":  " << att.second << std::endl;
            }
        }
        else {
            std::cout
                    << "AwsDoc::SNS::getSMSType - an empty map of attributes was
 retrieved."
                    << std::endl;
        }
    }
    else {
        std::cerr << "Error while getting SMS Type: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 GetSMSAttributes를 참조하십시오.

CLI

AWS CLI

기본 SMS 메시지 속성을 나열하려면

다음 `get-sms-attributes` 예제에서는 SMS 메시지 전송의 기본 속성을 나열합니다.

```
aws sns get-sms-attributes
```

출력:

```
{
    "attributes": {
        "DefaultSenderID": "MyName"
    }
}
```

- API 세부 정보는 AWS CLI 명령 참조의 GetSMSAttributes를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import
 software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesRequest;
import
 software.amazon.awssdk.services.sns.model.GetSubscriptionAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.Iterator;
```

```
import java.util.Map;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetSMSAtrributes {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn>

                Where:
                   topicArn - The ARN of the topic from which to retrieve
 attributes.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        getSNSAttrutes(snsClient, topicArn);
        snsClient.close();
    }

    public static void getSNSAttrutes(SnsClient snsClient, String topicArn) {
        try {
            GetSubscriptionAttributesRequest request =
 GetSubscriptionAttributesRequest.builder()
                    .subscriptionArn(topicArn)
                    .build();

            // Get the Subscription attributes
```

```
            GetSubscriptionAttributesResponse res =
 snsClient.getSubscriptionAttributes(request);
            Map<String, String> map = res.attributes();

            // Iterate through the map
            Iterator iter = map.entrySet().iterator();
            while (iter.hasNext()) {
                Map.Entry entry = (Map.Entry) iter.next();
                System.out.println("[Key] : " + entry.getKey() + " [Value] : " +
 entry.getValue());
            }

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        System.out.println("\n\nStatus was good");
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 GetSMSAttributes를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

별도의 모듈에서 클라이언트를 생성하고 내보냅니다.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
import { GetSMSAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

export const getSmsAttributes = async () => {
  const response = await snsClient.send(
    // If you have not modified the account-level mobile settings of SNS,
    // the DefaultSMSType is undefined. For this example, it was set to
    // Transactional.
    new GetSMSAttributesCommand({ attributes: ["DefaultSMSType"] }),
  );

  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '67ad8386-4169-58f1-bdb9-debd281d48d5',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   attributes: { DefaultSMSType: 'Transactional' }
  // }
  return response;
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 GetSMSAttributes를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Get the type of SMS Message sent by default from the AWS SNS service.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->getSMSAttributes([
        'attributes' => ['DefaultSMSType'],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 자세한 정보는 AWS SDK for PHP 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for PHP API 참조의 GetSMSAttributes를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오AWS SDK와 함께 Amazon SNS 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **GetTopicAttributes** CLI와 함께 사용

다음 코드 예제는 GetTopicAttributes의 사용 방법을 보여줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;

    /// <summary>
    /// This example shows how to retrieve the attributes of an Amazon Simple
    /// Notification Service (Amazon SNS) topic.
    /// </summary>
    public class GetTopicAttributes
    {
        public static async Task Main()
        {
            string topicArn = "arn:aws:sns:us-
west-2:000000000000:ExampleSNSTopic";
            IAmazonSimpleNotificationService client = new
 AmazonSimpleNotificationServiceClient();

            var attributes = await GetTopicAttributesAsync(client, topicArn);
```

```
            DisplayTopicAttributes(attributes);
        }

        /// <summary>
        /// Given the ARN of the Amazon SNS topic, this method retrieves the
topic
        /// attributes.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object used
        /// to retrieve the attributes for the Amazon SNS topic.</param>
        /// <param name="topicArn">The ARN of the topic for which to retrieve
        /// the attributes.</param>
        /// <returns>A Dictionary of topic attributes.</returns>
        public static async Task<Dictionary<string, string>>
GetTopicAttributesAsync(
            IAmazonSimpleNotificationService client,
            string topicArn)
        {
            var response = await client.GetTopicAttributesAsync(topicArn);

            return response.Attributes;
        }

        /// <summary>
        /// This method displays the attributes for an Amazon SNS topic.
        /// </summary>
        /// <param name="topicAttributes">A Dictionary containing the
        /// attributes for an Amazon SNS topic.</param>
        public static void DisplayTopicAttributes(Dictionary<string, string>
topicAttributes)
        {
            foreach (KeyValuePair<string, string> entry in topicAttributes)
            {
                Console.WriteLine($"{entry.Key}: {entry.Value}\n");
            }
        }
    }
```

- API 세부 정보는 AWS SDK for .NET API GetTopicAttributes참조를 참조하십시오.

C++

SDK for C++

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
//! Retrieve the properties of an Amazon Simple Notification Service (Amazon SNS)
 topic.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::getTopicAttributes(const Aws::String &topicARN,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);
    Aws::SNS::Model::GetTopicAttributesRequest request;
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::GetTopicAttributesOutcome outcome =
 snsClient.GetTopicAttributes(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "Topic Attributes:" << std::endl;
        for (auto const &attribute: outcome.GetResult().GetAttributes()) {
            std::cout << "  * " << attribute.first << " : " << attribute.second
                      << std::endl;
        }
    }
    else {
        std::cerr << "Error while getting Topic attributes "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
```

```
}
```

- API 세부 정보는 AWS SDK for C++ API GetTopicAttributes참조를 참조하십시오.

CLI

AWS CLI

주제의 속성을 검색하려면

다음 `get-topic-attributes` 예제에서는 지정된 주제의 속성을 표시합니다.

```
aws sns get-topic-attributes \
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic"
```

출력:

```
{
    "Attributes": {
        "SubscriptionsConfirmed": "1",
        "DisplayName": "my-topic",
        "SubscriptionsDeleted": "0",
        "EffectiveDeliveryPolicy": "{\"http\":{\"defaultHealthyRetryPolicy
\":{\"minDelayTarget\":20,\"maxDelayTarget\":20,\"numRetries\":3,
\"numMaxDelayRetries\":0,\"numNoDelayRetries\":0,\"numMinDelayRetries\":0,
\"backoffFunction\":\"linear\"},\"disableSubscriptionOverrides\":false}}",
        "Owner": "123456789012",
        "Policy": "{\"Version\":\"2008-10-17\",\"Id\":\"__default_policy_ID
\",\"Statement\":[{\"Sid\":\"__default_statement_ID\",\"Effect\":
\"Allow\",\"Principal\":{\"AWS\":\"*\"},\"Action\":[\"SNS:Subscribe\",
\"SNS:ListSubscriptionsByTopic\",\"SNS:DeleteTopic\",\"SNS:GetTopicAttributes
\",\"SNS:Publish\",\"SNS:RemovePermission\",\"SNS:AddPermission\",
\"SNS:SetTopicAttributes\"],\"Resource\":\"arn:aws:sns:us-west-2:123456789012:my-
topic\",\"Condition\":{\"StringEquals\":{\"AWS:SourceOwner\":
\"0123456789012\"}}}]}",
        "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic",
        "SubscriptionsPending": "0"
    }
}
```

- API 세부 정보는 AWS CLI 명령 GetTopicAttributes참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.GetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetTopicAttributes {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn>

                Where:
                   topicArn - The ARN of the topic to look up.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
```

```
                .region(Region.US_EAST_1)
                .build();

        System.out.println("Getting attributes for a topic with name: " +
topicArn);
        getSNSTopicAttributes(snsClient, topicArn);
        snsClient.close();
    }

    public static void getSNSTopicAttributes(SnsClient snsClient, String
topicArn) {
        try {
            GetTopicAttributesRequest request =
GetTopicAttributesRequest.builder()
                    .topicArn(topicArn)
                    .build();

            GetTopicAttributesResponse result =
snsClient.getTopicAttributes(request);
            System.out.println("\n\nStatus is " +
result.sdkHttpResponse().statusCode() + "\n\nAttributes: \n\n"
                    + result.attributes());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API GetTopicAttributes참조를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

별도의 모듈에서 클라이언트를 생성하고 내보냅니다.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
import { GetTopicAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic to retrieve attributes for.
 */
export const getTopicAttributes = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new GetTopicAttributesCommand({
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '36b6a24e-5473-5d4e-ac32-ff72d9a73d94',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Attributes: {
  //     Policy: '{...}',
  //     Owner: 'xxxxxxxxxxxx',
  //     SubscriptionsPending: '1',
  //     TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic',
  //     TracingConfig: 'PassThrough',
  //     EffectiveDeliveryPolicy: '{"http":{"defaultHealthyRetryPolicy":
{"minDelayTarget":20,"maxDelayTarget":20,"numRetries":3,"numMaxDelayRetries":0,"numNoDela
{"headerContentType":"text/plain; charset=UTF-8"}}}',
```

```
  //     SubscriptionsConfirmed: '0',
  //     DisplayName: '',
  //     SubscriptionsDeleted: '1'
  //   }
  // }
  return response;
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API GetTopicAttributes참조를 참조하십시오.

JavaScript (v2) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예제 리포지토리에서 더 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set region
AWS.config.update({ region: "REGION" });

// Create promise and SNS service object
var getTopicAttribsPromise = new AWS.SNS({ apiVersion: "2010-03-31" })
  .getTopicAttributes({ TopicArn: "TOPIC_ARN" })
  .promise();

// Handle promise's fulfilled/rejected states
getTopicAttribsPromise
  .then(function (data) {
    console.log(data);
  })
  .catch(function (err) {
    console.error(err, err.stack);
  });
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API GetTopicAttributes참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun getSNSTopicAttributes(topicArnVal: String) {

    val request = GetTopicAttributesRequest {
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.getTopicAttributes(request)
        println("${result.attributes}")
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 GetTopicAttributes.

PHP

SDK for PHP

> **Note**
>
> 자세한 내용은 여기에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$SnSclient = new SnsClient([
```

```
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->getTopicAttributes([
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- API 세부 정보는 AWS SDK for PHP API GetTopicAttributes참조를 참조하십시오.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        oo_result = lo_sns->gettopicattributes( iv_topicarn = iv_topic_arn ). "
oo_result is returned for testing purposes. "
        DATA(lt_attributes) = oo_result->get_attributes( ).
        MESSAGE 'Retrieved attributes/properties of a topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 SAP용AWS SDK ABAP API 참조를 참조하십시오 GetTopicAttributes.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. AWS SDK와 함께 Amazon SNS 사용 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **ListPhoneNumbersOptedOut** CLI와 함께 사용

다음 코드 예제는 ListPhoneNumbersOptedOut의 사용 방법을 보여줍니다.

CLI

AWS CLI

SMS 메시지 옵트아웃을 나열하려면

다음 list-phone-numbers-opted-out 예제에서는 SMS 메시지 수신을 옵트아웃한 전화번호를 나열합니다.

```
aws sns list-phone-numbers-opted-out
```

출력:

```
{
    "phoneNumbers": [
        "+15555550100"
    ]
}
```

- API 세부 정보는 AWS CLI 명령 ListPhoneNumbersOptedOut참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutRequest;
import
 software.amazon.awssdk.services.sns.model.ListPhoneNumbersOptedOutResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListOptOut {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listOpts(snsClient);
        snsClient.close();
    }

    public static void listOpts(SnsClient snsClient) {
        try {
            ListPhoneNumbersOptedOutRequest request =
 ListPhoneNumbersOptedOutRequest.builder().build();
            ListPhoneNumbersOptedOutResponse result =
 snsClient.listPhoneNumbersOptedOut(request);
            System.out.println("Status is " +
 result.sdkHttpResponse().statusCode() + "\n\nPhone Numbers: \n\n"
                    + result.phoneNumbers());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API ListPhoneNumbersOptedOut참조를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Returns a list of phone numbers that are opted out of receiving SMS messages
 from your AWS SNS account.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listPhoneNumbersOptedOut();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 자세한 정보는 AWS SDK for PHP 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for PHP API ListPhoneNumbersOptedOut참조를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오AWS SDK와 함께 Amazon SNS 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **ListSubscriptions** CLI와 함께 사용

다음 코드 예제는 `ListSubscriptions`의 사용 방법을 보여줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    using System;
    using System.Collections.Generic;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example will retrieve a list of the existing Amazon Simple
    /// Notification Service (Amazon SNS) subscriptions.
    /// </summary>
    public class ListSubscriptions
    {
        public static async Task Main()
        {
            IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();
```

```
            Console.WriteLine("Enter a topic ARN to list subscriptions for a
 specific topic, " +
                              "or press Enter to list subscriptions for all
 topics.");
            var topicArn = Console.ReadLine();
            Console.WriteLine();

            var subscriptions = await GetSubscriptionsListAsync(client,
 topicArn);

            DisplaySubscriptionList(subscriptions);
        }

        /// <summary>
        /// Gets a list of the existing Amazon SNS subscriptions, optionally by
 specifying a topic ARN.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object used
        /// to obtain the list of subscriptions.</param>
        /// <param name="topicArn">The optional ARN of a specific topic. Defaults
 to null.</param>
        /// <returns>A list containing information about each subscription.</
returns>
        public static async Task<List<Subscription>>
 GetSubscriptionsListAsync(IAmazonSimpleNotificationService client, string
 topicArn = null)
        {
            var results = new List<Subscription>();

            if (!string.IsNullOrEmpty(topicArn))
            {
                var paginateByTopic = client.Paginators.ListSubscriptionsByTopic(
                    new ListSubscriptionsByTopicRequest()
                    {
                        TopicArn = topicArn,
                    });

                // Get the entire list using the paginator.
                await foreach (var subscription in paginateByTopic.Subscriptions)
                {
                    results.Add(subscription);
                }
            }
            else
```

```
            {
                var paginateAllSubscriptions =
client.Paginators.ListSubscriptions(new ListSubscriptionsRequest());

                // Get the entire list using the paginator.
                await foreach (var subscription in
paginateAllSubscriptions.Subscriptions)
                {
                    results.Add(subscription);
                }
            }

            return results;
        }

        /// <summary>
        /// Display a list of Amazon SNS subscription information.
        /// </summary>
        /// <param name="subscriptionList">A list containing details for existing
        /// Amazon SNS subscriptions.</param>
        public static void DisplaySubscriptionList(List<Subscription>
subscriptionList)
        {
            foreach (var subscription in subscriptionList)
            {
                Console.WriteLine($"Owner: {subscription.Owner}");
                Console.WriteLine($"Subscription ARN:
{subscription.SubscriptionArn}");
                Console.WriteLine($"Topic ARN: {subscription.TopicArn}");
                Console.WriteLine($"Endpoint: {subscription.Endpoint}");
                Console.WriteLine($"Protocol: {subscription.Protocol}");
                Console.WriteLine();
            }
        }
    }
```

- API 세부 정보는 AWS SDK for .NET API ListSubscriptions참조를 참조하십시오.

C++

SDK for C++

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
//! Retrieve a list of Amazon Simple Notification Service (Amazon SNS)
 subscriptions.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::listSubscriptions(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::String nextToken; // Next token is used to handle a paginated response.
    bool result = true;
    Aws::Vector<Aws::SNS::Model::Subscription> subscriptions;
    do {
        Aws::SNS::Model::ListSubscriptionsRequest request;

        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        const Aws::SNS::Model::ListSubscriptionsOutcome outcome =
 snsClient.ListSubscriptions(
                request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::SNS::Model::Subscription> &newSubscriptions =
                    outcome.GetResult().GetSubscriptions();
            subscriptions.insert(subscriptions.cend(), newSubscriptions.begin(),
                                 newSubscriptions.end());
        }
        else {
            std::cerr << "Error listing subscriptions "
```

```
                        << outcome.GetError().GetMessage()
                        <<
                        std::endl;
            result = false;
            break;
        }

        nextToken = outcome.GetResult().GetNextToken();
    } while (!nextToken.empty());

    if (result) {
        if (subscriptions.empty()) {
            std::cout << "No subscriptions found" << std::endl;
        }
        else {
            std::cout << "Subscriptions list:" << std::endl;
            for (auto const &subscription: subscriptions) {
                std::cout << "  * " << subscription.GetSubscriptionArn() <<
 std::endl;
            }
        }
    }
    return result;
}
```

- API 세부 정보는 AWS SDK for C++ API ListSubscriptions참조를 참조하십시오.

CLI

AWS CLI

SNS 구독을 나열하려면

다음 `list-subscriptions` 예는 AWS 계정의 SNS 구독 목록을 표시합니다.

```
aws sns list-subscriptions
```

출력:

```
{
    "Subscriptions": [
```

```
        {
            "Owner": "123456789012",
            "Endpoint": "my-email@example.com",
            "Protocol": "email",
            "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic",
            "SubscriptionArn": "arn:aws:sns:us-west-2:123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f"
        }
    ]
}
```

- API 세부 정보는 AWS CLI 명령 ListSubscriptions참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListSubscriptionsRequest;
import software.amazon.awssdk.services.sns.model.ListSubscriptionsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListSubscriptions {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
```

```
                .build();

        listSNSSubscriptions(snsClient);
        snsClient.close();
    }

    public static void listSNSSubscriptions(SnsClient snsClient) {
        try {
            ListSubscriptionsRequest request = ListSubscriptionsRequest.builder()
                    .build();

            ListSubscriptionsResponse result =
 snsClient.listSubscriptions(request);
            System.out.println(result.subscriptions());

        } catch (SnsException e) {

            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API ListSubscriptions참조를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

별도의 모듈에서 클라이언트를 생성하고 내보냅니다.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
```

```
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
import { ListSubscriptionsByTopicCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic for which you wish to list
 subscriptions.
 */
export const listSubscriptionsByTopic = async (topicArn = "TOPIC_ARN") => {
  const response = await snsClient.send(
    new ListSubscriptionsByTopicCommand({ TopicArn: topicArn }),
  );

  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '0934fedf-0c4b-572e-9ed2-a3e38fadb0c8',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Subscriptions: [
  //     {
  //       SubscriptionArn: 'PendingConfirmation',
  //       Owner: '901487484989',
  //       Protocol: 'email',
  //       Endpoint: 'corepyle@amazon.com',
  //       TopicArn: 'arn:aws:sns:us-east-1:901487484989:mytopic'
  //     }
  //   ]
  // }
  return response;
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.

- API 세부 정보는 AWS SDK for JavaScript API ListSubscriptions참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listSNSSubscriptions() {

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listSubscriptions(ListSubscriptionsRequest {})
        response.subscriptions?.forEach { sub ->
            println("Sub ARN is ${sub.subscriptionArn}")
            println("Sub protocol is ${sub.protocol}")
        }
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 ListSubscriptions.

PHP

SDK for PHP

> **Note**
>
> 자세한 내용은 여기에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

```
use Aws\Sns\SnsClient;


/**
 * Returns a list of Amazon SNS subscriptions in the requested region.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listSubscriptions();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- API 세부 정보는 AWS SDK for PHP API ListSubscriptions참조를 참조하십시오.

Python

SDK for Python(Boto3)

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""
```

```
    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def list_subscriptions(self, topic=None):
        """
        Lists subscriptions for the current account, optionally limited to a
        specific topic.

        :param topic: When specified, only subscriptions to this topic are
returned.
        :return: An iterator that yields the subscriptions.
        """
        try:
            if topic is None:
                subs_iter = self.sns_resource.subscriptions.all()
            else:
                subs_iter = topic.subscriptions.all()
            logger.info("Got subscriptions.")
        except ClientError:
            logger.exception("Couldn't get subscriptions.")
            raise
        else:
            return subs_iter
```

- API에 대한 자세한 내용은 파이썬용AWS SDK (Boto3) API 레퍼런스를 참조하십시오 ListSubscriptions.

Ruby

SDK for Ruby

> **Note**
>
> 자세한 내용은 여기에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# This class demonstrates how to list subscriptions to an Amazon Simple
 Notification Service (SNS) topic
class SnsSubscriptionLister
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Lists subscriptions for a given SNS topic
  # @param topic_arn [String] The ARN of the SNS topic
  # @return [Types::ListSubscriptionsResponse] subscriptions: The response object
  def list_subscriptions(topic_arn)
    @logger.info("Listing subscriptions for topic: #{topic_arn}")
    subscriptions = @sns_client.list_subscriptions_by_topic(topic_arn: topic_arn)
    subscriptions.subscriptions.each do |subscription|
      @logger.info("Subscription endpoint: #{subscription.endpoint}")
    end
    subscriptions
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error listing subscriptions: #{e.message}")
    raise
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  sns_client = Aws::SNS::Client.new
  topic_arn = "SNS_TOPIC_ARN" # Replace with your SNS topic ARN
  lister = SnsSubscriptionLister.new(sns_client)

  begin
    lister.list_subscriptions(topic_arn)
  rescue StandardError => e
    puts "Failed to list subscriptions: #{e.message}"
    exit 1
  end
end
```

- 자세한 정보는 AWS SDK for Ruby 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for Ruby API ListSubscriptions참조를 참조하십시오.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        oo_result = lo_sns->listsubscriptions( ).                " oo_result is
returned for testing purposes. "
        DATA(lt_subscriptions) = oo_result->get_subscriptions( ).
        MESSAGE 'Retrieved list of subscribers.' TYPE 'I'.
      CATCH /aws1/cx_rt_generic.
        MESSAGE 'Unable to list subscribers.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 SAP용AWS SDK ABAP API 참조를 참조하십시오 ListSubscriptions.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. AWS SDK와 함께 Amazon SNS 사용 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **ListTopics** CLI와 함께 사용

다음 코드 예제는 `ListTopics`의 사용 방법을 보여줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
   using System;
   using System.Collections.Generic;
   using System.Threading.Tasks;
   using Amazon.SimpleNotificationService;
   using Amazon.SimpleNotificationService.Model;

   /// <summary>
   /// Lists the Amazon Simple Notification Service (Amazon SNS)
   /// topics for the current account.
   /// </summary>
   public class ListSNSTopics
   {
       public static async Task Main()
       {
           IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

           await GetTopicListAsync(client);
       }

       /// <summary>
       /// Retrieves the list of Amazon SNS topics in groups of up to 100
       /// topics.
       /// </summary>
       /// <param name="client">The initialized Amazon SNS client object used
       /// to retrieve the list of topics.</param>
       public static async Task
GetTopicListAsync(IAmazonSimpleNotificationService client)
       {
           // If there are more than 100 Amazon SNS topics, the call to
           // ListTopicsAsync will return a value to pass to the
           // method to retrieve the next 100 (or less) topics.
           string nextToken = string.Empty;

           do
           {
               var response = await client.ListTopicsAsync(nextToken);
               DisplayTopicsList(response.Topics);
               nextToken = response.NextToken;
           }
           while (!string.IsNullOrEmpty(nextToken));
       }
```

```
        /// <summary>
        /// Displays the list of Amazon SNS Topic ARNs.
        /// </summary>
        /// <param name="topicList">The list of Topic ARNs.</param>
        public static void DisplayTopicsList(List<Topic> topicList)
        {
            foreach (var topic in topicList)
            {
                Console.WriteLine($"{topic.TopicArn}");
            }
        }
    }
```

- API 세부 정보는 AWS SDK for .NET API ListTopics참조를 참조하십시오.

C++

SDK for C++

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
//! Retrieve a list of Amazon Simple Notification Service (Amazon SNS) topics.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::listTopics(const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::String nextToken; // Next token is used to handle a paginated response.
    bool result = true;
    do {
        Aws::SNS::Model::ListTopicsRequest request;
```

```
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        const Aws::SNS::Model::ListTopicsOutcome outcome = snsClient.ListTopics(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "Topics list:" << std::endl;
            for (auto const &topic: outcome.GetResult().GetTopics()) {
                std::cout << "  * " << topic.GetTopicArn() << std::endl;
            }
        }
        else {
            std::cerr << "Error listing topics " <<
 outcome.GetError().GetMessage() <<
                      std::endl;
            result = false;
            break;
        }

        nextToken = outcome.GetResult().GetNextToken();
    } while (!nextToken.empty());

    return result;
}
```

- API 세부 정보는 AWS SDK for C++ API ListTopics참조를 참조하십시오.

CLI

AWS CLI

SNS 주제를 나열하려면

다음 `list-topics` 예는 AWS 계정의 모든 SNS 주제를 나열합니다.

```
aws sns list-topics
```

출력:

```
{
```

```
    "Topics": [
        {
            "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
        }
    ]
}
```

- API 세부 정보는 AWS CLI 명령 ListTopics참조를 참조하십시오.

Go

SDK for Go V2

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 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() {
 sdkConfig, err := config.LoadDefaultConfig(context.TODO())
 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(context.TODO())
  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 ListTopics참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.ListTopicsRequest;
```

```
import software.amazon.awssdk.services.sns.model.ListTopicsResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListTopics {
    public static void main(String[] args) {
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listSNSTopics(snsClient);
        snsClient.close();
    }

    public static void listSNSTopics(SnsClient snsClient) {
        try {
            ListTopicsRequest request = ListTopicsRequest.builder()
                    .build();

            ListTopicsResponse result = snsClient.listTopics(request);
            System.out.println(
                    "Status was " + result.sdkHttpResponse().statusCode() + "\n
\nTopics\n\n" + result.topics());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API ListTopics참조를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

별도의 모듈에서 클라이언트를 생성하고 내보냅니다.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
import { ListTopicsCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

export const listTopics = async () => {
  const response = await snsClient.send(new ListTopicsCommand({}));
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '936bc5ad-83ca-53c2-b0b7-9891167b909e',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   Topics: [ { TopicArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic' } ]
  // }
  return response;
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API ListTopics참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun listSNSTopics() {

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.listTopics(ListTopicsRequest { })
        response.topics?.forEach { topic ->
            println("The topic ARN is ${topic.topicArn}")
        }
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 ListTopics.

PHP

SDK for PHP

> **Note**
>
> 자세한 내용은 여기에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

```
use Aws\Sns\SnsClient;


/**
 * Returns a list of the requester's topics from your AWS SNS account in the
 region specified.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->listTopics();
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- API 세부 정보는 AWS SDK for PHP API ListTopics참조를 참조하십시오.

Python

SDK for Python(Boto3)

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""
```

```
    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def list_topics(self):
        """
        Lists topics for the current account.

        :return: An iterator that yields the topics.
        """
        try:
            topics_iter = self.sns_resource.topics.all()
            logger.info("Got topics.")
        except ClientError:
            logger.exception("Couldn't get topics.")
            raise
        else:
            return topics_iter
```

- API에 대한 자세한 내용은 파이썬용AWS SDK (Boto3) API 레퍼런스를 참조하십시오 ListTopics.

Ruby

SDK for Ruby

> **Note**
>
> 자세한 내용은 여기에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require "aws-sdk-sns"  # v2: require 'aws-sdk'
```

```
def list_topics?(sns_client)
  sns_client.topics.each do |topic|
    puts topic.arn
rescue StandardError => e
  puts "Error while listing the topics: #{e.message}"
  end
end

def run_me

  region = "REGION"
  sns_client = Aws::SNS::Resource.new(region: region)

  puts "Listing the topics."

  if list_topics?(sns_client)
  else
    puts "The bucket was not created. Stopping program."
    exit 1
  end
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- 자세한 정보는 AWS SDK for Ruby 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for Ruby API ListTopics참조를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
async fn show_topics(client: &Client) -> Result<(), Error> {
```

```
    let resp = client.list_topics().send().await?;

    println!("Topic ARNs:");

    for topic in resp.topics() {
        println!("{}", topic.topic_arn().unwrap_or_default());
    }

    Ok(())
}
```

- API에 대한 자세한 내용은 Rust용AWS SDK API 레퍼런스를 참조하십시오 ListTopics.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 자세한 내용은 GitHub 다음과 같습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        oo_result = lo_sns->listtopics( ).            " oo_result is returned for
testing purposes. "
        DATA(lt_topics) = oo_result->get_topics( ).
        MESSAGE 'Retrieved list of topics.' TYPE 'I'.
      CATCH /aws1/cx_rt_generic.
        MESSAGE 'Unable to list topics.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 SAP용AWS SDK ABAP API 참조를 참조하십시오 ListTopics.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. AWS SDK와 함께 Amazon SNS 사용 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **Publish** CLI와 함께 사용

다음 코드 예제는 Publish의 사용 방법을 보여줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- FIFO 주제에 생성 및 게시
- SMS 문자 메시지 게시
- 대기열에 메시지 게시

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

주제에 메시지를 게시합니다.

```
    using System;
    using System.Threading.Tasks;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    /// <summary>
    /// This example publishes a message to an Amazon Simple Notification
    /// Service (Amazon SNS) topic.
    /// </summary>
    public class PublishToSNSTopic
    {
        public static async Task Main()
        {
            string topicArn = "arn:aws:sns:us-
east-2:000000000000:ExampleSNSTopic";
            string messageText = "This is an example message to publish to the
 ExampleSNSTopic.";
```

```
            IAmazonSimpleNotificationService client = new
AmazonSimpleNotificationServiceClient();

            await PublishToTopicAsync(client, topicArn, messageText);
        }

        /// <summary>
        /// Publishes a message to an Amazon SNS topic.
        /// </summary>
        /// <param name="client">The initialized client object used to publish
        /// to the Amazon SNS topic.</param>
        /// <param name="topicArn">The ARN of the topic.</param>
        /// <param name="messageText">The text of the message.</param>
        public static async Task PublishToTopicAsync(
            IAmazonSimpleNotificationService client,
            string topicArn,
            string messageText)
        {
            var request = new PublishRequest
            {
                TopicArn = topicArn,
                Message = messageText,
            };

            var response = await client.PublishAsync(request);

            Console.WriteLine($"Successfully published message ID:
{response.MessageId}");
        }
    }
```

그룹, 복제, 속성 옵션을 사용하여 주제에 메시지를 게시하세요.

```
    /// <summary>
    /// Publish messages using user settings.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task PublishMessages()
    {
        Console.WriteLine("Now we can publish messages.");
```

```
        var keepSendingMessages = true;
        string? deduplicationId = null;
        string? toneAttribute = null;
        while (keepSendingMessages)
        {
            Console.WriteLine();
            var message = GetUserResponse("Enter a message to publish.", "This is
a sample message");

            if (_useFifoTopic)
            {
                Console.WriteLine("Because you are using a FIFO topic, you must
set a message group ID." +
                                  "\r\nAll messages within the same group will be
received in the order " +
                                  "they were published.");

                Console.WriteLine();
                var messageGroupId = GetUserResponse("Enter a message group ID
for this message:", "1");

                if (!_useContentBasedDeduplication)
                {
                    Console.WriteLine("Because you are not using content-based
deduplication, " +
                                      "you must enter a deduplication ID.");

                    Console.WriteLine("Enter a deduplication ID for this
message.");
                    deduplicationId = GetUserResponse("Enter a deduplication ID
for this message.", "1");
                }

                if (GetYesNoResponse("Add an attribute to this message?"))
                {
                    Console.WriteLine("Enter a number for an attribute.");
                    for (int i = 0; i < _tones.Length; i++)
                    {
                        Console.WriteLine($"\t{i + 1}. {_tones[i]}");
                    }

                    var selection = GetUserResponse("", "1");
                    int.TryParse(selection, out var selectionNumber);
```

```
                    if (selectionNumber > 0 && selectionNumber < _tones.Length)
                    {
                        toneAttribute = _tones[selectionNumber - 1];
                    }
                }

                var messageID = await SnsWrapper.PublishToTopicWithAttribute(
                    _topicArn, message, "tone", toneAttribute, deduplicationId,
 messageGroupId);

                Console.WriteLine($"Message published with id {messageID}.");
            }

            keepSendingMessages = GetYesNoResponse("Send another message?",
 false);
        }
    }
```

사용자의 선택을 게시 작업에 적용합니다.

```
    /// <summary>
    /// Publish a message to a topic with an attribute and optional deduplication
 and group IDs.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="message">The message to publish.</param>
    /// <param name="attributeName">The optional attribute for the message.</
param>
    /// <param name="attributeValue">The optional attribute value for the
 message.</param>
    /// <param name="deduplicationId">The optional deduplication ID for the
 message.</param>
    /// <param name="groupId">The optional group ID for the message.</param>
    /// <returns>The ID of the message published.</returns>
    public async Task<string> PublishToTopicWithAttribute(
        string topicArn,
        string message,
        string? attributeName = null,
        string? attributeValue = null,
        string? deduplicationId = null,
        string? groupId = null)
```

```
    {
        var publishRequest = new PublishRequest()
        {
            TopicArn = topicArn,
            Message = message,
            MessageDeduplicationId = deduplicationId,
            MessageGroupId = groupId
        };

        if (attributeValue != null)
        {
            // Add the string attribute if it exists.
            publishRequest.MessageAttributes =
                new Dictionary<string, MessageAttributeValue>
                {
                    { attributeName!, new MessageAttributeValue() { StringValue =
attributeValue, DataType = "String"} }
                };
        }

        var publishResponse = await
_amazonSNSClient.PublishAsync(publishRequest);
        return publishResponse.MessageId;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 Publish를 참조하십시오.

C++

SDK for C++

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
//! Send a message to an Amazon Simple Notification Service (Amazon SNS) topic.
/*!
  \param message: The message to publish.
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
```

```
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::publishToTopic(const Aws::String &message,
                                 const Aws::String &topicARN,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

    if (outcome.IsSuccess()) {
        std::cout << "Message published successfully with id '"
                  << outcome.GetResult().GetMessageId() << "'." << std::endl;
    }
    else {
        std::cerr << "Error while publishing message "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

속성이 있는 메시지를 게시하세요.

```
        static const Aws::String TONE_ATTRIBUTE("tone");
        static const Aws::Vector<Aws::String> TONES = {"cheerful", "funny",
 "serious",
                                                       "sincere"};

        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

        Aws::SNS::Model::PublishRequest request;
```

```
        request.SetTopicArn(topicARN);
        Aws::String message = askQuestion("Enter a message text to publish.  ");
        request.SetMessage(message);

        if (filteringMessages && askYesNoQuestion(
                "Add an attribute to this message? (y/n) ")) {
            for (size_t i = 0; i < TONES.size(); ++i) {
                std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
            }
            int selection = askQuestionForIntRange(
                    "Enter a number for an attribute. ",
                    1, static_cast<int>(TONES.size()));
            Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
            messageAttributeValue.SetDataType("String");
            messageAttributeValue.SetStringValue(TONES[selection - 1]);
            request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
        }

        Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

        if (outcome.IsSuccess()) {
            std::cout << "Your message was successfully published." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::Publish. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
```

- API 세부 정보는 AWS SDK for C++ API 참조의 Publish를 참조하십시오.

CLI

AWS CLI

예제 1: 주제에 메시지를 게시하려면

다음 publish 예제에서는 지정된 Amazon SNS 주제에 지정된 메시지를 게시합니다. 메시지는 줄 바꿈을 포함할 수 있는 텍스트 파일에서 제공됩니다.

```
aws sns publish \
    --topic-arn "arn:aws:sns:us-west-2:123456789012:my-topic" \
    --message file://message.txt
```

message.txt의 콘텐츠:

```
Hello World
Second Line
```

출력:

```
{
    "MessageId": "123a45b6-7890-12c3-45d6-111122223333"
}
```

예제 2: 전화번호에 SMS 메시지를 게시하려면

다음 publish 예제에서는 Hello world! 메시지를 전화번호 +1-555-555-0100에 게시합니다.

```
aws sns publish \
    --message "Hello world!" \
    --phone-number +1-555-555-0100
```

출력:

```
{
    "MessageId": "123a45b6-7890-12c3-45d6-333322221111"
}
```

- API 세부 정보는 AWS CLI 명령 참조의 Publish를 참조하세요.

Go

SDK for Go V2

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// 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(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(context.TODO(), &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 Publish를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PublishTopic {
```

```
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <topicArn>

                Where:
                   message - The message text to send.
                   topicArn - The ARN of the topic to publish.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String topicArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTopic(snsClient, message, topicArn);
        snsClient.close();
    }

    public static void pubTopic(SnsClient snsClient, String message, String
 topicArn) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .topicArn(topicArn)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status is " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Publish를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

별도의 모듈에서 클라이언트를 생성하고 내보냅니다.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
import { PublishCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string | Record<string, any>} message - The message to send. Can be a
 plain string or an object
 *                                                  if you are using the `json`
 `MessageStructure`.
 * @param {string} topicArn - The ARN of the topic to which you would like to
 publish.
 */
export const publish = async (
  message = "Hello from SNS!",
  topicArn = "TOPIC_ARN",
) => {
```

```
  const response = await snsClient.send(
    new PublishCommand({
      Message: message,
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'e7f77526-e295-5325-9ee4-281a43ad1f05',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   MessageId: 'xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

그룹, 복제, 속성 옵션을 사용하여 주제에 메시지를 게시하세요.

```
  async publishMessages() {
    const message = await this.prompter.input({
      message: MESSAGES.publishMessagePrompt,
    });

    let groupId, deduplicationId, choices;

    if (this.isFifo) {
      await this.logger.log(MESSAGES.groupIdNotice);
      groupId = await this.prompter.input({
        message: MESSAGES.groupIdPrompt,
      });

      if (this.autoDedup === false) {
        await this.logger.log(MESSAGES.deduplicationIdNotice);
        deduplicationId = await this.prompter.input({
          message: MESSAGES.deduplicationIdPrompt,
        });
      }
```

```
      choices = await this.prompter.checkbox({
        message: MESSAGES.messageAttributesPrompt,
        choices: toneChoices,
      });
    }

    await this.snsClient.send(
      new PublishCommand({
        TopicArn: this.topicArn,
        Message: message,
        ...(groupId
          ? {
              MessageGroupId: groupId,
            }
          : {}),
        ...(deduplicationId
          ? {
              MessageDeduplicationId: deduplicationId,
            }
          : {}),
        ...(choices
          ? {
              MessageAttributes: {
                tone: {
                  DataType: "String.Array",
                  StringValue: JSON.stringify(choices),
                },
              },
            }
          : {}),
      }),
    );

    const publishAnother = await this.prompter.confirm({
      message: MESSAGES.publishAnother,
    });

    if (publishAnother) {
      await this.publishMessages();
    }
  }
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 Publish를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun pubTopic(topicArnVal: String, messageVal: String) {

    val request = PublishRequest {
        message = messageVal
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 Publish를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Sends a message to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 자세한 정보는 AWS SDK for PHP 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for PHP API 참조의 Publish를 참조하십시오.

PowerShell

를 위한 도구 PowerShell

예 1: 이 예제에서는 MessageAttribute 선언된 단일 인라인으로 메시지를 게시하는 방법을 보여 줍니다.

```
Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -
Message "Hello" -MessageAttribute
 @{'City'=[Amazon.SimpleNotificationService.Model.MessageAttributeValue]@{DataType='Strin
 StringValue ='AnyCity'}}
```

예 2: 이 예에서는 여러 개를 미리 MessageAttributes 선언하여 메시지를 게시하는 방법을 보여 줍니다.

```
$cityAttributeValue = New-Object
 Amazon.SimpleNotificationService.Model.MessageAttributeValue
$cityAttributeValue.DataType = "String"
$cityAttributeValue.StringValue = "AnyCity"

$populationAttributeValue = New-Object
 Amazon.SimpleNotificationService.Model.MessageAttributeValue
$populationAttributeValue.DataType = "Number"
$populationAttributeValue.StringValue = "1250800"

$messageAttributes = New-Object System.Collections.Hashtable
$messageAttributes.Add("City", $cityAttributeValue)
$messageAttributes.Add("Population", $populationAttributeValue)

Publish-SNSMessage -TopicArn "arn:aws:sns:us-west-2:123456789012:my-topic" -
Message "Hello" -MessageAttribute $messageAttributes
```

- API에 대한 세부 정보는 AWS Tools for PowerShell Cmdlet에 게시 참조를 참조하십시오.

Python

SDK for Python(Boto3)

> **Note**
>
> 자세한 내용은 다음과 같습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

구독이 속성을 기준으로 필터링할 수 있도록 속성이 포함된 메시지를 게시합니다.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def publish_message(topic, message, attributes):
        """
        Publishes a message, with attributes, to a topic. Subscriptions can be
 filtered
        based on message attributes so that a subscription receives messages only
        when specified attributes are present.

        :param topic: The topic to publish to.
        :param message: The message to publish.
        :param attributes: The key-value attributes to attach to the message.
 Values
                           must be either `str` or `bytes`.
        :return: The ID of the message.
        """
        try:
            att_dict = {}
            for key, value in attributes.items():
                if isinstance(value, str):
                    att_dict[key] = {"DataType": "String", "StringValue": value}
                elif isinstance(value, bytes):
```

```
                    att_dict[key] = {"DataType": "Binary", "BinaryValue": value}
            response = topic.publish(Message=message, MessageAttributes=att_dict)
            message_id = response["MessageId"]
            logger.info(
                "Published message with attributes %s to topic %s.",
                attributes,
                topic.arn,
            )
        except ClientError:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise
        else:
            return message_id
```

구독자의 프로토콜에 따라 다른 양식을 사용하는 메시지를 게시합니다.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def publish_multi_message(
        topic, subject, default_message, sms_message, email_message
    ):
        """
        Publishes a multi-format message to a topic. A multi-format message takes
        different forms based on the protocol of the subscriber. For example,
        an SMS subscriber might receive a short version of the message
        while an email subscriber could receive a longer version.

        :param topic: The topic to publish to.
        :param subject: The subject of the message.
        :param default_message: The default version of the message. This version
 is
```

```
                                sent to subscribers that have protocols that are
not
                                otherwise specified in the structured message.
        :param sms_message: The version of the message sent to SMS subscribers.
        :param email_message: The version of the message sent to email
subscribers.
        :return: The ID of the message.
        """
        try:
            message = {
                "default": default_message,
                "sms": sms_message,
                "email": email_message,
            }
            response = topic.publish(
                Message=json.dumps(message), Subject=subject,
MessageStructure="json"
            )
            message_id = response["MessageId"]
            logger.info("Published multi-format message to topic %s.", topic.arn)
        except ClientError:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise
        else:
            return message_id
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 Publish를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# Service class for sending messages using Amazon Simple Notification Service
 (SNS)
class SnsMessageSender
  # Initializes the SnsMessageSender with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end

  # Sends a message to a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param message [String] The message to send
  # @return [Boolean] true if message was successfully sent, false otherwise
  def send_message(topic_arn, message)
    @sns_client.publish(topic_arn: topic_arn, message: message)
    @logger.info("Message sent successfully to #{topic_arn}.")
    true
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error while sending the message: #{e.message}")
    false
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = "SNS_TOPIC_ARN" # Should be replaced with a real topic ARN
  message = "MESSAGE"         # Should be replaced with the actual message
 content

  sns_client = Aws::SNS::Client.new
  message_sender = SnsMessageSender.new(sns_client)

  @logger.info("Sending message.")
  unless message_sender.send_message(topic_arn, message)
    @logger.error("Message sending failed. Stopping program.")
    exit 1
  end
end
```

- 자세한 정보는 AWS SDK for Ruby 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for Ruby API 참조의 Publish를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
async fn subscribe_and_publish(
    client: &Client,
    topic_arn: &str,
    email_address: &str,
) -> Result<(), Error> {
    println!("Receiving on topic with ARN: `{}`", topic_arn);

    let rsp = client
        .subscribe()
        .topic_arn(topic_arn)
        .protocol("email")
        .endpoint(email_address)
        .send()
        .await?;

    println!("Added a subscription: {:?}", rsp);

    let rsp = client
        .publish()
        .topic_arn(topic_arn)
        .message("hello sns!")
        .send()
        .await?;

    println!("Published message: {:?}", rsp);

    Ok(())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 Publish을 참조하십시오.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        oo_result = lo_sns->publish(              " oo_result is returned for
testing purposes. "
          iv_topicarn = iv_topic_arn
          iv_message = iv_message
        ).
        MESSAGE 'Message published to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- API에 대한 세부 정보는 AWS SDK for SAP ABAP API 참조의 Publish를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오AWS SDK와 함께 Amazon SNS 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **SetSMSAttributes** CLI와 함께 사용

다음 코드 예제는 SetSMSAttributes의 사용 방법을 보여줍니다.

C++

SDK for C++

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon SNS를 사용하여 DefaultSMSType 속성을 설정하는 방법입니다.

```
//! Set the default settings for sending SMS messages.
/*!
  \param smsType: The type of SMS message that you will send by default.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::setSMSType(const Aws::String &smsType,
                             const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SetSMSAttributesRequest request;
    request.AddAttributes("DefaultSMSType", smsType);

    const Aws::SNS::Model::SetSMSAttributesOutcome outcome =
 snsClient.SetSMSAttributes(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "SMS Type set successfully " << std::endl;
    }
    else {
        std::cerr << "Error while setting SMS Type: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 SetSMSAttributes를 참조하십시오.

CLI

AWS CLI

SMS 메시지 속성을 설정하려면

다음 `set-sms-attributes` 예제에서는 SMS 메시지의 기본 발신자 ID를 `MyName`으로 설정합니다.

```
aws sns set-sms-attributes \
    --attributes DefaultSenderID=MyName
```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 참조의 SetSMSAttributes를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }

    public static void setSNSAttributes(SnsClient snsClient, HashMap<String,
 String> attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                    .attributes(attributes)
                    .build();

            SetSmsAttributesResponse result =
 snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ".
 Status was "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 SetSMSAttributes를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

별도의 모듈에서 클라이언트를 생성하고 내보냅니다.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
import { SetSMSAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {"Transactional" | "Promotional"} defaultSmsType
 */
export const setSmsType = async (defaultSmsType = "Transactional") => {
  const response = await snsClient.send(
    new SetSMSAttributesCommand({
      attributes: {
        // Promotional – (Default) Noncritical messages, such as marketing
 messages.
        // Transactional – Critical messages that support customer transactions,
        // such as one-time passcodes for multi-factor authentication.
        DefaultSMSType: defaultSmsType,
      },
    }),
  );
  console.log(response);
  // {
```

```
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '1885b977-2d7e-535e-8214-e44be727e265',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 SetSMSAttributes를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

try {
    $result = $SnSclient->SetSMSAttributes([
        'attributes' => [
            'DefaultSMSType' => 'Transactional',
        ],
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
```

```
    error_log($e->getMessage());
}
```

- 자세한 정보는 AWS SDK for PHP 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for PHP API 참조의 SetSMSAttributes를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오AWS SDK와 함께 Amazon SNS 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **SetSubscriptionAttributes** CLI와 함께 사용

다음 코드 예제는 SetSubscriptionAttributes의 사용 방법을 보여줍니다.

CLI

AWS CLI

구독 속성을 설정하려면

다음 set-subscription-attributes 예제에서는 SQS 구독에 RawMessageDelivery 속성을 설정합니다.

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name RawMessageDelivery \
    --attribute-value true
```

이 명령은 출력을 생성하지 않습니다.

다음 set-subscription-attributes 예제에서는 SQS 구독에 FilterPolicy 속성을 설정합니다.

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name FilterPolicy \
    --attribute-value "{ \"anyMandatoryKey\": [\"any\", \"of\", \"these\"] }"
```

이 명령은 출력을 생성하지 않습니다.

다음 `set-subscription-attributes` 예제에서는 SQS 구독에서 `FilterPolicy` 속성을 제거합니다.

```
aws sns set-subscription-attributes \
    --subscription-arn arn:aws:sns:us-
east-1:123456789012:mytopic:f248de18-2cf6-578c-8592-b6f1eaa877dc \
    --attribute-name FilterPolicy \
    --attribute-value "{}"
```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 SetSubscriptionAttributes참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.ArrayList;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class UseMessageFilterPolicy {
    public static void main(String[] args) {
        final String usage = """
```

```
            Usage:    <subscriptionArn>

            Where:
               subscriptionArn - The ARN of a subscription.

            """;

    if (args.length != 1) {
        System.out.println(usage);
        System.exit(1);
    }

    String subscriptionArn = args[0];
    SnsClient snsClient = SnsClient.builder()
            .region(Region.US_EAST_1)
            .build();

    usePolicy(snsClient, subscriptionArn);
    snsClient.close();
}

public static void usePolicy(SnsClient snsClient, String subscriptionArn) {
    try {
        SNSMessageFilterPolicy fp = new SNSMessageFilterPolicy();
        // Add a filter policy attribute with a single value
        fp.addAttribute("store", "example_corp");
        fp.addAttribute("event", "order_placed");

        // Add a prefix attribute
        fp.addAttributePrefix("customer_interests", "bas");

        // Add an anything-but attribute
        fp.addAttributeAnythingBut("customer_interests", "baseball");

        // Add a filter policy attribute with a list of values
        ArrayList<String> attributeValues = new ArrayList<>();
        attributeValues.add("rugby");
        attributeValues.add("soccer");
        attributeValues.add("hockey");
        fp.addAttribute("customer_interests", attributeValues);

        // Add a numeric attribute
        fp.addAttribute("price_usd", "=", 0);
```

```
            // Add a numeric attribute with a range
            fp.addAttributeRange("price_usd", ">", 0, "<=", 100);

            // Apply the filter policy attributes to an Amazon SNS subscription
            fp.apply(snsClient, subscriptionArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API SetSubscriptionAttributes참조를 참조하십시오.

Python

SDK for Python(Boto3)

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def add_subscription_filter(subscription, attributes):
        """
        Adds a filter policy to a subscription. A filter policy is a key and a
```

```
        list of values that are allowed. When a message is published, it must
have an
        attribute that passes the filter or it will not be sent to the
subscription.

        :param subscription: The subscription the filter policy is attached to.
        :param attributes: A dictionary of key-value pairs that define the
filter.
        """
        try:
            att_policy = {key: [value] for key, value in attributes.items()}
            subscription.set_attributes(
                AttributeName="FilterPolicy",
AttributeValue=json.dumps(att_policy)
            )
            logger.info("Added filter to subscription %s.", subscription.arn)
        except ClientError:
            logger.exception(
                "Couldn't add filter to subscription %s.", subscription.arn
            )
            raise
```

- API에 대한 자세한 내용은 파이썬용AWS SDK (Boto3) API 레퍼런스를 참조하십시오 SetSubscriptionAttributes.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. AWS SDK와 함께 Amazon SNS 사용 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **SetSubscriptionAttributesRedrivePolicy** CLI와 함께 사용

다음 코드 예시에서는 SetSubscriptionAttributesRedrivePolicy을 사용하는 방법을 보여 줍니다.

Java

SDK for Java 1.x

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
// Specify the ARN of the Amazon SNS subscription.
String subscriptionArn =
    "arn:aws:sns:us-east-2:123456789012:MyEndpoint:1234a567-
bc89-012d-3e45-6fg7h890123i";

// Specify the ARN of the Amazon SQS queue to use as a dead-letter queue.
String redrivePolicy =
    "{\"deadLetterTargetArn\":\"arn:aws:sqs:us-
east-2:123456789012:MyDeadLetterQueue\"}";

// Set the specified Amazon SQS queue as a dead-letter queue
// of the specified Amazon SNS subscription by setting the RedrivePolicy
 attribute.
SetSubscriptionAttributesRequest request = new SetSubscriptionAttributesRequest()
    .withSubscriptionArn(subscriptionArn)
    .withAttributeName("RedrivePolicy")
    .withAttributeValue(redrivePolicy);
sns.setSubscriptionAttributes(request);
```

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오AWS SDK와 함께 Amazon SNS 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **SetTopicAttributes** CLI와 함께 사용

다음 코드 예제는 SetTopicAttributes의 사용 방법을 보여줍니다.

CLI

AWS CLI

주제에 대한 속성을 설정하려면

다음 `set-topic-attributes` 예제에서는 지정된 주제에 `DisplayName` 속성을 설정합니다.

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name DisplayName \
    --attribute-value MyTopicDisplayName
```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 SetTopicAttributes참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetTopicAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
```

```
public class SetTopicAttributes {

    public static void main(String[] args) {
        final String usage = """

                Usage:    <attribute> <topicArn> <value>

                Where:
                   attribute - The attribute action to use. Valid parameters are:
 Policy | DisplayName | DeliveryPolicy .
                   topicArn - The ARN of the topic.\s
                   value - The value for the attribute.
                """;

        if (args.length < 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String attribute = args[0];
        String topicArn = args[1];
        String value = args[2];

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        setTopAttr(snsClient, attribute, topicArn, value);
        snsClient.close();
    }

    public static void setTopAttr(SnsClient snsClient, String attribute, String
 topicArn, String value) {
        try {
            SetTopicAttributesRequest request =
 SetTopicAttributesRequest.builder()
                    .attributeName(attribute)
                    .attributeValue(value)
                    .topicArn(topicArn)
                    .build();

            SetTopicAttributesResponse result =
 snsClient.setTopicAttributes(request);
            System.out.println(
```

```
                    "\n\nStatus was " + result.sdkHttpResponse().statusCode() +
 "\n\nTopic " + request.topicArn()
                            + " updated " + request.attributeName() + " to " +
 request.attributeValue());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API SetTopicAttributes참조를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

별도의 모듈에서 클라이언트를 생성하고 내보냅니다.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
import { SetTopicAttributesCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

export const setTopicAttributes = async (
```

```
  topicArn = "TOPIC_ARN",
  attributeName = "DisplayName",
  attributeValue = "Test Topic",
) => {
  const response = await snsClient.send(
    new SetTopicAttributesCommand({
      AttributeName: attributeName,
      AttributeValue: attributeValue,
      TopicArn: topicArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'd1b08d0e-e9a4-54c3-b8b1-d03238d2b935',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API SetTopicAttributes참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun setTopAttr(attribute: String?, topicArnVal: String?, value: String?)
 {
```

```
    val request = SetTopicAttributesRequest {
        attributeName = attribute
        attributeValue = value
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.setTopicAttributes(request)
        println("Topic ${request.topicArn} was updated.")
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 SetTopicAttributes.

PHP

SDK for PHP

> **Note**
>
> 자세한 내용은 여기에서 확인할 수 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Configure the message delivery status attributes for an Amazon SNS Topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
```

```
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);
$attribute = 'Policy | DisplayName | DeliveryPolicy';
$value = 'First Topic';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->setTopicAttributes([
        'AttributeName' => $attribute,
        'AttributeValue' => $value,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- API 세부 정보는 AWS SDK for PHP API SetTopicAttributes참조를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
# Service class to enable an SNS resource with a specified policy
class SnsResourceEnabler
  # Initializes the SnsResourceEnabler with an SNS resource client
  #
  # @param sns_resource [Aws::SNS::Resource] The SNS resource client
  def initialize(sns_resource)
    @sns_resource = sns_resource
    @logger = Logger.new($stdout)
  end
```

```
  # Sets a policy on a specified SNS topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource to include in the policy
  # @param policy_name [String] The name of the policy attribute to set
  def enable_resource(topic_arn, resource_arn, policy_name)
    policy = generate_policy(topic_arn, resource_arn)
    topic = @sns_resource.topic(topic_arn)

    topic.set_attributes({
                           attribute_name: policy_name,
                           attribute_value: policy
                         })
    @logger.info("Policy #{policy_name} set successfully for topic
#{topic_arn}.")
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Failed to set policy: #{e.message}")
  end

  private

  # Generates a policy string with dynamic resource ARNs
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param resource_arn [String] The ARN of the resource
  # @return [String] The policy as a JSON string
  def generate_policy(topic_arn, resource_arn)
    {
      Version: "2008-10-17",
      Id: "__default_policy_ID",
      Statement: [{
                    Sid: "__default_statement_ID",
                    Effect: "Allow",
                    Principal: { "AWS": "*" },
                    Action: ["SNS:Publish"],
                    Resource: topic_arn,
                    Condition: {
                      ArnEquals: {
                        "AWS:SourceArn": resource_arn
                      }
                    }
                  }]
    }.to_json
```

```
  end
end

# Example usage:
if $PROGRAM_NAME == __FILE__
  topic_arn = "MY_TOPIC_ARN"     # Should be replaced with a real topic ARN
  resource_arn = "MY_RESOURCE_ARN" # Should be replaced with a real resource ARN
  policy_name = "POLICY_NAME"    # Typically, this is "Policy"

  sns_resource = Aws::SNS::Resource.new
  enabler = SnsResourceEnabler.new(sns_resource)

  enabler.enable_resource(topic_arn, resource_arn, policy_name)
end
```

- 자세한 정보는 AWS SDK for Ruby 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for Ruby API SetTopicAttributes참조를 참조하십시오.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        lo_sns->settopicattributes(
            iv_topicarn = iv_topic_arn
            iv_attributename  = iv_attribute_name
            iv_attributevalue = iv_attribute_value
        ).
        MESSAGE 'Set/updated SNS topic attributes.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
    ENDTRY.
```

- API 세부 정보는 SAP용AWS SDK ABAP API 참조를 참조하십시오 SetTopicAttributes.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. AWS SDK와 함께 Amazon SNS 사용 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **Subscribe** CLI와 함께 사용

다음 코드 예제는 Subscribe의 사용 방법을 보여줍니다.

작업 예시는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- FIFO 주제에 생성 및 게시
- 대기열에 메시지 게시

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

주제에 대한 이메일 주소를 구독하세요.

```
        /// <summary>
        /// Creates a new subscription to a topic.
        /// </summary>
        /// <param name="client">The initialized Amazon SNS client object, used
        /// to create an Amazon SNS subscription.</param>
        /// <param name="topicArn">The ARN of the topic to subscribe to.</param>
        /// <returns>A SubscribeResponse object which includes the subscription
        /// ARN for the new subscription.</returns>
        public static async Task<SubscribeResponse> TopicSubscribeAsync(
            IAmazonSimpleNotificationService client,
```

```
            string topicArn)
        {
            SubscribeRequest request = new SubscribeRequest()
            {
                TopicArn = topicArn,
                ReturnSubscriptionArn = true,
                Protocol = "email",
                Endpoint = "recipient@example.com",
            };

            var response = await client.SubscribeAsync(request);

            return response;
        }
```

선택적 필터를 사용하여 주제 대기열을 구독하세요.

```
    /// <summary>
    /// Subscribe a queue to a topic with optional filters.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="useFifoTopic">The optional filtering policy for the
subscription.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <returns>The ARN of the new subscription.</returns>
    public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
filterPolicy, string queueArn)
    {
        var subscribeRequest = new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "sqs",
            Endpoint = queueArn
        };

        if (!string.IsNullOrEmpty(filterPolicy))
        {
            subscribeRequest.Attributes = new Dictionary<string, string>
{ { "FilterPolicy", filterPolicy } };
        }
```

```
        var subscribeResponse = await
 _amazonSNSClient.SubscribeAsync(subscribeRequest);
        return subscribeResponse.SubscriptionArn;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 Subscribe를 참조하십시오.

C++

SDK for C++

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

주제에 대한 이메일 주소를 구독하세요.

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to an email address.
/*!
  \param topicARN: An SNS topic Amazon Resource Name (ARN).
  \param emailAddress: An email address.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::subscribeEmail(const Aws::String &topicARN,
                                 const Aws::String &emailAddress,
                                 const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("email");
    request.SetEndpoint(emailAddress);

    const Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);
```

```
    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

모바일 애플리케이션을 구독하여 주제를 구독하십시오.

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to a mobile app.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param endpointARN: The ARN for a mobile app or device endpoint.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::subscribeApp(const Aws::String &topicARN,
                          const Aws::String &endpointARN,
                          const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("application");
    request.SetEndpoint(endpointARN);

    const Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
```

```
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

Lambda 함수를 구독하여 주제를 등록하십시오.

```
//! Subscribe to an Amazon Simple Notification Service (Amazon SNS) topic with
 delivery to an AWS Lambda function.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param lambdaFunctionARN: The ARN for an AWS Lambda function.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::subscribeLambda(const Aws::String &topicARN,
                                  const Aws::String &lambdaFunctionARN,
                                  const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::SubscribeRequest request;
    request.SetTopicArn(topicARN);
    request.SetProtocol("lambda");
    request.SetEndpoint(lambdaFunctionARN);

    const Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Subscribed successfully." << std::endl;
        std::cout << "Subscription ARN '" <<
 outcome.GetResult().GetSubscriptionArn()
```

```
                  << "'." << std::endl;
    }
    else {
        std::cerr << "Error while subscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

SQS 대기열에서 주제를 구독하십시오.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);

            Aws::SNS::Model::SubscribeOutcome outcome =
 snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
 outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN <<
 "."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
```

```
                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
```

필터를 사용하여 주제를 구독하십시오.

```
        static const Aws::String TONE_ATTRIBUTE("tone");
        static const Aws::Vector<Aws::String> TONES = {"cheerful", "funny",
 "serious",
                                                       "sincere"};

        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::SNS::SNSClient snsClient(clientConfiguration);

            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);
            if (isFifoTopic) {
                if (first) {
                    std::cout << "Subscriptions to a FIFO topic can have
 filters."
                              << std::endl;
                    std::cout
                            << "If you add a filter to this subscription, then
 only the filtered messages "
                            << "will be received in the queue." << std::endl;
                    std::cout << "For information about message filtering, "
                              << "see https://docs.aws.amazon.com/sns/latest/dg/
sns-message-filtering.html"
                              << std::endl;
                    std::cout << "For this example, you can filter messages by a
 \""
```

```
                            << TONE_ATTRIBUTE << "\" attribute." << std::endl;
                }

                std::ostringstream ostringstream;
                ostringstream << "Filter messages for \"" << queueName
                              << "\"'s subscription to the topic \""
                              << topicName << "\"?  (y/n)";

                // Add filter if user answers yes.
                if (askYesNoQuestion(ostringstream.str())) {
                    Aws::String jsonPolicy = getFilterPolicyFromUser();
                    if (!jsonPolicy.empty()) {
                        filteringMessages = true;

                        std::cout << "This is the filter policy for this
subscription."
                                  << std::endl;
                        std::cout << jsonPolicy << std::endl;

                        request.AddAttributes("FilterPolicy", jsonPolicy);
                    }
                    else {
                        std::cout
                                << "Because you did not select any attributes, no
filter "
                                << "will be added to this subscription." <<
std::endl;
                    }
                }
            }  // if (isFifoTopic)
            Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN <<
"."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
```

```
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }

//! Routine that lets the user select attributes for a subscription filter
 policy.
/*!
 \sa getFilterPolicyFromUser()
 \return Aws::String: The filter policy as JSON.
 */
Aws::String AwsDoc::TopicsAndQueues::getFilterPolicyFromUser() {
    std::cout
            << "You can filter messages by one or more of the following \""
            << TONE_ATTRIBUTE << "\" attributes." << std::endl;

    std::vector<Aws::String> filterSelections;
    int selection;
    do {
        for (size_t j = 0; j < TONES.size(); ++j) {
            std::cout << "  " << (j + 1) << ". " << TONES[j]
                      << std::endl;
        }
        selection = askQuestionForIntRange(
                "Enter a number (or enter zero to stop adding more). ",
                0, static_cast<int>(TONES.size()));

        if (selection != 0) {
            const Aws::String &selectedTone(TONES[selection - 1]);
            // Add the tone to the selection if it is not already added.
            if (std::find(filterSelections.begin(),
                          filterSelections.end(),
                          selectedTone)
                == filterSelections.end()) {
                filterSelections.push_back(selectedTone);
```

```
            }
        }
    } while (selection != 0);

    Aws::String result;
    if (!filterSelections.empty()) {
        std::ostringstream jsonPolicyStream;
        jsonPolicyStream << "{ \"" << TONE_ATTRIBUTE << "\": [";


        for (size_t j = 0; j < filterSelections.size(); ++j) {
            jsonPolicyStream << "\"" << filterSelections[j] << "\"";
            if (j < filterSelections.size() - 1) {
                jsonPolicyStream << ",";
            }
        }
        jsonPolicyStream << "] }";

        result = jsonPolicyStream.str();
    }

    return result;
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 Subscribe를 참조하십시오.

CLI

AWS CLI

주제를 구독하려면

다음 `subscribe` 명령은 이메일 주소로 지정된 주제를 구독합니다.

```
aws sns subscribe \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --protocol email \
    --notification-endpoint my-email@example.com
```

출력:

```
{
```

```
    "SubscriptionArn": "pending confirmation"
}
```

- API 세부 정보는 AWS CLI 명령 참조의 Subscribe를 참조하세요.

Go

SDK for Go V2

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

선택적 필터를 사용하여 주제 대기열을 구독하세요.

```
// 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(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(context.TODO(), &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}
```

- API 세부 정보는 AWS SDK for Go API 참조의 Subscribe를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

주제에 대한 이메일 주소를 구독하세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeEmail {
    public static void main(String[] args) {
        final String usage = """
                Usage:     <topicArn> <email>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   email - The email address to use.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String email = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subEmail(snsClient, topicArn, email);
        snsClient.close();
    }

    public static void subEmail(SnsClient snsClient, String topicArn, String
 email) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("email")
                    .endpoint(email)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();
```

```
            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN: " + result.subscriptionArn() +
 "\n\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

주제에 대한 HTTP 엔드포인트를 구독하십시오.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeHTTPS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn> <url>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   url - The HTTPS endpoint that you want to receive
 notifications.
                """;

        if (args.length < 2) {
```

```
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String url = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subHTTPS(snsClient, topicArn, url);
        snsClient.close();
    }

    public static void subHTTPS(SnsClient snsClient, String topicArn, String url)
 {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("https")
                    .endpoint(url)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            System.out.println("Subscription ARN is " + result.subscriptionArn()
 + "\n\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

Lambda 함수를 구독하여 주제를 등록하십시오.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
```

```
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeLambda {

    public static void main(String[] args) {

        final String usage = """

                Usage:    <topicArn> <lambdaArn>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   lambdaArn - The ARN of an AWS Lambda function.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String lambdaArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnValue = subLambda(snsClient, topicArn, lambdaArn);
        System.out.println("Subscription ARN: " + arnValue);
        snsClient.close();
    }

    public static String subLambda(SnsClient snsClient, String topicArn, String
 lambdaArn) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
```

```
                    .protocol("lambda")
                    .endpoint(lambdaArn)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
            return result.subscriptionArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Subscribe를 참조하십시오.

JavaScript

(v3) 용 JavaScript SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

별도의 모듈에서 클라이언트를 생성하고 내보냅니다.

```
import { SNSClient } from "@aws-sdk/client-sns";

// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic for which you wish to confirm
 a subscription.
 * @param {string} emailAddress - The email address that is subscribed to the
 topic.
 */
export const subscribeEmail = async (
  topicArn = "TOPIC_ARN",
  emailAddress = "usern@me.com",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "email",
      TopicArn: topicArn,
      Endpoint: emailAddress,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
};
```

모바일 애플리케이션에서 주제를 구독하세요.

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
```

```
 * @param {string} topicArn - The ARN of the topic the subscriber is subscribing
 to.
 * @param {string} endpoint - The Endpoint ARN of an application. This endpoint
 is created
 *                            when an application registers for notifications.
 */
export const subscribeApp = async (
  topicArn = "TOPIC_ARN",
  endpoint = "ENDPOINT",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "application",
      TopicArn: topicArn,
      Endpoint: endpoint,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
  return response;
};
```

Lambda 함수를 구독하여 주제를 등록하십시오.

```
import { SubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} topicArn - The ARN of the topic the subscriber is subscribing
 to.
 * @param {string} endpoint - The Endpoint ARN of and AWS Lambda function.
 */
```

```
export const subscribeLambda = async (
  topicArn = "TOPIC_ARN",
  endpoint = "ENDPOINT",
) => {
  const response = await snsClient.send(
    new SubscribeCommand({
      Protocol: "lambda",
      TopicArn: topicArn,
      Endpoint: endpoint,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: 'c8e35bcd-b3c0-5940-9f66-06f6fcc108f0',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'pending confirmation'
  // }
  return response;
};
```

SQS 대기열에서 주제를 구독하십시오.

```
import { SubscribeCommand, SNSClient } from "@aws-sdk/client-sns";

const client = new SNSClient({});

export const subscribeQueue = async (
  topicArn = "TOPIC_ARN",
  queueArn = "QUEUE_ARN",
) => {
  const command = new SubscribeCommand({
    TopicArn: topicArn,
    Protocol: "sqs",
    Endpoint: queueArn,
  });
```

```
  const response = await client.send(command);
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '931e13d9-5e2b-543f-8781-4e9e494c5ff2',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:subscribe-queue-
test-430895:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

필터를 사용하여 주제를 구독하십시오.

```
import { SubscribeCommand, SNSClient } from "@aws-sdk/client-sns";

const client = new SNSClient({});

export const subscribeQueueFiltered = async (
  topicArn = "TOPIC_ARN",
  queueArn = "QUEUE_ARN",
) => {
  const command = new SubscribeCommand({
    TopicArn: topicArn,
    Protocol: "sqs",
    Endpoint: queueArn,
    Attributes: {
      // This subscription will only receive messages with the 'event' attribute
 set to 'order_placed'.
      FilterPolicyScope: "MessageAttributes",
      FilterPolicy: JSON.stringify({
        event: ["order_placed"],
      }),
    },
  });

  const response = await client.send(command);
```

```
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '931e13d9-5e2b-543f-8781-4e9e494c5ff2',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   },
  //   SubscriptionArn: 'arn:aws:sns:us-east-1:xxxxxxxxxxxx:subscribe-queue-
test-430895:xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx'
  // }
  return response;
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 Subscribe를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

주제에 대한 이메일 주소를 구독하세요.

```
suspend fun subEmail(topicArnVal: String, email: String): String {

    val request = SubscribeRequest {
        protocol = "email"
        endpoint = email
        returnSubscriptionArn = true
        topicArn = topicArnVal
    }
```

```
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        return result.subscriptionArn.toString()
    }
}
```

Lambda 함수를 구독하여 주제를 등록하십시오.

```
suspend fun subLambda(topicArnVal: String?, lambdaArn: String?) {

    val request = SubscribeRequest {
        protocol = "lambda"
        endpoint = lambdaArn
        returnSubscriptionArn = true
        topicArn = topicArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.subscribe(request)
        println(" The subscription Arn is ${result.subscriptionArn}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 Subscribe를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> 더 많은 정보가 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

주제에 대한 이메일 주소를 구독하세요.

```
require 'vendor/autoload.php';
```

```
use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation
 message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'email';
$endpoint = 'sample@example.com';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

주제에 대한 HTTP 엔드포인트를 구독하십시오.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
```

```
use Aws\Sns\SnsClient;


/**
 * Prepares to subscribe an endpoint by sending the endpoint a confirmation
 message.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$protocol = 'https';
$endpoint = 'https://';
$topic = 'arn:aws:sns:us-east-1:111122223333:MyTopic';

try {
    $result = $SnSclient->subscribe([
        'Protocol' => $protocol,
        'Endpoint' => $endpoint,
        'ReturnSubscriptionArn' => true,
        'TopicArn' => $topic,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- API 세부 정보는 AWS SDK for PHP API 참조의 Subscribe를 참조하십시오.

Python

SDK for Python(Boto3)

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

주제에 대한 이메일 주소를 구독하세요.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def subscribe(topic, protocol, endpoint):
        """
        Subscribes an endpoint to the topic. Some endpoint types, such as email,
        must be confirmed before their subscriptions are active. When a
 subscription
        is not confirmed, its Amazon Resource Number (ARN) is set to
        'PendingConfirmation'.

        :param topic: The topic to subscribe to.
        :param protocol: The protocol of the endpoint, such as 'sms' or 'email'.
        :param endpoint: The endpoint that receives messages, such as a phone
 number
                         (in E.164 format) for SMS messages, or an email address
 for
                         email messages.
        :return: The newly added subscription.
        """
        try:
            subscription = topic.subscribe(
                Protocol=protocol, Endpoint=endpoint, ReturnSubscriptionArn=True
```

```
            )
            logger.info("Subscribed %s %s to topic %s.", protocol, endpoint,
 topic.arn)
        except ClientError:
            logger.exception(
                "Couldn't subscribe %s %s to topic %s.", protocol, endpoint,
 topic.arn
            )
            raise
        else:
            return subscription
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 Subscribe를 참조하십시오.

Ruby

SDK for Ruby

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

주제에 대한 이메일 주소를 구독하세요.

```
require "aws-sdk-sns"
require "logger"

# Represents a service for creating subscriptions in Amazon Simple Notification
 Service (SNS)
class SubscriptionService
  # Initializes the SubscriptionService with an SNS client
  #
  # @param sns_client [Aws::SNS::Client] The SNS client
  def initialize(sns_client)
    @sns_client = sns_client
    @logger = Logger.new($stdout)
  end
```

```
  # Attempts to create a subscription to a topic
  #
  # @param topic_arn [String] The ARN of the SNS topic
  # @param protocol [String] The subscription protocol (e.g., email)
  # @param endpoint [String] The endpoint that receives the notifications (email
 address)
  # @return [Boolean] true if subscription was successfully created, false
 otherwise
  def create_subscription(topic_arn, protocol, endpoint)
    @sns_client.subscribe(topic_arn: topic_arn, protocol: protocol, endpoint:
 endpoint)
    @logger.info("Subscription created successfully.")
    true
  rescue Aws::SNS::Errors::ServiceError => e
    @logger.error("Error while creating the subscription: #{e.message}")
    false
  end
end

# Main execution if the script is run directly
if $PROGRAM_NAME == __FILE__
  protocol = "email"
  endpoint = "EMAIL_ADDRESS" # Should be replaced with a real email address
  topic_arn = "TOPIC_ARN"    # Should be replaced with a real topic ARN

  sns_client = Aws::SNS::Client.new
  subscription_service = SubscriptionService.new(sns_client)

  @logger.info("Creating the subscription.")
  unless subscription_service.create_subscription(topic_arn, protocol, endpoint)
    @logger.error("Subscription creation failed. Stopping program.")
    exit 1
  end
end
```

- 자세한 정보는 AWS SDK for Ruby 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for Ruby API 참조의 Subscribe를 참조하십시오.

Rust

SDK for Rust

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

주제에 대한 이메일 주소를 구독하세요.

```
async fn subscribe_and_publish(
    client: &Client,
    topic_arn: &str,
    email_address: &str,
) -> Result<(), Error> {
    println!("Receiving on topic with ARN: `{}`", topic_arn);

    let rsp = client
        .subscribe()
        .topic_arn(topic_arn)
        .protocol("email")
        .endpoint(email_address)
        .send()
        .await?;

    println!("Added a subscription: {:?}", rsp);

    let rsp = client
        .publish()
        .topic_arn(topic_arn)
        .message("hello sns!")
        .send()
        .await?;

    println!("Published message: {:?}", rsp);

    Ok(())
}
```

- API 세부 정보는 AWS SDK for Rust API 참조의 Subscribe을 참조하십시오.

SAP ABAP

SAP ABAP용 SDK

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

주제에 대한 이메일 주소를 구독하세요.

```
   TRY.
       oo_result = lo_sns->subscribe(                       "oo_result is
returned for testing purposes."
               iv_topicarn = iv_topic_arn
               iv_protocol = 'email'
               iv_endpoint = iv_email_address
               iv_returnsubscriptionarn = abap_true
           ).
       MESSAGE 'Email address subscribed to SNS topic.' TYPE 'I'.
     CATCH /aws1/cx_snsnotfoundexception.
       MESSAGE 'Topic does not exist.' TYPE 'E'.
     CATCH /aws1/cx_snssubscriptionlmte00.
       MESSAGE 'Unable to create subscriptions. You have reached the maximum
number of subscriptions allowed.' TYPE 'E'.
   ENDTRY.
```

- API에 대한 세부 정보는 AWS SDK for SAP ABAP API 참조의 Subscribe를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오AWS SDK와 함께 Amazon SNS 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **TagResource** CLI와 함께 사용

다음 코드 예제는 TagResource의 사용 방법을 보여줍니다.

CLI

AWS CLI

주제에 태그를 추가하려면

다음 `tag-resource` 예제에서는 지정된 Amazon SNS 주제에 메타데이터 태그를 추가합니다.

```
aws sns tag-resource \
    --resource-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --tags Key=Team,Value=Alpha
```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 TagResource참조를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.Tag;
import software.amazon.awssdk.services.sns.model.TagResourceRequest;
import java.util.ArrayList;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
```

```
 */
public class AddTags {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn>

                Where:
                   topicArn - The ARN of the topic to which tags are added.

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        addTopicTags(snsClient, topicArn);
        snsClient.close();
    }

    public static void addTopicTags(SnsClient snsClient, String topicArn) {
        try {
            Tag tag = Tag.builder()
                    .key("Team")
                    .value("Development")
                    .build();

            Tag tag2 = Tag.builder()
                    .key("Environment")
                    .value("Gamma")
                    .build();

            List<Tag> tagList = new ArrayList<>();
            tagList.add(tag);
            tagList.add(tag2);

            TagResourceRequest tagResourceRequest = TagResourceRequest.builder()
                    .resourceArn(topicArn)
```

```
                    .tags(tagList)
                    .build();

            snsClient.tagResource(tagResourceRequest);
            System.out.println("Tags have been added to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API TagResource참조를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 자세한 내용은 에서 확인할 수 GitHub 있습니다. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun addTopicTags(topicArn: String) {

    val tag = Tag {
        key = "Team"
        value = "Development"
    }

    val tag2 = Tag {
        key = "Environment"
        value = "Gamma"
    }

    val tagList = mutableListOf<Tag>()
    tagList.add(tag)
    tagList.add(tag2)
```

```
    val request = TagResourceRequest {
        resourceArn = topicArn
        tags = tagList
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.tagResource(request)
        println("Tags have been added to $topicArn")
    }
}
```

- API 세부 정보는 Kotlin API용AWS SDK 레퍼런스를 참조하세요 TagResource.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오. AWS SDK와 함께 Amazon SNS 사용 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK 또는 **Unsubscribe** CLI와 함께 사용

다음 코드 예제는 Unsubscribe의 사용 방법을 보여줍니다.

작업 예제는 대규모 프로그램에서 발췌한 코드이며 컨텍스트에 맞춰 실행해야 합니다. 다음 코드 예제에서는 컨텍스트 내에서 이 작업을 확인할 수 있습니다.

- 대기열에 메시지 게시

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

구독 ARN으로 주제 구독을 취소합니다.

```
    /// <summary>
```

```
    /// Unsubscribe from a topic by a subscription ARN.
    /// </summary>
    /// <param name="subscriptionArn">The ARN of the subscription.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> UnsubscribeByArn(string subscriptionArn)
    {
        var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
            new UnsubscribeRequest()
            {
                SubscriptionArn = subscriptionArn
            });
        return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
    }
```

- API 세부 정보는 AWS SDK for .NET API 참조의 Unsubscribe를 참조하십시오.

C++

SDK for C++

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
//! Delete a subscription to an Amazon Simple Notification Service (Amazon SNS)
 topic.
/*!
  \param subscriptionARN: The Amazon Resource Name (ARN) for an Amazon SNS topic
 subscription.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::unsubscribe(const Aws::String &subscriptionARN,
                              const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::UnsubscribeRequest request;
    request.SetSubscriptionArn(subscriptionARN);
```

```
    const Aws::SNS::Model::UnsubscribeOutcome outcome =
 snsClient.Unsubscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Unsubscribed successfully " << std::endl;
    }
    else {
        std::cerr << "Error while unsubscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 Unsubscribe를 참조하십시오.

CLI

AWS CLI

주제 구독을 취소하려면

다음 unsubscribe 예제에서는 주제에서 지정된 구독을 삭제합니다.

```
aws sns unsubscribe \
    --subscription-arn arn:aws:sns:us-west-2:0123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f
```

이 명령은 출력을 생성하지 않습니다.

- API 세부 정보는 AWS CLI 명령 참조의 Unsubscribe를 참조하세요.

Java

SDK for Java 2.x

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class Unsubscribe {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <subscriptionArn>

                Where:
                   subscriptionArn - The ARN of the subscription to delete.
                """;

        if (args.length < 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String subscriptionArn = args[0];
        SnsClient snsClient = SnsClient.builder()
```

```
                .region(Region.US_EAST_1)
                .build();

        unSub(snsClient, subscriptionArn);
        snsClient.close();
    }

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                    .subscriptionArn(subscriptionArn)
                    .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode()
                    + "\n\nSubscription was removed for " +
 request.subscriptionArn());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Unsubscribe를 참조하십시오.

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예시 리포지토리에서 전체 예시를 찾고 설정 및 실행하는 방법을 배워보세요.

별도의 모듈에서 클라이언트를 생성하고 내보냅니다.

```
import { SNSClient } from "@aws-sdk/client-sns";
```

```
// The AWS Region can be provided here using the `region` property. If you leave
 it blank
// the SDK will default to the region set in your AWS config.
export const snsClient = new SNSClient({});
```

SDK 및 클라이언트 모듈을 가져오고 API를 호출합니다.

```
import { UnsubscribeCommand } from "@aws-sdk/client-sns";
import { snsClient } from "../libs/snsClient.js";

/**
 * @param {string} subscriptionArn - The ARN of the subscription to cancel.
 */
const unsubscribe = async (
  subscriptionArn = "arn:aws:sns:us-east-1:xxxxxxxxxxxx:mytopic:xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxxx",
) => {
  const response = await snsClient.send(
    new UnsubscribeCommand({
      SubscriptionArn: subscriptionArn,
    }),
  );
  console.log(response);
  // {
  //   '$metadata': {
  //     httpStatusCode: 200,
  //     requestId: '0178259a-9204-507c-b620-78a7570a44c6',
  //     extendedRequestId: undefined,
  //     cfId: undefined,
  //     attempts: 1,
  //     totalRetryDelay: 0
  //   }
  // }
  return response;
};
```

- 자세한 정보는 AWS SDK for JavaScript 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for JavaScript API 참조의 Unsubscribe를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun unSub(subscriptionArnVal: String) {

    val request = UnsubscribeRequest {
        subscriptionArn = subscriptionArnVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for ${request.subscriptionArn}")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 Unsubscribe를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;
```

```
/**
 * Deletes a subscription to an Amazon SNS topic.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$subscription = 'arn:aws:sns:us-east-1:111122223333:MySubscription';

try {
    $result = $SnSclient->unsubscribe([
        'SubscriptionArn' => $subscription,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
    error_log($e->getMessage());
}
```

- 자세한 정보는 AWS SDK for PHP 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for PHP API 참조의 Unsubscribe를 참조하십시오.

Python

SDK for Python(Boto3)

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    @staticmethod
    def delete_subscription(subscription):
        """
        Unsubscribes and deletes a subscription.
        """
        try:
            subscription.delete()
            logger.info("Deleted subscription %s.", subscription.arn)
        except ClientError:
            logger.exception("Couldn't delete subscription %s.",
 subscription.arn)
            raise
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 Unsubscribe를 참조하십시오.

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
    TRY.
        lo_sns->unsubscribe( iv_subscriptionarn = iv_subscription_arn ).
        MESSAGE 'Subscription deleted.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
```

```
        MESSAGE 'Subscription does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snsinvalidparameterex.
        MESSAGE 'Subscription with "PendingConfirmation" status cannot be
deleted/unsubscribed. Confirm subscription before performing unsubscribe
operation.' TYPE 'E'.
    ENDTRY.
```

- API에 대한 세부 정보는 AWS SDK for SAP ABAP API 참조의 Unsubscribe를 참조하세요.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오AWS SDK와 함께 Amazon SNS 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하는 Amazon SNS의 시나리오

다음 코드 예제는 AWS SDK를 사용하여 Amazon SNS에서 일반적인 시나리오를 구현하는 방법을 보여줍니다. 이러한 시나리오에서는 Amazon SNS 내에서 여러 함수를 호출하여 특정 태스크를 수행하는 방법을 보여줍니다. 각 시나리오에는 코드 설정 및 실행 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. GitHub

예제

- AWS SDK를 사용하여 Amazon SNS 푸시 알림을 위한 플랫폼 엔드포인트를 생성합니다.
- SDK를 사용하여 FIFO Amazon SNS 주제를 만들고 게시하십시오. AWS
- AWS SDK를 사용하여 Amazon SNS 주제에 SMS 메시지를 게시합니다.
- AWS SDK를 사용하여 Amazon S3를 사용하여 Amazon SNS에 대용량 메시지를 게시합니다.
- AWS SDK를 사용하여 Amazon SNS SMS 문자 메시지를 게시합니다.
- SDK를 사용하여 Amazon SNS 메시지를 Amazon SQS 대기열에 게시합니다. AWS

AWS SDK를 사용하여 Amazon SNS 푸시 알림을 위한 플랫폼 엔드포인트를 생성합니다.

다음 코드 예제에서는 Amazon SNS 푸시 알림에 대한 플랫폼 엔드포인트를 생성하는 방법을 보여줍니다.

CLI

AWS CLI

플랫폼 애플리케이션 엔드포인트를 생성하려면

다음 `create-platform-endpoint` 예제에서는 지정된 토큰을 사용하여 지정된 플랫폼 애플리케이션의 엔드포인트를 생성합니다.

```
aws sns create-platform-endpoint \
    --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/GCM/
MyApplication \
    --token EXAMPLE12345...
```

출력:

```
{
      "EndpointArn": "arn:aws:sns:us-west-2:1234567890:endpoint/GCM/
MyApplication/12345678-abcd-9012-efgh-345678901234"
}
```

Java

SDK for Java 2.x

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointRequest;
import software.amazon.awssdk.services.sns.model.CreatePlatformEndpointResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * In addition, create a platform application using the AWS Management Console.
 * See this doc topic:
 *
 * https://docs.aws.amazon.com/sns/latest/dg/mobile-push-send-register.html
 *
 * Without the values created by following the previous link, this code examples
 * does not work.
 */

public class RegistrationExample {
    public static void main(String[] args) {
        final String usage = """

                Usage:     <token> <platformApplicationArn>

                Where:
                   token - The name of the FIFO topic.\s
                   platformApplicationArn - The ARN value of platform
 application. You can get this value from the AWS Management Console.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String token = args[0];
        String platformApplicationArn = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createEndpoint(snsClient, token, platformApplicationArn);
    }

    public static void createEndpoint(SnsClient snsClient, String token, String
 platformApplicationArn) {
        System.out.println("Creating platform endpoint with token " + token);
        try {
```

```
            CreatePlatformEndpointRequest endpointRequest =
 CreatePlatformEndpointRequest.builder()
                    .token(token)
                    .platformApplicationArn(platformApplicationArn)
                    .build();

            CreatePlatformEndpointResponse response =
 snsClient.createPlatformEndpoint(endpointRequest);
            System.out.println("The ARN of the endpoint is " +
 response.endpointArn());
        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오AWS SDK와 함께 Amazon SNS 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

SDK를 사용하여 FIFO Amazon SNS 주제를 만들고 게시하십시오. AWS

다음 코드 예제에서는 FIFO Amazon SNS 주제를 생성하고 거기에 게시하는 방법을 보여줍니다.

Java

SDK for Java 2.x

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

이 예에서는

- Amazon SNS FIFO 주제 1개, Amazon SQS FIFO 대기열 2개, 표준 대기열 1개를 생성합니다.
- 대기열에서 주제를 구독하고 해당 주제에 메시지를 게시합니다.

테스트에서는 각 대기열에 대한 메시지 수신 여부를 확인합니다. 또한 전체 예제에서는 액세스 정책을 추가하는 것을 보여주고 마지막에 리소스를 삭제합니다.

```
public class PriceUpdateExample {
    public final static SnsClient snsClient = SnsClient.create();
    public final static SqsClient sqsClient = SqsClient.create();

    public static void main(String[] args) {

        final String usage = "\n" +
                "Usage: " +
                "    <topicName> <wholesaleQueueFifoName> <retailQueueFifoName>
 <analyticsQueueName>\n\n" +
                "Where:\n" +
                "   fifoTopicName - The name of the FIFO topic that you want to
 create. \n\n" +
                "   wholesaleQueueARN - The name of a SQS FIFO queue that will be
 created for the wholesale consumer. \n\n"
                +
                "   retailQueueARN - The name of a SQS FIFO queue that will
 created for the retail consumer. \n\n" +
                "   analyticsQueueARN - The name of a SQS standard queue that
 will be created for the analytics consumer. \n\n";
        if (args.length != 4) {
            System.out.println(usage);
            System.exit(1);
        }

        final String fifoTopicName = args[0];
        final String wholeSaleQueueName = args[1];
        final String retailQueueName = args[2];
        final String analyticsQueueName = args[3];

        // For convenience, the QueueData class holds metadata about a queue:
 ARN, URL,
        // name and type.
        List<QueueData> queues = List.of(
                new QueueData(wholeSaleQueueName, QueueType.FIFO),
                new QueueData(retailQueueName, QueueType.FIFO),
                new QueueData(analyticsQueueName, QueueType.Standard));

        // Create queues.
        createQueues(queues);
```

```
        // Create a topic.
        String topicARN = createFIFOTopic(fifoTopicName);

        // Subscribe each queue to the topic.
        subscribeQueues(queues, topicARN);

        // Allow the newly created topic to send messages to the queues.
        addAccessPolicyToQueuesFINAL(queues, topicARN);

        // Publish a sample price update message with payload.
        publishPriceUpdate(topicARN, "{\"product\": 214, \"price\": 79.99}",
"Consumables");

        // Clean up resources.
        deleteSubscriptions(queues);
        deleteQueues(queues);
        deleteTopic(topicARN);
    }

    public static String createFIFOTopic(String topicName) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = Map.of(
                    "FifoTopic", "true",
                    "ContentBasedDeduplication", "false");

            CreateTopicRequest topicRequest = CreateTopicRequest.builder()
                    .name(topicName)
                    .attributes(topicAttributes)
                    .build();

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            String topicArn = response.topicArn();
            System.out.println("The topic ARN is" + topicArn);

            return topicArn;

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
```

```
   public static void subscribeQueues(List<QueueData> queues, String topicARN) {
       queues.forEach(queue -> {
           SubscribeRequest subscribeRequest = SubscribeRequest.builder()
                   .topicArn(topicARN)
                   .endpoint(queue.queueARN)
                   .protocol("sqs")
                   .build();

           // Subscribe to the endpoint by using the SNS service client.
           // Only Amazon SQS queues can receive notifications from an Amazon
SNS FIFO
           // topic.
           SubscribeResponse subscribeResponse =
snsClient.subscribe(subscribeRequest);
           System.out.println("The queue [" + queue.queueARN + "] subscribed to
the topic [" + topicARN + "]");
           queue.subscriptionARN = subscribeResponse.subscriptionArn();
       });
   }

   public static void publishPriceUpdate(String topicArn, String payload, String
groupId) {

       try {
           // Create and publish a message that updates the wholesale price.
           String subject = "Price Update";
           String dedupId = UUID.randomUUID().toString();
           String attributeName = "business";
           String attributeValue = "wholesale";

           MessageAttributeValue msgAttValue = MessageAttributeValue.builder()
                   .dataType("String")
                   .stringValue(attributeValue)
                   .build();

           Map<String, MessageAttributeValue> attributes = new HashMap<>();
           attributes.put(attributeName, msgAttValue);
           PublishRequest pubRequest = PublishRequest.builder()
                   .topicArn(topicArn)
                   .subject(subject)
                   .message(payload)
                   .messageGroupId(groupId)
                   .messageDeduplicationId(dedupId)
```

```
                    .messageAttributes(attributes)
                    .build();

            final PublishResponse response = snsClient.publish(pubRequest);
            System.out.println(response.messageId());
            System.out.println(response.sequenceNumber());
            System.out.println("Message was published to " + topicArn);

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 다음 항목을 참조하세요.
  - CreateTopic
  - 게시
  - Subscribe

Python

SDK for Python(Boto3)

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

Amazon FIFO 주제를 생성하고, Amazon SQS FIFO 및 표준 대기열에서 주제를 구독하고, 해당 주제에 메시지를 게시합니다.

```
def usage_demo():
    """Shows how to subscribe queues to a FIFO topic."""
    print("-" * 88)
    print("Welcome to the `Subscribe queues to a FIFO topic` demo!")
    print("-" * 88)

    sns = boto3.resource("sns")
```

```
    sqs = boto3.resource("sqs")
    fifo_topic_wrapper = FifoTopicWrapper(sns)
    sns_wrapper = SnsWrapper(sns)

    prefix = "sqs-subscribe-demo-"
    queues = set()
    subscriptions = set()

    wholesale_queue = sqs.create_queue(
        QueueName=prefix + "wholesale.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(wholesale_queue)
    print(f"Created FIFO queue with URL: {wholesale_queue.url}.")

    retail_queue = sqs.create_queue(
        QueueName=prefix + "retail.fifo",
        Attributes={
            "MaximumMessageSize": str(4096),
            "ReceiveMessageWaitTimeSeconds": str(10),
            "VisibilityTimeout": str(300),
            "FifoQueue": str(True),
            "ContentBasedDeduplication": str(True),
        },
    )
    queues.add(retail_queue)
    print(f"Created FIFO queue with URL: {retail_queue.url}.")

    analytics_queue = sqs.create_queue(QueueName=prefix + "analytics",
Attributes={})
    queues.add(analytics_queue)
    print(f"Created standard queue with URL: {analytics_queue.url}.")

    topic = fifo_topic_wrapper.create_fifo_topic("price-updates-topic.fifo")
    print(f"Created FIFO topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        fifo_topic_wrapper.add_access_policy(q, topic.attributes["TopicArn"])
```

```
    print(f"Added access policies for topic: {topic.attributes['TopicArn']}.")

    for q in queues:
        sub = fifo_topic_wrapper.subscribe_queue_to_topic(
            topic, q.attributes["QueueArn"]
        )
        subscriptions.add(sub)

    print(f"Subscribed queues to topic: {topic.attributes['TopicArn']}.")

    input("Press Enter to publish a message to the topic.")

    message_id = fifo_topic_wrapper.publish_price_update(
        topic, '{"product": 214, "price": 79.99}', "Consumables"
    )

    print(f"Published price update with message ID: {message_id}.")

    # Clean up the subscriptions, queues, and topic.
    input("Press Enter to clean up resources.")
    for s in subscriptions:
        sns_wrapper.delete_subscription(s)

    sns_wrapper.delete_topic(topic)

    for q in queues:
        fifo_topic_wrapper.delete_queue(q)

    print(f"Deleted subscriptions, queues, and topic.")

    print("Thanks for watching!")
    print("-" * 88)



class FifoTopicWrapper:
    """Encapsulates Amazon SNS FIFO topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource
```

```
    def create_fifo_topic(self, topic_name):
        """
        Create a FIFO topic.
        Topic names must be made up of only uppercase and lowercase ASCII
letters,
        numbers, underscores, and hyphens, and must be between 1 and 256
characters long.
        For a FIFO topic, the name must end with the .fifo suffix.

        :param topic_name: The name for the topic.
        :return: The new topic.
        """
        try:
            topic = self.sns_resource.create_topic(
                Name=topic_name,
                Attributes={
                    "FifoTopic": str(True),
                    "ContentBasedDeduplication": str(False),
                },
            )
            logger.info("Created FIFO topic with name=%s.", topic_name)
            return topic
        except ClientError as error:
            logger.exception("Couldn't create topic with name=%s!", topic_name)
            raise error


    @staticmethod
    def add_access_policy(queue, topic_arn):
        """
        Add the necessary access policy to a queue, so
        it can receive messages from a topic.

        :param queue: The queue resource.
        :param topic_arn: The ARN of the topic.
        :return: None.
        """
        try:
            queue.set_attributes(
                Attributes={
                    "Policy": json.dumps(
                        {
                            "Version": "2012-10-17",
```

```
                        "Statement": [
                            {
                                "Sid": "test-sid",
                                "Effect": "Allow",
                                "Principal": {"AWS": "*"},
                                "Action": "SQS:SendMessage",
                                "Resource": queue.attributes["QueueArn"],
                                "Condition": {
                                    "ArnLike": {"aws:SourceArn": topic_arn}
                                },
                            }
                        ],
                    }
                )
            }
        )
        logger.info("Added trust policy to the queue.")
    except ClientError as error:
        logger.exception("Couldn't add trust policy to the queue!")
        raise error


@staticmethod
def subscribe_queue_to_topic(topic, queue_arn):
    """
    Subscribe a queue to a topic.

    :param topic: The topic resource.
    :param queue_arn: The ARN of the queue.
    :return: The subscription resource.
    """
    try:
        subscription = topic.subscribe(
            Protocol="sqs",
            Endpoint=queue_arn,
        )
        logger.info("The queue is subscribed to the topic.")
        return subscription
    except ClientError as error:
        logger.exception("Couldn't subscribe queue to topic!")
        raise error


@staticmethod
```

```
    def publish_price_update(topic, payload, group_id):
        """
        Compose and publish a message that updates the wholesale price.

        :param topic: The topic to publish to.
        :param payload: The message to publish.
        :param group_id: The group ID for the message.
        :return: The ID of the message.
        """
        try:
            att_dict = {"business": {"DataType": "String", "StringValue":
"wholesale"}}
            dedup_id = uuid.uuid4()
            response = topic.publish(
                Subject="Price Update",
                Message=payload,
                MessageAttributes=att_dict,
                MessageGroupId=group_id,
                MessageDeduplicationId=str(dedup_id),
            )
            message_id = response["MessageId"]
            logger.info("Published message to topic %s.", topic.arn)
        except ClientError as error:
            logger.exception("Couldn't publish message to topic %s.", topic.arn)
            raise error
        return message_id


    @staticmethod
    def delete_queue(queue):
        """
        Removes an SQS queue. When run against an AWS account, it can take up to
        60 seconds before the queue is actually deleted.

        :param queue: The queue to delete.
        :return: None
        """
        try:
            queue.delete()
            logger.info("Deleted queue with URL=%s.", queue.url)
        except ClientError as error:
            logger.exception("Couldn't delete queue with URL=%s!", queue.url)
            raise error
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 다음 주제를 참조하십시오.
  - CreateTopic
  - 게시
  - Subscribe

SAP ABAP

SDK for SAP ABAP

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

FIFO 주제를 생성하고, 주제에 대한 Amazon SQS FIFO 대기열을 구독하고, Amazon SNS 주제에 메시지를 게시합니다.

```
    " Creates a FIFO topic. "
    DATA lt_tpc_attributes TYPE /aws1/
cl_snstopicattrsmap_w=>tt_topicattributesmap.
    DATA ls_tpc_attributes TYPE /aws1/
cl_snstopicattrsmap_w=>ts_topicattributesmap_maprow.
    ls_tpc_attributes-key = 'FifoTopic'.
    ls_tpc_attributes-value = NEW /aws1/cl_snstopicattrsmap_w( iv_value =
 'true' ).
    INSERT ls_tpc_attributes INTO TABLE lt_tpc_attributes.

    TRY.
        DATA(lo_create_result) = lo_sns->createtopic(
               iv_name = iv_topic_name
               it_attributes = lt_tpc_attributes
        ).
        DATA(lv_topic_arn) = lo_create_result->get_topicarn( ).
```

```
        ov_topic_arn = lv_topic_arn.                                    "
 ov_topic_arn is returned for testing purposes. "
        MESSAGE 'FIFO topic created' TYPE 'I'.
      CATCH /aws1/cx_snstopiclimitexcdex.
        MESSAGE 'Unable to create more topics. You have reached the maximum
 number of topics allowed.' TYPE 'E'.
    ENDTRY.

    " Subscribes an endpoint to an Amazon Simple Notification Service (Amazon
 SNS) topic. "
    " Only Amazon Simple Queue Service (Amazon SQS) FIFO queues can be subscribed
 to an SNS FIFO topic. "
    TRY.
        DATA(lo_subscribe_result) = lo_sns->subscribe(
               iv_topicarn = lv_topic_arn
               iv_protocol = 'sqs'
               iv_endpoint = iv_queue_arn
           ).
        DATA(lv_subscription_arn) = lo_subscribe_result->get_subscriptionarn( ).
        ov_subscription_arn = lv_subscription_arn.                      "
 ov_subscription_arn is returned for testing purposes. "
        MESSAGE 'SQS queue was subscribed to SNS topic.' TYPE 'I'.
      CATCH /aws1/cx_snsnotfoundexception.
        MESSAGE 'Topic does not exist.' TYPE 'E'.
      CATCH /aws1/cx_snssubscriptionlmte00.
        MESSAGE 'Unable to create subscriptions. You have reached the maximum
 number of subscriptions allowed.' TYPE 'E'.
    ENDTRY.

    " Publish message to SNS topic. "
    TRY.
        DATA lt_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>tt_messageattributemap.
        DATA ls_msg_attributes TYPE /aws1/
cl_snsmessageattrvalue=>ts_messageattributemap_maprow.
        ls_msg_attributes-key = 'Importance'.
        ls_msg_attributes-value = NEW /aws1/cl_snsmessageattrvalue( iv_datatype =
 'String' iv_stringvalue = 'High' ).
        INSERT ls_msg_attributes INTO TABLE lt_msg_attributes.

        DATA(lo_result) = lo_sns->publish(
             iv_topicarn = lv_topic_arn
             iv_message = 'The price of your mobile plan has been increased from
 $19 to $23'
```

```
            iv_subject = 'Changes to mobile plan'
            iv_messagegroupid = 'Update-2'
            iv_messagededuplicationid = 'Update-2.1'
            it_messageattributes = lt_msg_attributes
        ).
          ov_message_id = lo_result->get_messageid( ).                    "
ov_message_id is returned for testing purposes. "
          MESSAGE 'Message was published to SNS topic.' TYPE 'I'.
        CATCH /aws1/cx_snsnotfoundexception.
          MESSAGE 'Topic does not exist.' TYPE 'E'.
      ENDTRY.
```

- API 세부 정보는 AWS SDK for SAP ABAP API 참조의 다음 주제를 참조하세요.
  - CreateTopic
  - 게시
  - Subscribe

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오AWS SDK와 함께 Amazon SNS 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 Amazon SNS 주제에 SMS 메시지를 게시합니다.

다음 코드 예제에서는 작업 방법을 보여줍니다.

- Amazon SNS 주제를 생성합니다.
- 전화번호를 주제에 구독시킵니다.
- 모든 구독 전화번호가 한 번에 메시지를 받을 수 있도록 주제에 SMS 메시지를 게시합니다.

Java

SDK for Java 2.x

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

주제를 만들고 해당 ARN을 반환합니다.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.CreateTopicRequest;
import software.amazon.awssdk.services.sns.model.CreateTopicResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateTopic {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicName>

                Where:
                   topicName - The name of the topic to create (for example,
 mytopic).

                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String topicName = args[0];
        System.out.println("Creating a topic with name: " + topicName);
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String arnVal = createSNSTopic(snsClient, topicName);
        System.out.println("The topic ARN is" + arnVal);
        snsClient.close();
    }

    public static String createSNSTopic(SnsClient snsClient, String topicName) {
        CreateTopicResponse result;
        try {
            CreateTopicRequest request = CreateTopicRequest.builder()
                    .name(topicName)
                    .build();

            result = snsClient.createTopic(request);
            return result.topicArn();

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

주제에 엔드포인트를 구독 설정합니다.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
```

```
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SubscribeTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <topicArn> <phoneNumber>

                Where:
                   topicArn - The ARN of the topic to subscribe.
                   phoneNumber - A mobile phone number that receives
 notifications (for example, +1XXX5550100).
                """;

        if (args.length < 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String topicArn = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();

        subTextSNS(snsClient, topicArn, phoneNumber);
        snsClient.close();
    }

    public static void subTextSNS(SnsClient snsClient, String topicArn, String
 phoneNumber) {
        try {
            SubscribeRequest request = SubscribeRequest.builder()
                    .protocol("sms")
                    .endpoint(phoneNumber)
                    .returnSubscriptionArn(true)
                    .topicArn(topicArn)
                    .build();

            SubscribeResponse result = snsClient.subscribe(request);
```

```
            System.out.println("Subscription ARN: " + result.subscriptionArn() +
 "\n\n Status is "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

발신자의 ID, 최고 가격 및 유형과 같은 메시지의 속성을 설정합니다. 메시지 속성은 선택 사항입니다.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesRequest;
import software.amazon.awssdk.services.sns.model.SetSmsAttributesResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
import java.util.HashMap;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class SetSMSAttributes {
    public static void main(String[] args) {
        HashMap<String, String> attributes = new HashMap<>(1);
        attributes.put("DefaultSMSType", "Transactional");
        attributes.put("UsageReportS3Bucket", "janbucket");

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        setSNSAttributes(snsClient, attributes);
        snsClient.close();
    }
```

```
    public static void setSNSAttributes(SnsClient snsClient, HashMap<String,
 String> attributes) {
        try {
            SetSmsAttributesRequest request = SetSmsAttributesRequest.builder()
                    .attributes(attributes)
                    .build();

            SetSmsAttributesResponse result =
 snsClient.setSMSAttributes(request);
            System.out.println("Set default Attributes to " + attributes + ".
 Status was "
                    + result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

주제에 메시지를 게시합니다. 메시지는 모든 구독자에게 전송됩니다.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = """
```

```
                Usage:    <message> <phoneNumber>

                Where:
                   message - The message text to send.
                   phoneNumber - The mobile phone number to which a message is
 sent (for example, +1XXX5550100).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }

    public static void pubTextSMS(SnsClient snsClient, String message, String
 phoneNumber) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .phoneNumber(phoneNumber)
                    .build();

            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오AWS SDK와 함께 Amazon SNS 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 Amazon S3를 사용하여 Amazon SNS에 대용량 메시지를 게시합니다.

다음 코드 예제에서는 Amazon S3를 사용하여 메시지 페이로드를 저장하는 대량 메시지를 Amazon SNS에 게시하는 방법을 보여줍니다.

Java

SDK for Java 1.x

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

대용량 메시지를 게시하려면 Java용 Amazon SNS 확장 클라이언트 라이브러리를 사용하세요. 보내는 메시지는 실제 메시지 내용이 포함된 Amazon S3 객체를 참조합니다.

```
import com.amazon.sqs.javamessaging.AmazonSQSExtendedClient;
import com.amazon.sqs.javamessaging.ExtendedClientConfiguration;
import com.amazonaws.regions.Region;
import com.amazonaws.regions.Regions;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3ClientBuilder;
import com.amazonaws.services.sns.AmazonSNS;
import com.amazonaws.services.sns.AmazonSNSClientBuilder;
import com.amazonaws.services.sns.model.CreateTopicRequest;
import com.amazonaws.services.sns.model.PublishRequest;
import com.amazonaws.services.sns.model.SetSubscriptionAttributesRequest;
import com.amazonaws.services.sns.util.Topics;
import com.amazonaws.services.sqs.AmazonSQS;
import com.amazonaws.services.sqs.AmazonSQSClientBuilder;
import com.amazonaws.services.sqs.model.CreateQueueRequest;
import com.amazonaws.services.sqs.model.ReceiveMessageResult;
import software.amazon.sns.AmazonSNSExtendedClient;
import software.amazon.sns.SNSExtendedClientConfiguration;
```

```
public class Example {

        public static void main(String[] args) {
                final String BUCKET_NAME = "extended-client-bucket";
                final String TOPIC_NAME = "extended-client-topic";
                final String QUEUE_NAME = "extended-client-queue";
                final Regions region = Regions.DEFAULT_REGION;

                // Message threshold controls the maximum message size that will
 be allowed to
                // be published
                // through SNS using the extended client. Payload of messages
 exceeding this
                // value will be stored in
                // S3. The default value of this parameter is 256 KB which is the
 maximum
                // message size in SNS (and SQS).
                final int EXTENDED_STORAGE_MESSAGE_SIZE_THRESHOLD = 32;

                // Initialize SNS, SQS and S3 clients
                final AmazonSNS snsClient =
 AmazonSNSClientBuilder.standard().withRegion(region).build();
                final AmazonSQS sqsClient =
 AmazonSQSClientBuilder.standard().withRegion(region).build();
                final AmazonS3 s3Client =
 AmazonS3ClientBuilder.standard().withRegion(region).build();

                // Create bucket, topic, queue and subscription
                s3Client.createBucket(BUCKET_NAME);
                final String topicArn = snsClient.createTopic(
                                new
 CreateTopicRequest().withName(TOPIC_NAME)).getTopicArn();
                final String queueUrl = sqsClient.createQueue(
                                new
 CreateQueueRequest().withQueueName(QUEUE_NAME)).getQueueUrl();
                final String subscriptionArn = Topics.subscribeQueue(
                                snsClient, sqsClient, topicArn, queueUrl);

                // To read message content stored in S3 transparently through SQS
 extended
                // client,
                // set the RawMessageDelivery subscription attribute to TRUE
```

```
                final SetSubscriptionAttributesRequest
subscriptionAttributesRequest = new SetSubscriptionAttributesRequest();

subscriptionAttributesRequest.setSubscriptionArn(subscriptionArn);

subscriptionAttributesRequest.setAttributeName("RawMessageDelivery");
                subscriptionAttributesRequest.setAttributeValue("TRUE");

snsClient.setSubscriptionAttributes(subscriptionAttributesRequest);

                // Initialize SNS extended client
                // PayloadSizeThreshold triggers message content storage in S3
when the
                // threshold is exceeded
                // To store all messages content in S3, use AlwaysThroughS3 flag
                final SNSExtendedClientConfiguration
snsExtendedClientConfiguration = new SNSExtendedClientConfiguration()
                                .withPayloadSupportEnabled(s3Client, BUCKET_NAME)

.withPayloadSizeThreshold(EXTENDED_STORAGE_MESSAGE_SIZE_THRESHOLD);
                final AmazonSNSExtendedClient snsExtendedClient = new
AmazonSNSExtendedClient(snsClient,
                                snsExtendedClientConfiguration);

                // Publish message via SNS with storage in S3
                final String message = "This message is stored in S3 as it
exceeds the threshold of 32 bytes set above.";
                snsExtendedClient.publish(topicArn, message);

                // Initialize SQS extended client
                final ExtendedClientConfiguration sqsExtendedClientConfiguration
= new ExtendedClientConfiguration()
                                .withPayloadSupportEnabled(s3Client,
BUCKET_NAME);
                final AmazonSQSExtendedClient sqsExtendedClient = new
AmazonSQSExtendedClient(sqsClient,
                                sqsExtendedClientConfiguration);

                // Read the message from the queue
                final ReceiveMessageResult result =
sqsExtendedClient.receiveMessage(queueUrl);
                System.out.println("Received message is " +
result.getMessages().get(0).getBody());
        }
```

```
}
```

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오AWS SDK와 함께 Amazon SNS 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

AWS SDK를 사용하여 Amazon SNS SMS 문자 메시지를 게시합니다.

다음 코드 예제에서는 Amazon SNS를 사용하여 SMS 메시지를 게시하는 방법을 보여줍니다.

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
namespace SNSMessageExample
{
    using System;
    using System.Threading.Tasks;
    using Amazon;
    using Amazon.SimpleNotificationService;
    using Amazon.SimpleNotificationService.Model;

    public class SNSMessage
    {
        private AmazonSimpleNotificationServiceClient snsClient;

        /// <summary>
        /// Initializes a new instance of the <see cref="SNSMessage"/> class.
        /// Constructs a new SNSMessage object initializing the Amazon Simple
        /// Notification Service (Amazon SNS) client using the supplied
        /// Region endpoint.
        /// </summary>
        /// <param name="regionEndpoint">The Amazon Region endpoint to use in
        /// sending test messages with this object.</param>
```

```
        public SNSMessage(RegionEndpoint regionEndpoint)
        {
            snsClient = new
 AmazonSimpleNotificationServiceClient(regionEndpoint);
        }

        /// <summary>
        /// Sends the SMS message passed in the text parameter to the phone
 number
        /// in phoneNum.
        /// </summary>
        /// <param name="phoneNum">The ten-digit phone number to which the text
        /// message will be sent.</param>
        /// <param name="text">The text of the message to send.</param>
        /// <returns>Async task.</returns>
        public async Task SendTextMessageAsync(string phoneNum, string text)
        {
            if (string.IsNullOrEmpty(phoneNum) || string.IsNullOrEmpty(text))
            {
                return;
            }

            // Now actually send the message.
            var request = new PublishRequest
            {
                Message = text,
                PhoneNumber = phoneNum,
            };

            try
            {
                var response = await snsClient.PublishAsync(request);
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Error sending message: {ex}");
            }
        }
    }
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 Publish를 참조하십시오.

C++

SDK for C++

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
/**
 * Publish SMS: use Amazon Simple Notification Service (Amazon SNS) to send an
 SMS text message to a phone number.
 * Note: This requires additional AWS configuration prior to running example.
 *
 *  NOTE: When you start using Amazon SNS to send SMS messages, your AWS account
 is in the SMS sandbox and you can only
 *  use verified destination phone numbers. See https://docs.aws.amazon.com/sns/
latest/dg/sns-sms-sandbox.html.
 *  NOTE: If destination is in the US, you also have an additional restriction
 that you have use a dedicated
 *  origination ID (phone number). You can request an origination number using
 Amazon Pinpoint for a fee.
 *  See https://aws.amazon.com/blogs/compute/provisioning-and-using-10dlc-
origination-numbers-with-amazon-sns/
 *  for more information.
 *
 *  <phone_number_value> input parameter uses E.164 format.
 *  For example, in United States, this input value should be of the form:
 +12223334444
 */

//! Send an SMS text message to a phone number.
/*!
  \param message: The message to publish.
  \param phoneNumber: The phone number of the recipient in E.164 format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::publishSms(const Aws::String &message,
                             const Aws::String &phoneNumber,
```

```
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetPhoneNumber(phoneNumber);

    const Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);

    if (outcome.IsSuccess()) {
        std::cout << "Message published successfully with message id, '"
                  << outcome.GetResult().GetMessageId() << "'."
                  << std::endl;
    }
    else {
        std::cerr << "Error while publishing message "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 Publish를 참조하십시오.

Java

SDK for Java 2.x

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;
import software.amazon.awssdk.services.sns.model.SnsException;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class PublishTextSMS {
    public static void main(String[] args) {
        final String usage = """

                Usage:    <message> <phoneNumber>

                Where:
                   message - The message text to send.
                   phoneNumber - The mobile phone number to which a message is
 sent (for example, +1XXX5550100).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String message = args[0];
        String phoneNumber = args[1];
        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)
                .build();
        pubTextSMS(snsClient, message, phoneNumber);
        snsClient.close();
    }

    public static void pubTextSMS(SnsClient snsClient, String message, String
 phoneNumber) {
        try {
            PublishRequest request = PublishRequest.builder()
                    .message(message)
                    .phoneNumber(phoneNumber)
                    .build();
```

```
            PublishResponse result = snsClient.publish(request);
            System.out
                    .println(result.messageId() + " Message sent. Status was " +
 result.sdkHttpResponse().statusCode());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- API 세부 정보는 AWS SDK for Java 2.x API 참조의 Publish를 참조하십시오.

Kotlin

SDK for Kotlin

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
suspend fun pubTextSMS(messageVal: String?, phoneNumberVal: String?) {

    val request = PublishRequest {
        message = messageVal
        phoneNumber = phoneNumberVal
    }

    SnsClient { region = "us-east-1" }.use { snsClient ->
        val result = snsClient.publish(request)
        println("${result.messageId} message sent.")
    }
}
```

- API 세부 정보는 AWS SDK for Kotlin API 참조의 Publish를 참조하십시오.

PHP

SDK for PHP

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
require 'vendor/autoload.php';

use Aws\Exception\AwsException;
use Aws\Sns\SnsClient;


/**
 * Sends a text message (SMS message) directly to a phone number using Amazon
 SNS.
 *
 * This code expects that you have AWS credentials set up per:
 * https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/
guide_credentials.html
 */

$SnSclient = new SnsClient([
    'profile' => 'default',
    'region' => 'us-east-1',
    'version' => '2010-03-31'
]);

$message = 'This message is sent from a Amazon SNS code sample.';
$phone = '+1XXX5550100';

try {
    $result = $SnSclient->publish([
        'Message' => $message,
        'PhoneNumber' => $phone,
    ]);
    var_dump($result);
} catch (AwsException $e) {
    // output error message if fails
```

```
    error_log($e->getMessage());
}
```

- 자세한 정보는 AWS SDK for PHP 개발자 안내서를 참조하십시오.
- API 세부 정보는 AWS SDK for PHP API 참조의 Publish를 참조하십시오.

Python

SDK for Python(Boto3)

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
class SnsWrapper:
    """Encapsulates Amazon SNS topic and subscription functions."""

    def __init__(self, sns_resource):
        """
        :param sns_resource: A Boto3 Amazon SNS resource.
        """
        self.sns_resource = sns_resource


    def publish_text_message(self, phone_number, message):
        """
        Publishes a text message directly to a phone number without need for a
        subscription.

        :param phone_number: The phone number that receives the message. This
 must be
                             in E.164 format. For example, a United States phone
                             number might be +12065550101.
        :param message: The message to send.
        :return: The ID of the message.
        """
        try:
```

```
            response = self.sns_resource.meta.client.publish(
                PhoneNumber=phone_number, Message=message
            )
            message_id = response["MessageId"]
            logger.info("Published message to %s.", phone_number)
        except ClientError:
            logger.exception("Couldn't publish message to %s.", phone_number)
            raise
        else:
            return message_id
```

- API 세부 정보는 AWS SDK for Python (Boto3) API 참조의 Publish를 참조하십시오.

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오AWS SDK와 함께 Amazon SNS 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

SDK를 사용하여 Amazon SNS 메시지를 Amazon SQS 대기열에 게시합니다. AWS

다음 코드 예제는 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.

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
/// <summary>
/// Console application to run a workflow scenario for topics and queues.
/// </summary>
public static class TopicsAndQueues
{
    private static bool _useFifoTopic = false;
    private static bool _useContentBasedDeduplication = false;
    private static string _topicName = null!;
    private static string _topicArn = null!;

    private static readonly int _queueCount = 2;
    private static readonly string[] _queueUrls = new string[_queueCount];
    private static readonly string[] _subscriptionArns = new string[_queueCount];
    private static readonly string[] _tones = { "cheerful", "funny", "serious",
 "sincere" };
    public static SNSWrapper SnsWrapper { get; set; } = null!;
    public static SQSWrapper SqsWrapper { get; set; } = null!;
    public static bool UseConsole { get; set; } = true;
    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon EventBridge.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonSQS>()
                    .AddAWSService<IAmazonSimpleNotificationService>()
                    .AddTransient<SNSWrapper>()
                    .AddTransient<SQSWrapper>()
            )
            .Build();

        ServicesSetup(host);
        PrintDescription();

        await RunScenario();

    }
```

```
    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        SnsWrapper = host.Services.GetRequiredService<SNSWrapper>();
        SqsWrapper = host.Services.GetRequiredService<SQSWrapper>();
    }

    /// <summary>
    /// Run the scenario for working with topics and queues.
    /// </summary>
    /// <returns>True if successful.</returns>
    public static async Task<bool> RunScenario()
    {
        try
        {
            await SetupTopic();

            await SetupQueues();

            await PublishMessages();

            foreach (var queueUrl in _queueUrls)
            {
                var messages = await PollForMessages(queueUrl);
                if (messages.Any())
                {
                    await DeleteMessages(queueUrl, messages);
                }
            }
            await CleanupResources();

            Console.WriteLine("Messaging with topics and queues workflow is
complete.");
            return true;
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
```

```
            await CleanupResources();
            Console.WriteLine(new string('-', 80));
            return false;
        }
    }

    /// <summary>
    /// Print a description for the tasks in the workflow.
    /// </summary>
    /// <returns>Async task.</returns>
    private static void PrintDescription()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Welcome to messaging with topics and queues.");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"In this workflow, you will create an SNS topic and
 subscribe {_queueCount} SQS queues to the topic." +
                          $"\r\nYou can select from several options for
 configuring the topic and the subscriptions for the 2 queues." +
                          $"\r\nYou can then post to the topic and see the
 results in the queues.\r\n");

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up the SNS topic to be used with the queues.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task<string> SetupTopic()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"SNS topics can be configured as FIFO (First-In-First-
Out)." +
                          $"\r\nFIFO topics deliver messages in order and support
 deduplication and message filtering." +
                          $"\r\nYou can then post to the topic and see the
 results in the queues.\r\n");

        _useFifoTopic = GetYesNoResponse("Would you like to work with FIFO
 topics?");

        if (_useFifoTopic)
```

```
        {
            Console.WriteLine(new string('-', 80));
            _topicName = GetUserResponse("Enter a name for your SNS topic: ",
 "example-topic");
            Console.WriteLine(
                "Because you have selected a FIFO topic, '.fifo' must be appended
 to the topic name.\r\n");

            Console.WriteLine(new string('-', 80));
            Console.WriteLine($"Because you have chosen a FIFO topic,
 deduplication is supported." +
                              $"\r\nDeduplication IDs are either set in the
 message or automatically generated " +
                              $"\r\nfrom content using a hash function.\r\n" +
                              $"\r\nIf a message is successfully published to an
 SNS FIFO topic, any message " +
                              $"\r\npublished and determined to have the same
 deduplication ID, " +
                              $"\r\nwithin the five-minute deduplication
 interval, is accepted but not delivered.\r\n" +
                              $"\r\nFor more information about deduplication, " +
                              $"\r\nsee https://docs.aws.amazon.com/sns/latest/
dg/fifo-message-dedup.html.");

            _useContentBasedDeduplication = GetYesNoResponse("Use content-based
 deduplication instead of entering a deduplication ID?");
            Console.WriteLine(new string('-', 80));
        }

        _topicArn = await SnsWrapper.CreateTopicWithName(_topicName,
 _useFifoTopic, _useContentBasedDeduplication);

        Console.WriteLine($"Your new topic with the name {_topicName}" +
                          $"\r\nand Amazon Resource Name (ARN) {_topicArn}" +
                          $"\r\nhas been created.\r\n");

        Console.WriteLine(new string('-', 80));
        return _topicArn;
    }

    /// <summary>
    /// Set up the queues.
    /// </summary>
    /// <returns>Async task.</returns>
```

```
    private static async Task SetupQueues()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Now you will create {_queueCount} Amazon Simple Queue
Service (Amazon SQS) queues to subscribe to the topic.");

        // Repeat this section for each queue.
        for (int i = 0; i < _queueCount; i++)
        {
            var queueName = GetUserResponse("Enter a name for an Amazon SQS
queue: ", $"example-queue-{i}");
            if (_useFifoTopic)
            {
                // Only explain this once.
                if (i == 0)
                {
                    Console.WriteLine(
                        "Because you have selected a FIFO topic, '.fifo' must be
appended to the queue name.");
                }

                var queueUrl = await SqsWrapper.CreateQueueWithName(queueName,
_useFifoTopic);

                _queueUrls[i] = queueUrl;

                Console.WriteLine($"Your new queue with the name {queueName}" +
                                  $"\r\nand queue URL {queueUrl}" +
                                  $"\r\nhas been created.\r\n");

                if (i == 0)
                {
                    Console.WriteLine(
                        $"The queue URL is used to retrieve the queue ARN,\r\n" +
                        $"which is used to create a subscription.");
                    Console.WriteLine(new string('-', 80));
                }

                var queueArn = await SqsWrapper.GetQueueArnByUrl(queueUrl);

                if (i == 0)
                {
                    Console.WriteLine(
```

```
                        $"An AWS Identity and Access Management (IAM) policy must
 be attached to an SQS queue, enabling it to receive\r\n" +
                        $"messages from an SNS topic");
                }

                await SqsWrapper.SetQueuePolicyForTopic(queueArn, _topicArn,
 queueUrl);

                await SetupFilters(i, queueArn, queueName);
            }
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Set up filters with user options for a queue.
    /// </summary>
    /// <param name="queueCount">The number of this queue.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="queueName">The name of the queue.</param>
    /// <returns>Async Task.</returns>
    public static async Task SetupFilters(int queueCount, string queueArn, string
 queueName)
    {
        if (_useFifoTopic)
        {
            Console.WriteLine(new string('-', 80));
            // Only explain this once.
            if (queueCount == 0)
            {
                Console.WriteLine(
                    "Subscriptions to a FIFO topic can have filters." +
                    "If you add a filter to this subscription, then only the
 filtered messages " +
                    "will be received in the queue.");

                Console.WriteLine(
                    "For information about message filtering, " +
                    "see https://docs.aws.amazon.com/sns/latest/dg/sns-message-
filtering.html");

                Console.WriteLine(
                    "For this example, you can filter messages by a" +
```

```
                    "TONE attribute.");
            }

            var useFilter = GetYesNoResponse($"Filter messages for {queueName}'s
subscription to the topic?");

            string? filterPolicy = null;
            if (useFilter)
            {
                filterPolicy = CreateFilterPolicy();
            }
            var subscriptionArn = await
SnsWrapper.SubscribeTopicWithFilter(_topicArn, filterPolicy,
                queueArn);
            _subscriptionArns[queueCount] = subscriptionArn;

            Console.WriteLine(
                $"The queue {queueName} has been subscribed to the topic
{_topicName} " +
                $"with the subscription ARN {subscriptionArn}");
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Use user input to create a filter policy for a subscription.
    /// </summary>
    /// <returns>The serialized filter policy.</returns>
    public static string CreateFilterPolicy()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            $"You can filter messages by one or more of the following" +
            $"TONE attributes.");

        List<string> filterSelections = new List<string>();

        var selectionNumber = 0;
        do
        {
            Console.WriteLine(
                $"Enter a number to add a TONE filter, or enter 0 to stop adding
filters.");
            for (int i = 0; i < _tones.Length; i++)
```

```
            {
                Console.WriteLine($"\t{i + 1}. {_tones[i]}");
            }

            var selection = GetUserResponse("", filterSelections.Any() ? "0" :
 "1");
            int.TryParse(selection, out selectionNumber);
            if (selectionNumber > 0 && !
filterSelections.Contains(_tones[selectionNumber - 1]))
            {
                filterSelections.Add(_tones[selectionNumber - 1]);
            }
        } while (selectionNumber != 0);

        var filters = new Dictionary<string, List<string>>
        {
            { "tone", filterSelections }
        };
        string filterPolicy = JsonSerializer.Serialize(filters);
        return filterPolicy;
    }

    /// <summary>
    /// Publish messages using user settings.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task PublishMessages()
    {
        Console.WriteLine("Now we can publish messages.");

        var keepSendingMessages = true;
        string? deduplicationId = null;
        string? toneAttribute = null;
        while (keepSendingMessages)
        {
            Console.WriteLine();
            var message = GetUserResponse("Enter a message to publish.", "This is
 a sample message");

            if (_useFifoTopic)
            {
                Console.WriteLine("Because you are using a FIFO topic, you must
 set a message group ID." +
```

```
                                    "\r\nAll messages within the same group will be
received in the order " +
                                    "they were published.");

                Console.WriteLine();
                var messageGroupId = GetUserResponse("Enter a message group ID
for this message:", "1");

                if (!_useContentBasedDeduplication)
                {
                    Console.WriteLine("Because you are not using content-based
deduplication, " +
                                      "you must enter a deduplication ID.");

                    Console.WriteLine("Enter a deduplication ID for this
message.");
                    deduplicationId = GetUserResponse("Enter a deduplication ID
for this message.", "1");
                }

                if (GetYesNoResponse("Add an attribute to this message?"))
                {
                    Console.WriteLine("Enter a number for an attribute.");
                    for (int i = 0; i < _tones.Length; i++)
                    {
                        Console.WriteLine($"\t{i + 1}. {_tones[i]}");
                    }

                    var selection = GetUserResponse("", "1");
                    int.TryParse(selection, out var selectionNumber);

                    if (selectionNumber > 0 && selectionNumber < _tones.Length)
                    {
                        toneAttribute = _tones[selectionNumber - 1];
                    }
                }

                var messageID = await SnsWrapper.PublishToTopicWithAttribute(
                    _topicArn, message, "tone", toneAttribute, deduplicationId,
messageGroupId);

                Console.WriteLine($"Message published with id {messageID}.");
            }
```

```
            keepSendingMessages = GetYesNoResponse("Send another message?",
false);
        }
    }

    /// <summary>
    /// Poll for the published messages to see the results of the user's choices.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task<List<Message>> PollForMessages(string queueUrl)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Now the SQS queue at {queueUrl} will be polled to
retrieve the messages." +
                          "\r\nPress any key to continue.");
        if (UseConsole)
        {
            Console.ReadLine();
        }

        var moreMessages = true;
        var messages = new List<Message>();
        while (moreMessages)
        {
            var newMessages = await SqsWrapper.ReceiveMessagesByUrl(queueUrl,
10);

            moreMessages = newMessages.Any();
            if (moreMessages)
            {
                messages.AddRange(newMessages);
            }
        }

        Console.WriteLine($"{messages.Count} message(s) were received by the
queue at {queueUrl}.");

        foreach (var message in messages)
        {
            Console.WriteLine("\tMessage:" +
                              $"\n\t{message.Body}");
        }

        Console.WriteLine(new string('-', 80));
```

```
        return messages;
    }

    /// <summary>
    /// Delete the message using handles in a batch.
    /// </summary>
    /// <returns>Async task.</returns>
    public static async Task DeleteMessages(string queueUrl, List<Message>
messages)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Now we can delete the messages in this queue in a
batch.");
        await SqsWrapper.DeleteMessageBatchByUrl(queueUrl, messages);
        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <returns>Async task.</returns>
    private static async Task CleanupResources()
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine($"Clean up resources.");

        try
        {
            foreach (var queueUrl in _queueUrls)
            {
                if (!string.IsNullOrEmpty(queueUrl))
                {
                    var deleteQueue =
                        GetYesNoResponse($"Delete queue with url {queueUrl}?");
                    if (deleteQueue)
                    {
                        await SqsWrapper.DeleteQueueByUrl(queueUrl);
                    }
                }
            }

            foreach (var subscriptionArn in _subscriptionArns)
            {
                if (!string.IsNullOrEmpty(subscriptionArn))
```

```
                {
                    await SnsWrapper.UnsubscribeByArn(subscriptionArn);
                }
            }

            var deleteTopic = GetYesNoResponse($"Delete topic {_topicName}?");
            if (deleteTopic)
            {
                await SnsWrapper.DeleteTopicByArn(_topicArn);
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine($"Unable to clean up resources. Here's why:
 {ex.Message}.");
        }

        Console.WriteLine(new string('-', 80));
    }

    /// <summary>
    /// Helper method to get a yes or no response from the user.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static bool GetYesNoResponse(string question, bool defaultAnswer =
 true)
    {
        if (UseConsole)
        {
            Console.WriteLine(question);
            var ynResponse = Console.ReadLine();
            var response = ynResponse != null &&
                           ynResponse.Equals("y",
                               StringComparison.InvariantCultureIgnoreCase);
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }

    /// <summary>
```

```
    /// Helper method to get a string response from the user through the console.
    /// </summary>
    /// <param name="question">The question string to print on the console.</
param>
    /// <param name="defaultAnswer">Optional default answer to use.</param>
    /// <returns>True if the user responds with a yes.</returns>
    private static string GetUserResponse(string question, string defaultAnswer)
    {
        if (UseConsole)
        {
            var response = "";
            while (string.IsNullOrEmpty(response))
            {
                Console.WriteLine(question);
                response = Console.ReadLine();
            }
            return response;
        }
        // If not using the console, use the default.
        return defaultAnswer;
    }
}
```

Amazon SQS 작업을 래핑하는 클래스를 만듭니다.

```
/// <summary>
/// Wrapper for Amazon Simple Queue Service (SQS) operations.
/// </summary>
public class SQSWrapper
{
    private readonly IAmazonSQS _amazonSQSClient;

    /// <summary>
    /// Constructor for the Amazon SQS wrapper.
    /// </summary>
    /// <param name="amazonSQS">The injected Amazon SQS client.</param>
    public SQSWrapper(IAmazonSQS amazonSQS)
    {
        _amazonSQSClient = amazonSQS;
    }
```

```
    /// <summary>
    /// Create a queue with a specific name.
    /// </summary>
    /// <param name="queueName">The name for the queue.</param>
    /// <param name="useFifoQueue">True to use a FIFO queue.</param>
    /// <returns>The url for the queue.</returns>
    public async Task<string> CreateQueueWithName(string queueName, bool
useFifoQueue)
    {
        int maxMessage = 256 * 1024;
        var queueAttributes = new Dictionary<string, string>
        {
            {
                QueueAttributeName.MaximumMessageSize,
                maxMessage.ToString()
            }
        };

        var createQueueRequest = new CreateQueueRequest()
        {
            QueueName = queueName,
            Attributes = queueAttributes
        };

        if (useFifoQueue)
        {
            // Update the name if it is not correct for a FIFO queue.
            if (!queueName.EndsWith(".fifo"))
            {
                createQueueRequest.QueueName = queueName + ".fifo";
            }

            // Add an attribute for a FIFO queue.
            createQueueRequest.Attributes.Add(
                QueueAttributeName.FifoQueue, "true");
        }

        var createResponse = await _amazonSQSClient.CreateQueueAsync(
            new CreateQueueRequest()
            {
                QueueName = queueName
            });
        return createResponse.QueueUrl;
    }
```

```
    /// <summary>
    /// Get the ARN for a queue from its URL.
    /// </summary>
    /// <param name="queueUrl">The URL of the queue.</param>
    /// <returns>The ARN of the queue.</returns>
    public async Task<string> GetQueueArnByUrl(string queueUrl)
    {
        var getAttributesRequest = new GetQueueAttributesRequest()
        {
            QueueUrl = queueUrl,
            AttributeNames = new List<string>() { QueueAttributeName.QueueArn }
        };

        var getAttributesResponse = await
_amazonSQSClient.GetQueueAttributesAsync(
            getAttributesRequest);

        return getAttributesResponse.QueueARN;
    }

    /// <summary>
    /// Set the policy attribute of a queue for a topic.
    /// </summary>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="queueUrl">The url for the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> SetQueuePolicyForTopic(string queueArn, string
topicArn, string queueUrl)
    {
        var queuePolicy = "{" +
                                "\"Version\": \"2012-10-17\"," +
                                "\"Statement\": [{" +
                                     "\"Effect\": \"Allow\"," +
                                     "\"Principal\": {" +
                                         $"\"Service\": " +
                                              "\"sns.amazonaws.com\"" +
                                            "}," +
                                     "\"Action\": \"sqs:SendMessage\"," +
                                     $"\"Resource\": \"{queueArn}\"," +
                                      "\"Condition\": {" +
                                           "\"ArnEquals\": {" +
```

```
                                                $"\"aws:SourceArn\":
 \"{topicArn}\"" +
                                            "}" +
                                        "}" +
                                    "}]" +
                                "}";
        var attributesResponse = await _amazonSQSClient.SetQueueAttributesAsync(
            new SetQueueAttributesRequest()
            {
                QueueUrl = queueUrl,
                Attributes = new Dictionary<string, string>() { { "Policy",
 queuePolicy } }
            });
        return attributesResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Receive messages from a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>The list of messages.</returns>
    public async Task<List<Message>> ReceiveMessagesByUrl(string queueUrl, int
 maxMessages)
    {
        // Setting WaitTimeSeconds to non-zero enables long polling.
        // For information about long polling, see
        // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
        var messageResponse = await _amazonSQSClient.ReceiveMessageAsync(
            new ReceiveMessageRequest()
            {
                QueueUrl = queueUrl,
                MaxNumberOfMessages = maxMessages,
                WaitTimeSeconds = 1
            });
        return messageResponse.Messages;
    }

    /// <summary>
    /// Delete a batch of messages from a queue by its url.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>True if successful.</returns>
```

```
    public async Task<bool> DeleteMessageBatchByUrl(string queueUrl,
 List<Message> messages)
    {
        var deleteRequest = new DeleteMessageBatchRequest()
        {
            QueueUrl = queueUrl,
            Entries = new List<DeleteMessageBatchRequestEntry>()
        };
        foreach (var message in messages)
        {
            deleteRequest.Entries.Add(new DeleteMessageBatchRequestEntry()
            {
                ReceiptHandle = message.ReceiptHandle,
                Id = message.MessageId
            });
        }

        var deleteResponse = await
 _amazonSQSClient.DeleteMessageBatchAsync(deleteRequest);

        return deleteResponse.Failed.Any();
    }

    /// <summary>
    /// Delete a queue by its URL.
    /// </summary>
    /// <param name="queueUrl">The url of the queue.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteQueueByUrl(string queueUrl)
    {
        var deleteResponse = await _amazonSQSClient.DeleteQueueAsync(
            new DeleteQueueRequest()
            {
                QueueUrl = queueUrl
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

Amazon SNS 작업을 래핑하는 클래스를 만듭니다.

```
/// <summary>
/// Wrapper for Amazon Simple Notification Service (SNS) operations.
/// </summary>
public class SNSWrapper
{
    private readonly IAmazonSimpleNotificationService _amazonSNSClient;

    /// <summary>
    /// Constructor for the Amazon SNS wrapper.
    /// </summary>
    /// <param name="amazonSQS">The injected Amazon SNS client.</param>
    public SNSWrapper(IAmazonSimpleNotificationService amazonSNS)
    {
        _amazonSNSClient = amazonSNS;
    }

    /// <summary>
    /// Create a new topic with a name and specific FIFO and de-duplication
 attributes.
    /// </summary>
    /// <param name="topicName">The name for the topic.</param>
    /// <param name="useFifoTopic">True to use a FIFO topic.</param>
    /// <param name="useContentBasedDeduplication">True to use content-based de-
duplication.</param>
    /// <returns>The ARN of the new topic.</returns>
    public async Task<string> CreateTopicWithName(string topicName, bool
 useFifoTopic, bool useContentBasedDeduplication)
    {
        var createTopicRequest = new CreateTopicRequest()
        {
            Name = topicName,
        };

        if (useFifoTopic)
        {
            // Update the name if it is not correct for a FIFO topic.
            if (!topicName.EndsWith(".fifo"))
            {
                createTopicRequest.Name = topicName + ".fifo";
            }

            // Add the attributes from the method parameters.
            createTopicRequest.Attributes = new Dictionary<string, string>
            {
```

```
                { "FifoTopic", "true" }
            };
            if (useContentBasedDeduplication)
            {
                createTopicRequest.Attributes.Add("ContentBasedDeduplication",
"true");
            }
        }

        var createResponse = await
_amazonSNSClient.CreateTopicAsync(createTopicRequest);
        return createResponse.TopicArn;
    }

    /// <summary>
    /// Subscribe a queue to a topic with optional filters.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="useFifoTopic">The optional filtering policy for the
subscription.</param>
    /// <param name="queueArn">The ARN of the queue.</param>
    /// <returns>The ARN of the new subscription.</returns>
    public async Task<string> SubscribeTopicWithFilter(string topicArn, string?
filterPolicy, string queueArn)
    {
        var subscribeRequest = new SubscribeRequest()
        {
            TopicArn = topicArn,
            Protocol = "sqs",
            Endpoint = queueArn
        };

        if (!string.IsNullOrEmpty(filterPolicy))
        {
            subscribeRequest.Attributes = new Dictionary<string, string>
{ { "FilterPolicy", filterPolicy } };
        }

        var subscribeResponse = await
_amazonSNSClient.SubscribeAsync(subscribeRequest);
        return subscribeResponse.SubscriptionArn;
    }

    /// <summary>
```

```
    /// Publish a message to a topic with an attribute and optional deduplication
 and group IDs.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <param name="message">The message to publish.</param>
    /// <param name="attributeName">The optional attribute for the message.</
param>
    /// <param name="attributeValue">The optional attribute value for the
 message.</param>
    /// <param name="deduplicationId">The optional deduplication ID for the
 message.</param>
    /// <param name="groupId">The optional group ID for the message.</param>
    /// <returns>The ID of the message published.</returns>
    public async Task<string> PublishToTopicWithAttribute(
        string topicArn,
        string message,
        string? attributeName = null,
        string? attributeValue = null,
        string? deduplicationId = null,
        string? groupId = null)
    {
        var publishRequest = new PublishRequest()
        {
            TopicArn = topicArn,
            Message = message,
            MessageDeduplicationId = deduplicationId,
            MessageGroupId = groupId
        };

        if (attributeValue != null)
        {
            // Add the string attribute if it exists.
            publishRequest.MessageAttributes =
                new Dictionary<string, MessageAttributeValue>
                {
                    { attributeName!, new MessageAttributeValue() { StringValue =
 attributeValue, DataType = "String"} }
                };
        }

        var publishResponse = await
 _amazonSNSClient.PublishAsync(publishRequest);
        return publishResponse.MessageId;
    }
```

```
    /// <summary>
    /// Unsubscribe from a topic by a subscription ARN.
    /// </summary>
    /// <param name="subscriptionArn">The ARN of the subscription.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> UnsubscribeByArn(string subscriptionArn)
    {
        var unsubscribeResponse = await _amazonSNSClient.UnsubscribeAsync(
            new UnsubscribeRequest()
            {
                SubscriptionArn = subscriptionArn
            });
        return unsubscribeResponse.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Delete a topic by its topic ARN.
    /// </summary>
    /// <param name="topicArn">The ARN of the topic.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> DeleteTopicByArn(string topicArn)
    {
        var deleteResponse = await _amazonSNSClient.DeleteTopicAsync(
            new DeleteTopicRequest()
            {
                TopicArn = topicArn
            });
        return deleteResponse.HttpStatusCode == HttpStatusCode.OK;
    }
}
```

- API 세부 정보는 AWS SDK for .NET API 참조의 다음 주제를 참조하십시오.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes

- 게시
- ReceiveMessage
- SetQueueAttributes
- Subscribe
- Unsubscribe

C++

SDK for C++

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Workflow for messaging with topics and queues using Amazon SNS and Amazon
 SQS.
/*!
 \param clientConfig Aws client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::TopicsAndQueues::messagingWithTopicsAndQueues(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    std::cout << "Welcome to messaging with topics and queues." << std::endl;
    printAsterisksLine();
    std::cout << "In this workflow, you will create an SNS topic and subscribe "
              << NUMBER_OF_QUEUES <<
              " SQS queues to the topic." << std::endl;
    std::cout
            << "You can select from several options for configuring the topic and
 the subscriptions for the "
            << NUMBER_OF_QUEUES << " queues." << std::endl;
    std::cout << "You can then post to the topic and see the results in the
 queues."
              << std::endl;
```

```
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    printAsterisksLine();

    std::cout << "SNS topics can be configured as FIFO (First-In-First-Out)."
              << std::endl;
    std::cout
            << "FIFO topics deliver messages in order and support deduplication
 and message filtering."
            << std::endl;
    bool isFifoTopic = askYesNoQuestion(
            "Would you like to work with FIFO topics? (y/n) ");

    bool contentBasedDeduplication = false;
    Aws::String topicName;
    if (isFifoTopic) {
        printAsterisksLine();
        std::cout << "Because you have chosen a FIFO topic, deduplication is
 supported."
                  << std::endl;
        std::cout
                << "Deduplication IDs are either set in the message or
 automatically generated "
                << "from content using a hash function." << std::endl;
        std::cout
                << "If a message is successfully published to an SNS FIFO topic,
 any message "
                << "published and determined to have the same deduplication ID, "
                << std::endl;
        std::cout
                << "within the five-minute deduplication interval, is accepted
 but not delivered."
                << std::endl;
        std::cout
                << "For more information about deduplication, "
                << "see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-
dedup.html."
                << std::endl;
        contentBasedDeduplication = askYesNoQuestion(
                "Use content-based deduplication instead of entering a
 deduplication ID? (y/n) ");
    }
```

```
    printAsterisksLine();

    Aws::SQS::SQSClient sqsClient(clientConfiguration);
    Aws::Vector<Aws::String> queueURLS;
    Aws::Vector<Aws::String> subscriptionARNS;

    Aws::String topicARN;
    {
        topicName = askQuestion("Enter a name for your SNS topic. ");

        // 1.  Create an Amazon SNS topic, either FIFO or non-FIFO.
        Aws::SNS::Model::CreateTopicRequest request;

        if (isFifoTopic) {
            request.AddAttributes("FifoTopic", "true");
            if (contentBasedDeduplication) {
                request.AddAttributes("ContentBasedDeduplication", "true");
            }
            topicName = topicName + FIFO_SUFFIX;

            std::cout
                    << "Because you have selected a FIFO topic, '.fifo' must be
appended to the topic name."
                    << std::endl;
        }

        request.SetName(topicName);

        Aws::SNS::Model::CreateTopicOutcome outcome =
snsClient.CreateTopic(request);

        if (outcome.IsSuccess()) {
            topicARN = outcome.GetResult().GetTopicArn();
            std::cout << "Your new topic with the name '" << topicName
                      << "' and the topic Amazon Resource Name (ARN) " <<
std::endl;
            std::cout << "'" << topicARN << "' has been created." << std::endl;

        }
        else {
            std::cerr << "Error with TopicsAndQueues::CreateTopic. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
```

```
            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }
    }

    printAsterisksLine();

    std::cout << "Now you will create " << NUMBER_OF_QUEUES
              << " SQS queues to subscribe to the topic." << std::endl;
    Aws::Vector<Aws::String> queueNames;
    bool filteringMessages = false;
    bool first = true;
    for (int i = 1; i <= NUMBER_OF_QUEUES; ++i) {
        Aws::String queueURL;
        Aws::String queueName;
        {
            printAsterisksLine();
            std::ostringstream ostringstream;
            ostringstream << "Enter a name for " << (first ? "an" : "the next")
                          << " SQS queue. ";
            queueName = askQuestion(ostringstream.str());

            // 2.  Create an SQS queue.
            Aws::SQS::Model::CreateQueueRequest request;
            if (isFifoTopic) {

request.AddAttributes(Aws::SQS::Model::QueueAttributeName::FifoQueue,
                                      "true");
                queueName = queueName + FIFO_SUFFIX;

                if (first) // Only explain this once.
                {
                    std::cout
                            << "Because you are creating a FIFO SQS queue,
'.fifo' must "
                            << "be appended to the queue name." << std::endl;
                }
            }
```

```
            request.SetQueueName(queueName);
            queueNames.push_back(queueName);

            Aws::SQS::Model::CreateQueueOutcome outcome =
                    sqsClient.CreateQueue(request);

            if (outcome.IsSuccess()) {
                queueURL = outcome.GetResult().GetQueueUrl();
                std::cout << "Your new SQS queue with the name '" << queueName
                          << "' and the queue URL " << std::endl;
                std::cout << "'" << queueURL << "' has been created." <<
std::endl;
            }
            else {
                std::cerr << "Error with SQS::CreateQueue. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }
        queueURLS.push_back(queueURL);

        if (first) // Only explain this once.
        {
            std::cout
                    << "The queue URL is used to retrieve the queue ARN, which is
"
                    << "used to create a subscription." << std::endl;
        }

        Aws::String queueARN;
        {
            // 3.  Get the SQS queue ARN attribute.
            Aws::SQS::Model::GetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);

request.AddAttributeNames(Aws::SQS::Model::QueueAttributeName::QueueArn);
```

```
            Aws::SQS::Model::GetQueueAttributesOutcome outcome =
                    sqsClient.GetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                const Aws::Map<Aws::SQS::Model::QueueAttributeName, Aws::String>
&attributes =
                        outcome.GetResult().GetAttributes();
                const auto &iter = attributes.find(
                        Aws::SQS::Model::QueueAttributeName::QueueArn);
                if (iter != attributes.end()) {
                    queueARN = iter->second;
                    std::cout << "The queue ARN '" << queueARN
                              << "' has been retrieved."
                              << std::endl;
                }
                else {
                    std::cerr
                            << "Error ARN attribute not returned by
GetQueueAttribute."
                            << std::endl;

                    cleanUp(topicARN,
                            queueURLS,
                            subscriptionARNS,
                            snsClient,
                            sqsClient);

                    return false;
                }
            }
            else {
                std::cerr << "Error with SQS::GetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
```

```
        }

        if (first) {
            std::cout
                    << "An IAM policy must be attached to an SQS queue, enabling
it to receive "
                       "messages from an SNS topic." << std::endl;
        }

        {
            // 4.  Set the SQS queue policy attribute with a policy enabling the
receipt of SNS messages.
            Aws::SQS::Model::SetQueueAttributesRequest request;
            request.SetQueueUrl(queueURL);
            Aws::String policy = createPolicyForQueue(queueARN, topicARN);
            request.AddAttributes(Aws::SQS::Model::QueueAttributeName::Policy,
                                  policy);

            Aws::SQS::Model::SetQueueAttributesOutcome outcome =
                    sqsClient.SetQueueAttributes(request);

            if (outcome.IsSuccess()) {
                std::cout << "The attributes for the queue '" << queueName
                          << "' were successfully updated." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::SetQueueAttributes. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        printAsterisksLine();

        {
            // 5.  Subscribe the SQS queue to the SNS topic.
```

```
            Aws::SNS::Model::SubscribeRequest request;
            request.SetTopicArn(topicARN);
            request.SetProtocol("sqs");
            request.SetEndpoint(queueARN);
            if (isFifoTopic) {
                if (first) {
                    std::cout << "Subscriptions to a FIFO topic can have
 filters."
                              << std::endl;
                    std::cout
                            << "If you add a filter to this subscription, then
 only the filtered messages "
                            << "will be received in the queue." << std::endl;
                    std::cout << "For information about message filtering, "
                              << "see https://docs.aws.amazon.com/sns/latest/dg/
sns-message-filtering.html"
                              << std::endl;
                    std::cout << "For this example, you can filter messages by a
 \""
                              << TONE_ATTRIBUTE << "\" attribute." << std::endl;
                }

                std::ostringstream ostringstream;
                ostringstream << "Filter messages for \"" << queueName
                              << "\"'s subscription to the topic \""
                              << topicName << "\"?  (y/n)";

                // Add filter if user answers yes.
                if (askYesNoQuestion(ostringstream.str())) {
                    Aws::String jsonPolicy = getFilterPolicyFromUser();
                    if (!jsonPolicy.empty()) {
                        filteringMessages = true;

                        std::cout << "This is the filter policy for this
 subscription."
                                  << std::endl;
                        std::cout << jsonPolicy << std::endl;

                        request.AddAttributes("FilterPolicy", jsonPolicy);
                    }
                    else {
                        std::cout
                                << "Because you did not select any attributes, no
 filter "
```

```
                                << "will be added to this subscription." <<
std::endl;
                    }
                }
            }  // if (isFifoTopic)
            Aws::SNS::Model::SubscribeOutcome outcome =
snsClient.Subscribe(request);

            if (outcome.IsSuccess()) {
                Aws::String subscriptionARN =
outcome.GetResult().GetSubscriptionArn();
                std::cout << "The queue '" << queueName
                          << "' has been subscribed to the topic '"
                          << "'" << topicName << "'" << std::endl;
                std::cout << "with the subscription ARN '" << subscriptionARN <<
"."
                          << std::endl;
                subscriptionARNS.push_back(subscriptionARN);
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Subscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        first = false;
    }

    first = true;
    do {
        printAsterisksLine();

        // 6.  Publish a message to the SNS topic.
        Aws::SNS::Model::PublishRequest request;
        request.SetTopicArn(topicARN);
```

```
        Aws::String message = askQuestion("Enter a message text to publish.  ");
        request.SetMessage(message);
        if (isFifoTopic) {
            if (first) {
                std::cout
                        << "Because you are using a FIFO topic, you must set a
message group ID."
                        << std::endl;
                std::cout
                        << "All messages within the same group will be received
in the "
                        << "order they were published." << std::endl;
            }
            Aws::String messageGroupID = askQuestion(
                    "Enter a message group ID for this message. ");
            request.SetMessageGroupId(messageGroupID);
            if (!contentBasedDeduplication) {
                if (first) {
                    std::cout
                            << "Because you are not using content-based
deduplication, "
                            << "you must enter a deduplication ID." << std::endl;
                }
                Aws::String deduplicationID = askQuestion(
                        "Enter a deduplication ID for this message. ");
                request.SetMessageDeduplicationId(deduplicationID);
            }
        }

        if (filteringMessages && askYesNoQuestion(
                "Add an attribute to this message? (y/n) ")) {
            for (size_t i = 0; i < TONES.size(); ++i) {
                std::cout << "  " << (i + 1) << ". " << TONES[i] << std::endl;
            }
            int selection = askQuestionForIntRange(
                    "Enter a number for an attribute. ",
                    1, static_cast<int>(TONES.size()));
            Aws::SNS::Model::MessageAttributeValue messageAttributeValue;
            messageAttributeValue.SetDataType("String");
            messageAttributeValue.SetStringValue(TONES[selection - 1]);
            request.AddMessageAttributes(TONE_ATTRIBUTE, messageAttributeValue);
        }

        Aws::SNS::Model::PublishOutcome outcome = snsClient.Publish(request);
```

```
        if (outcome.IsSuccess()) {
            std::cout << "Your message was successfully published." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::Publish. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

            cleanUp(topicARN,
                    queueURLS,
                    subscriptionARNS,
                    snsClient,
                    sqsClient);

            return false;
        }

        first = false;
    } while (askYesNoQuestion("Post another message? (y/n) "));

    printAsterisksLine();

    std::cout << "Now the SQS queue will be polled to retrieve the messages."
              << std::endl;
    askQuestion("Press any key to continue...", alwaysTrueTest);

    for (size_t i = 0; i < queueURLS.size(); ++i) {
        // 7.  Poll an SQS queue for its messages.
        std::vector<Aws::String> messages;
        std::vector<Aws::String> receiptHandles;
        while (true) {
            Aws::SQS::Model::ReceiveMessageRequest request;
            request.SetMaxNumberOfMessages(10);
            request.SetQueueUrl(queueURLS[i]);

            // Setting WaitTimeSeconds to non-zero enables long polling.
            // For information about long polling, see
            // https://docs.aws.amazon.com/AWSSimpleQueueService/latest/
SQSDeveloperGuide/sqs-short-and-long-polling.html
            request.SetWaitTimeSeconds(1);
            Aws::SQS::Model::ReceiveMessageOutcome outcome =
                    sqsClient.ReceiveMessage(request);
```

```
            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::SQS::Model::Message> &newMessages =
outcome.GetResult().GetMessages();
                if (newMessages.empty()) {
                    break;
                }
                else {
                    for (const Aws::SQS::Model::Message &message: newMessages) {
                        messages.push_back(message.GetBody());
                        receiptHandles.push_back(message.GetReceiptHandle());
                    }
                }
            }
            else {
                std::cerr << "Error with SQS::ReceiveMessage. "
                          << outcome.GetError().GetMessage()
                          << std::endl;

                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }

        printAsterisksLine();

        if (messages.empty()) {
            std::cout << "No messages were ";
        }
        else if (messages.size() == 1) {
            std::cout << "One message was ";
        }
        else {
            std::cout << messages.size() << " messages were ";
        }
        std::cout << "received by the queue '" << queueNames[i]
                  << "'." << std::endl;
        for (const Aws::String &message: messages) {
            std::cout << "  Message : '" << message << "'."
                      << std::endl;
```

```
        }

        // 8.  Delete a batch of messages from an SQS queue.
        if (!receiptHandles.empty()) {
            Aws::SQS::Model::DeleteMessageBatchRequest request;
            request.SetQueueUrl(queueURLS[i]);
            int id = 1; // Ids must be unique within a batch delete request.
            for (const Aws::String &receiptHandle: receiptHandles) {
                Aws::SQS::Model::DeleteMessageBatchRequestEntry entry;
                entry.SetId(std::to_string(id));
                ++id;
                entry.SetReceiptHandle(receiptHandle);
                request.AddEntries(entry);
            }

            Aws::SQS::Model::DeleteMessageBatchOutcome outcome =
                    sqsClient.DeleteMessageBatch(request);

            if (outcome.IsSuccess()) {
                std::cout << "The batch deletion of messages was successful."
                          << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteMessageBatch. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                cleanUp(topicARN,
                        queueURLS,
                        subscriptionARNS,
                        snsClient,
                        sqsClient);

                return false;
            }
        }
    }

    return cleanUp(topicARN,
                   queueURLS,
                   subscriptionARNS,
                   snsClient,
                   sqsClient,
                   true); // askUser
}
```

```
bool AwsDoc::TopicsAndQueues::cleanUp(const Aws::String &topicARN,
                                      const Aws::Vector<Aws::String> &queueURLS,
                                      const Aws::Vector<Aws::String>
 &subscriptionARNS,
                                      const Aws::SNS::SNSClient &snsClient,
                                      const Aws::SQS::SQSClient &sqsClient,
                                      bool askUser) {
    bool result = true;
    printAsterisksLine();
    if (!queueURLS.empty() && askUser &&
        askYesNoQuestion("Delete the SQS queues? (y/n) ")) {

        for (const auto &queueURL: queueURLS) {
            // 9.  Delete an SQS queue.
            Aws::SQS::Model::DeleteQueueRequest request;
            request.SetQueueUrl(queueURL);

            Aws::SQS::Model::DeleteQueueOutcome outcome =
                    sqsClient.DeleteQueue(request);

            if (outcome.IsSuccess()) {
                std::cout << "The queue with URL '" << queueURL
                          << "' was successfully deleted." << std::endl;
            }
            else {
                std::cerr << "Error with SQS::DeleteQueue. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }

        for (const auto &subscriptionARN: subscriptionARNS) {
            // 10. Unsubscribe an SNS subscription.
            Aws::SNS::Model::UnsubscribeRequest request;
            request.SetSubscriptionArn(subscriptionARN);

            Aws::SNS::Model::UnsubscribeOutcome outcome =
                    snsClient.Unsubscribe(request);

            if (outcome.IsSuccess()) {
```

```
                std::cout << "Unsubscribe of subscription ARN '" <<
 subscriptionARN
                          << "' was successful." << std::endl;
            }
            else {
                std::cerr << "Error with TopicsAndQueues::Unsubscribe. "
                          << outcome.GetError().GetMessage()
                          << std::endl;
                result = false;
            }
        }
    }

    printAsterisksLine();
    if (!topicARN.empty() && askUser &&
        askYesNoQuestion("Delete the SNS topic? (y/n) ")) {

        // 11. Delete an SNS topic.
        Aws::SNS::Model::DeleteTopicRequest request;
        request.SetTopicArn(topicARN);

        Aws::SNS::Model::DeleteTopicOutcome outcome =
 snsClient.DeleteTopic(request);

        if (outcome.IsSuccess()) {
            std::cout << "The topic with ARN '" << topicARN
                      << "' was successfully deleted." << std::endl;
        }
        else {
            std::cerr << "Error with TopicsAndQueues::DeleteTopicRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            result = false;
        }
    }

    return result;
}

//! Create an IAM policy that gives an SQS queue permission to receive messages
 from an SNS topic.
/*!
 \sa createPolicyForQueue()
 \param queueARN: The SQS queue Amazon Resource Name (ARN).
```

```
 \param topicARN: The SNS topic ARN.
 \return Aws::String: The policy as JSON.
 */
Aws::String AwsDoc::TopicsAndQueues::createPolicyForQueue(const Aws::String
 &queueARN,
                                                          const Aws::String
 &topicARN) {
    std::ostringstream policyStream;
    policyStream << R"({
        "Statement": [
        {
            "Effect": "Allow",
                    "Principal": {
                "Service": "sns.amazonaws.com"
            },
            "Action": "sqs:SendMessage",
                    "Resource": ")" << queueARN << R"(",
                    "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": ")" << topicARN << R"("
                }
            }
        }
        ]
    })";

    return policyStream.str();
}
```

- API 세부 정보는 AWS SDK for C++ API 참조의 다음 주제를 참조하십시오.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - 게시
  - ReceiveMessage
  - SetQueueAttributes

- Subscribe
- Unsubscribe

Go

SDK for Go V2

> **Note**
>
> 더 많은 것이 있어요 GitHub. AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

명령 프롬프트에서 대화형 시나리오를 실행합니다.

```
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() (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(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(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(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(
 queueName string, queueUrl string, topicName string, topicArn string, ordinal
 string,
 isFifoTopic bool) (string, bool) {

 queueArn, err := runner.sqsActor.GetQueueArn(queueUrl)
 if err != nil {
  panic(err)
 }
 log.Printf("The ARN of your queue is: %v.\n", queueArn)

 err = runner.sqsActor.AttachSendMessagePolicy(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(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(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(topicArn, message, groupId, dedupId, TONE_KEY,
 toneSelection)
  if err != nil {
   panic(err)
  }
  log.Println(("Your message was published."))

  publishMore = runner.questioner.AskBool("Do you want to publish another
 messsage? (y/n) ", "y")
 }
}

func (runner ScenarioRunner) PollForMessages(queueUrls []string) {
```

```
 log.Println("Polling queues for messages...")
 for _, queueUrl := range queueUrls {
  var messages []types.Message
  for {
   currentMsgs, err := runner.sqsActor.GetMessages(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(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(
 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()
  }
 }()
 queueCount := 2

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome to messaging with topics and queues.\n\n"+
  "In this workflow, 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()
 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(ordinal, isFifoTopic)
  resources.queueUrls = append(resources.queueUrls, queueUrl)

  _, filtering := runner.SubscribeQueueToTopic(queueName, queueUrl, topicName,
 topicArn, ordinal, isFifoTopic)
  usingFilters = usingFilters || filtering
 }

 log.Println(strings.Repeat("-", 88))
 runner.PublishMessages(topicArn, isFifoTopic, contentBasedDeduplication,
 usingFilters)
 log.Println(strings.Repeat("-", 88))
 runner.PollForMessages(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()
 }

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

이 예제에 사용된 Amazon SNS 작업을 래핑하는 구조체를 정의하십시오.

```
// 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(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(context.TODO(), &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(topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(context.TODO(), &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(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(context.TODO(), &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}
```

```
// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
func (actor SnsActions) Publish(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(context.TODO(), &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
 return err
}
```

이 예제에 사용된 Amazon SQS 작업을 래핑하는 구조체를 정의합니다.

```
// 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(queueName string, isFifoQueue bool) (string,
 error) {
 var queueUrl string
 queueAttributes := map[string]string{}
 if isFifoQueue {
  queueAttributes["FifoQueue"] = "true"
 }
 queue, err := actor.SqsClient.CreateQueue(context.TODO(), &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(queueUrl string) (string, error) {
 var queueArn string
 arnAttributeName := types.QueueAttributeNameQueueArn
 attribute, err := actor.SqsClient.GetQueueAttributes(context.TODO(),
 &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(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(context.TODO(),
 &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(queueUrl string, maxMessages int32, waitTime
 int32) ([]types.Message, error) {
 var messages []types.Message
 result, err := actor.SqsClient.ReceiveMessage(context.TODO(),
 &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(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(context.TODO(),
 &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(queueUrl string) error {
 _, err := actor.SqsClient.DeleteQueue(context.TODO(), &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 다음 주제를 참조하십시오.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue

- DeleteTopic
- GetQueueAttributes
- 게시
- ReceiveMessage
- SetQueueAttributes
- Subscribe
- Unsubscribe

JavaScript

(v3) 용 SDK JavaScript

> **Note**
>
> 더 많은 내용이 있습니다. GitHub AWS 코드 예제 리포지토리에서 전체 예제를 찾고 설정 및 실행하는 방법을 배워보세요.

이 워크플로의 시작점입니다.

```
import { SNSClient } from "@aws-sdk/client-sns";
import { SQSClient } from "@aws-sdk/client-sqs";

import { TopicsQueuesWkflw } from "./TopicsQueuesWkflw.js";
import { Prompter } from "@aws-doc-sdk-examples/lib/prompter.js";
import { SlowLogger } from "@aws-doc-sdk-examples/lib/slow-logger.js";

export const startSnsWorkflow = () => {
  const noLoggerDelay = process.argv.find((arg) => arg === "--no-logger-delay");
  const snsClient = new SNSClient({});
  const sqsClient = new SQSClient({});
  const prompter = new Prompter();
  const logger = noLoggerDelay ? console : new SlowLogger(25);

  const wkflw = new TopicsQueuesWkflw(snsClient, sqsClient, prompter, logger);

  wkflw.start();
};
```

위 코드는 필요한 종속성을 제공하고 워크플로를 시작합니다. 다음 섹션에는 대부분의 예제가 포함되어 있습니다.

```
const toneChoices = [
  { name: "cheerful", value: "cheerful" },
  { name: "funny", value: "funny" },
  { name: "serious", value: "serious" },
  { name: "sincere", value: "sincere" },
];

export class TopicsQueuesWkflw {
  // SNS topic is configured as First-In-First-Out
  isFifo = true;

  // Automatic content-based deduplication is enabled.
  autoDedup = false;

  snsClient;
  sqsClient;
  topicName;
  topicArn;
  subscriptionArns = [];
  /**
   * @type {{ queueName: string, queueArn: string, queueUrl: string, policy?:
 string }[]}
   */
  queues = [];
  prompter;

  /**
   * @param {import('@aws-sdk/client-sns').SNSClient} snsClient
   * @param {import('@aws-sdk/client-sqs').SQSClient} sqsClient
   * @param {import('../../libs/prompter.js').Prompter} prompter
   * @param {import('../../libs/logger.js').Logger} logger
   */
  constructor(snsClient, sqsClient, prompter, logger) {
    this.snsClient = snsClient;
    this.sqsClient = sqsClient;
    this.prompter = prompter;
    this.logger = logger;
```

```
  }

  async welcome() {
    await this.logger.log(MESSAGES.description);
  }

  async confirmFifo() {
    await this.logger.log(MESSAGES.snsFifoDescription);
    this.isFifo = await this.prompter.confirm({
      message: MESSAGES.snsFifoPrompt,
    });

    if (this.isFifo) {
      this.logger.logSeparator(MESSAGES.headerDedup);
      await this.logger.log(MESSAGES.deduplicationNotice);
      await this.logger.log(MESSAGES.deduplicationDescription);
      this.autoDedup = await this.prompter.confirm({
        message: MESSAGES.deduplicationPrompt,
      });
    }
  }

  async createTopic() {
    await this.logger.log(MESSAGES.creatingTopics);
    this.topicName = await this.prompter.input({
      message: MESSAGES.topicNamePrompt,
    });
    if (this.isFifo) {
      this.topicName += ".fifo";
      this.logger.logSeparator(MESSAGES.headerFifoNaming);
      await this.logger.log(MESSAGES.appendFifoNotice);
    }

    const response = await this.snsClient.send(
      new CreateTopicCommand({
        Name: this.topicName,
        Attributes: {
          FifoTopic: this.isFifo ? "true" : "false",
          ...(this.autoDedup ? { ContentBasedDeduplication: "true" } : {}),
        },
      }),
    );

    this.topicArn = response.TopicArn;
```

```
    await this.logger.log(
      MESSAGES.topicCreatedNotice
        .replace("${TOPIC_NAME}", this.topicName)
        .replace("${TOPIC_ARN}", this.topicArn),
    );
  }

  async createQueues() {
    await this.logger.log(MESSAGES.createQueuesNotice);
    // Increase this number to add more queues.
    let maxQueues = 2;

    for (let i = 0; i < maxQueues; i++) {
      await this.logger.log(MESSAGES.queueCount.replace("${COUNT}", i + 1));
      let queueName = await this.prompter.input({
        message: MESSAGES.queueNamePrompt.replace(
          "${EXAMPLE_NAME}",
          i === 0 ? "good-news" : "bad-news",
        ),
      });

      if (this.isFifo) {
        queueName += ".fifo";
        await this.logger.log(MESSAGES.appendFifoNotice);
      }

      const response = await this.sqsClient.send(
        new CreateQueueCommand({
          QueueName: queueName,
          Attributes: { ...(this.isFifo ? { FifoQueue: "true" } : {}) },
        }),
      );

      const { Attributes } = await this.sqsClient.send(
        new GetQueueAttributesCommand({
          QueueUrl: response.QueueUrl,
          AttributeNames: ["QueueArn"],
        }),
      );

      this.queues.push({
        queueName,
        queueArn: Attributes.QueueArn,
```

```
        queueUrl: response.QueueUrl,
      });

      await this.logger.log(
        MESSAGES.queueCreatedNotice
          .replace("${QUEUE_NAME}", queueName)
          .replace("${QUEUE_URL}", response.QueueUrl)
          .replace("${QUEUE_ARN}", Attributes.QueueArn),
      );
    }
  }

  async attachQueueIamPolicies() {
    for (const [index, queue] of this.queues.entries()) {
      const policy = JSON.stringify(
        {
          Statement: [
            {
              Effect: "Allow",
              Principal: {
                Service: "sns.amazonaws.com",
              },
              Action: "sqs:SendMessage",
              Resource: queue.queueArn,
              Condition: {
                ArnEquals: {
                  "aws:SourceArn": this.topicArn,
                },
              },
            },
          ],
        },
        null,
        2,
      );

      if (index !== 0) {
        this.logger.logSeparator();
      }

      await this.logger.log(MESSAGES.attachPolicyNotice);
      console.log(policy);
      const addPolicy = await this.prompter.confirm({
        message: MESSAGES.addPolicyConfirmation.replace(
```

```
          "${QUEUE_NAME}",
          queue.queueName,
        ),
      });

      if (addPolicy) {
        await this.sqsClient.send(
          new SetQueueAttributesCommand({
            QueueUrl: queue.queueUrl,
            Attributes: {
              Policy: policy,
            },
          }),
        );
        queue.policy = policy;
      } else {
        await this.logger.log(
          MESSAGES.policyNotAttachedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }
  }

  async subscribeQueuesToTopic() {
    for (const [index, queue] of this.queues.entries()) {
      /**
       * @type {import('@aws-sdk/client-sns').SubscribeCommandInput}
       */
      const subscribeParams = {
        TopicArn: this.topicArn,
        Protocol: "sqs",
        Endpoint: queue.queueArn,
      };
      let tones = [];

      if (this.isFifo) {
        if (index === 0) {
          await this.logger.log(MESSAGES.fifoFilterNotice);
        }
        tones = await this.prompter.checkbox({
          message: MESSAGES.fifoFilterSelect.replace(
```

```
            "${QUEUE_NAME}",
            queue.queueName,
          ),
          choices: toneChoices,
        });

        if (tones.length) {
          subscribeParams.Attributes = {
            FilterPolicyScope: "MessageAttributes",
            FilterPolicy: JSON.stringify({
              tone: tones,
            }),
          };
        }
      }

      const { SubscriptionArn } = await this.snsClient.send(
        new SubscribeCommand(subscribeParams),
      );

      this.subscriptionArns.push(SubscriptionArn);

      await this.logger.log(
        MESSAGES.queueSubscribedNotice
          .replace("${QUEUE_NAME}", queue.queueName)
          .replace("${TOPIC_NAME}", this.topicName)
          .replace("${TONES}", tones.length ? tones.join(", ") : "none"),
      );
    }
  }

  async publishMessages() {
    const message = await this.prompter.input({
      message: MESSAGES.publishMessagePrompt,
    });

    let groupId, deduplicationId, choices;

    if (this.isFifo) {
      await this.logger.log(MESSAGES.groupIdNotice);
      groupId = await this.prompter.input({
        message: MESSAGES.groupIdPrompt,
      });
```

```
      if (this.autoDedup === false) {
        await this.logger.log(MESSAGES.deduplicationIdNotice);
        deduplicationId = await this.prompter.input({
          message: MESSAGES.deduplicationIdPrompt,
        });
      }

      choices = await this.prompter.checkbox({
        message: MESSAGES.messageAttributesPrompt,
        choices: toneChoices,
      });
    }

    await this.snsClient.send(
      new PublishCommand({
        TopicArn: this.topicArn,
        Message: message,
        ...(groupId
          ? {
              MessageGroupId: groupId,
            }
          : {}),
        ...(deduplicationId
          ? {
              MessageDeduplicationId: deduplicationId,
            }
          : {}),
        ...(choices
          ? {
              MessageAttributes: {
                tone: {
                  DataType: "String.Array",
                  StringValue: JSON.stringify(choices),
                },
              },
            }
          : {}),
      }),
    );

    const publishAnother = await this.prompter.confirm({
      message: MESSAGES.publishAnother,
    });
```

```
    if (publishAnother) {
      await this.publishMessages();
    }
  }

  async receiveAndDeleteMessages() {
    for (const queue of this.queues) {
      const { Messages } = await this.sqsClient.send(
        new ReceiveMessageCommand({
          QueueUrl: queue.queueUrl,
        }),
      );

      if (Messages) {
        await this.logger.log(
          MESSAGES.messagesReceivedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
        console.log(Messages);

        await this.sqsClient.send(
          new DeleteMessageBatchCommand({
            QueueUrl: queue.queueUrl,
            Entries: Messages.map((message) => ({
              Id: message.MessageId,
              ReceiptHandle: message.ReceiptHandle,
            })),
          }),
        );
      } else {
        await this.logger.log(
          MESSAGES.noMessagesReceivedNotice.replace(
            "${QUEUE_NAME}",
            queue.queueName,
          ),
        );
      }
    }

    const deleteAndPoll = await this.prompter.confirm({
      message: MESSAGES.deleteAndPollConfirmation,
    });
```

```
    if (deleteAndPoll) {
      await this.receiveAndDeleteMessages();
    }
  }

  async destroyResources() {
    for (const subscriptionArn of this.subscriptionArns) {
      await this.snsClient.send(
        new UnsubscribeCommand({ SubscriptionArn: subscriptionArn }),
      );
    }

    for (const queue of this.queues) {
      await this.sqsClient.send(
        new DeleteQueueCommand({ QueueUrl: queue.queueUrl }),
      );
    }

    if (this.topicArn) {
      await this.snsClient.send(
        new DeleteTopicCommand({ TopicArn: this.topicArn }),
      );
    }
  }

  async start() {
    console.clear();

    try {
      this.logger.logSeparator(MESSAGES.headerWelcome);
      await this.welcome();
      this.logger.logSeparator(MESSAGES.headerFifo);
      await this.confirmFifo();
      this.logger.logSeparator(MESSAGES.headerCreateTopic);
      await this.createTopic();
      this.logger.logSeparator(MESSAGES.headerCreateQueues);
      await this.createQueues();
      this.logger.logSeparator(MESSAGES.headerAttachPolicy);
      await this.attachQueueIamPolicies();
      this.logger.logSeparator(MESSAGES.headerSubscribeQueues);
      await this.subscribeQueuesToTopic();
      this.logger.logSeparator(MESSAGES.headerPublishMessage);
      await this.publishMessages();
```

```
      this.logger.logSeparator(MESSAGES.headerReceiveMessages);
      await this.receiveAndDeleteMessages();
    } catch (err) {
      console.error(err);
    } finally {
      await this.destroyResources();
    }
  }
}
```

- API 세부 정보는 AWS SDK for JavaScript API 참조의 다음 주제를 참조하십시오.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - 게시
  - ReceiveMessage
  - SetQueueAttributes
  - Subscribe
  - Unsubscribe

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오AWS SDK와 함께 Amazon SNS 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

SDK를 사용하는 AWS Amazon SNS의 서버리스 예제

다음 코드 예제는 AWS SDK와 함께 Amazon SNS를 사용하는 방법을 보여줍니다.

예제

- Amazon SNS 트리거를 사용하여 Lambda 함수 호출

Amazon SNS 트리거를 사용하여 Lambda 함수 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.

.NET

AWS SDK for .NET

> **Note**
>
> 더 많은 정보가 있습니다 GitHub. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

.NET을 사용하여 Lambda로 SNS 이벤트를 사용합니다.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
using Amazon.Lambda.Core;
using Amazon.Lambda.SNSEvents;


// Assembly attribute to enable the Lambda function's JSON input to be converted
 into a .NET class.
[assembly:
 LambdaSerializer(typeof(Amazon.Lambda.Serialization.SystemTextJson.DefaultLambdaJsonSeri

namespace SnsIntegration;

public class Function
{
    public async Task FunctionHandler(SNSEvent evnt, ILambdaContext context)
    {
        foreach (var record in evnt.Records)
        {
            await ProcessRecordAsync(record, context);
        }
        context.Logger.LogInformation("done");
    }
```

```
    private async Task ProcessRecordAsync(SNSEvent.SNSRecord record,
 ILambdaContext context)
    {
        try
        {
            context.Logger.LogInformation($"Processed record
 {record.Sns.Message}");

            // TODO: Do interesting work based on the new message
            await Task.CompletedTask;
        }
        catch (Exception e)
        {
            //You can use Dead Letter Queue to handle failures. By configuring a
 Lambda DLQ.
            context.Logger.LogError($"An error occurred");
            throw;
        }
    }
}
```

Go

SDK for Go V2

> **Note**
>
> 더 많은 것이 있어요 GitHub. 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)
}
```

Java

SDK for Java 2.x

> **Note**
>
> 더 많은 것이 있어요 GitHub. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Java를 사용하여 Lambda로 SNS 이벤트를 사용합니다.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package example;

import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SNSEvent;
import com.amazonaws.services.lambda.runtime.events.SNSEvent.SNSRecord;
```

```
import java.util.Iterator;
import java.util.List;

public class SNSEventHandler implements RequestHandler<SNSEvent, Boolean> {
    LambdaLogger logger;

    @Override
    public Boolean handleRequest(SNSEvent event, Context context) {
        logger = context.getLogger();
        List<SNSRecord> records = event.getRecords();
        if (!records.isEmpty()) {
            Iterator<SNSRecord> recordsIter = records.iterator();
            while (recordsIter.hasNext()) {
                processRecord(recordsIter.next());
            }
        }
        return Boolean.TRUE;
    }

    public void processRecord(SNSRecord record) {
        try {
            String message = record.getSNS().getMessage();
            logger.log("message: " + message);
        } catch (Exception e) {
            throw new RuntimeException(e);
        }
    }

}
```

JavaScript

JavaScript (v3) 용 SDK

> **Note**
>
> 더 많은 내용이 있습니다. GitHub 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Lambda를 사용하여 SNS 이벤트를 소비합니다. JavaScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
exports.handler = async (event, context) => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record) {
  try {
    const message = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

Lambda를 사용하여 SNS 이벤트를 소비합니다. TypeScript

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
import { SNSEvent, Context, SNSHandler, SNSEventRecord } from "aws-lambda";

export const functionHandler: SNSHandler = async (
  event: SNSEvent,
  context: Context
```

```
): Promise<void> => {
  for (const record of event.Records) {
    await processMessageAsync(record);
  }
  console.info("done");
};

async function processMessageAsync(record: SNSEventRecord): Promise<any> {
  try {
    const message: string = JSON.stringify(record.Sns.Message);
    console.log(`Processed message ${message}`);
    await Promise.resolve(1); //Placeholder for actual async work
  } catch (err) {
    console.error("An error occurred");
    throw err;
  }
}
```

PHP

SDK for PHP

> **Note**
>
> 더 많은 정보가 있습니다. GitHub 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

PHP를 사용하여 Lambda로 SNS 이벤트를 사용합니다.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
<?php

/*
Since native PHP support for AWS Lambda is not available, we are utilizing Bref's
 PHP functions runtime for AWS Lambda.
For more information on Bref's PHP runtime for Lambda, refer to: https://bref.sh/
docs/runtimes/function

Another approach would be to create a custom runtime.
```

```
A practical example can be found here: https://aws.amazon.com/blogs/apn/aws-
lambda-custom-runtime-for-php-a-practical-example/
*/

// Additional composer packages may be required when using Bref or any other PHP
 functions runtime.
// require __DIR__ . '/vendor/autoload.php';

use Bref\Context\Context;
use Bref\Event\Sns\SnsEvent;
use Bref\Event\Sns\SnsHandler;

class Handler extends SnsHandler
{
    public function handleSns(SnsEvent $event, Context $context): void
    {
        foreach ($event->getRecords() as $record) {
            $message = $record->getMessage();

            // TODO: Implement your custom processing logic here
            // Any exception thrown will be logged and the invocation will be
 marked as failed

            echo "Processed Message: $message" . PHP_EOL;
        }
    }
}

return new Handler();
```

Python

SDK for Python(Boto3)

> **Note**
>
> 더 많은 것이 있어요 GitHub. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Python을 사용하여 Lambda로 SNS 이벤트를 사용합니다.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event, context):
    for record in event['Records']:
        process_message(record)
    print("done")

def process_message(record):
    try:
        message = record['Sns']['Message']
        print(f"Processed message {message}")
        # TODO; Process your record here

    except Exception as e:
        print("An error occurred")
        raise e
```

Ruby

SDK for Ruby

> **Note**
>
> 더 많은 것이 있어요 GitHub. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Ruby를 사용하여 Lambda로 SNS 이벤트를 사용합니다.

```
# Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
# SPDX-License-Identifier: Apache-2.0
def lambda_handler(event:, context:)
  event['Records'].map { |record| process_message(record) }
end

def process_message(record)
  message = record['Sns']['Message']
  puts("Processing message: #{message}")
rescue StandardError => e
  puts("Error processing message: #{e}")
```

```
  raise
end
```

Rust

SDK for Rust

> **Note**
>
> 더 많은 것이 있어요 GitHub. 서버리스 예제 리포지토리에서 전체 예제를 찾아보고 설정 및 실행 방법을 알아봅니다.

Rust를 사용하여 Lambda로 SNS 이벤트를 사용합니다.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
use aws_lambda_events::event::sns::SnsEvent;
use aws_lambda_events::sns::SnsRecord;
use lambda_runtime::{run, service_fn, Error, LambdaEvent};
use tracing::info;

// Built with the following dependencies:
//  aws_lambda_events = { version = "0.10.0", default-features = false, features
 = ["sns"] }
//  lambda_runtime = "0.8.1"
//  tokio = { version = "1", features = ["macros"] }
//  tracing = { version = "0.1", features = ["log"] }
//  tracing-subscriber = { version = "0.3", default-features = false, features =
 ["fmt"] }

async fn function_handler(event: LambdaEvent<SnsEvent>) -> Result<(), Error> {
    for event in event.payload.records {
        process_record(&event)?;
    }

    Ok(())
}

fn process_record(record: &SnsRecord) -> Result<(), Error> {
    info!("Processing SNS Message: {}", record.sns.message);
```

```
    // Implement your record handling code here.

    Ok(())
}

#[tokio::main]
async fn main() -> Result<(), Error> {
    tracing_subscriber::fmt()
        .with_max_level(tracing::Level::INFO)
        .with_target(false)
        .without_time()
        .init();

    run(service_fn(function_handler)).await
}
```

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오AWS SDK와 함께 Amazon SNS 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

SDK를 사용하는 AWS Amazon SNS의 크로스 서비스 예제

다음 샘플 애플리케이션은 AWS SDK를 사용하여 Amazon SNS를 다른 AWS 서비스애플리케이션과 결합합니다. 각 예제에는 애플리케이션 설정 및 실행 방법에 대한 지침을 찾을 수 있는 링크가 포함되어 있습니다. GitHub

예제

- DynamoDB 테이블에 데이터를 제출하기 위한 애플리케이션 구축
- 메시지를 번역하는 게시 및 구독 애플리케이션 구축
- 사용자가 레이블을 사용하여 사진을 관리할 수 있는 사진 자산 관리 애플리케이션 만들기
- Amazon Textract 탐색기 애플리케이션 생성
- Amazon Rekognition에서 SDK를 사용하여 동영상 속 사람과 물체를 감지합니다. AWS
- SDK를 사용하여 Amazon SNS 메시지를 Amazon SQS 대기열에 게시합니다. AWS
- API Gateway를 사용하여 Lambda 함수 호출
- 예약된 이벤트를 사용하여 Lambda 함수 호출

DynamoDB 테이블에 데이터를 제출하기 위한 애플리케이션 구축

다음 코드 예제에서는 Amazon DynamoDB 테이블에 데이터를 제출하고 사용자가 테이블을 업데이트할 때 알려주는 애플리케이션을 구축하는 방법을 보여줍니다.

Java

SDK for Java 2.x

Amazon DynamoDB Java API를 사용하여 데이터를 제출하고 Amazon Simple Notification Service Java API를 사용하여 문자 메시지를 전송하는 동적 웹 애플리케이션을 생성하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 의 전체 예제를 참조하십시오 GitHub.

이 예제에서 사용되는 서비스

- DynamoDB
- Amazon SNS

JavaScript

JavaScript (v3) 용 SDK

이 예제에서는 사용자가 Amazon DynamoDB 테이블에 데이터를 제출하고 Amazon Simple Notification Service(Amazon SNS)를 사용하여 관리자에게 문자 메시지를 전송하는 앱을 구축하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 의 전체 예제를 참조하십시오. GitHub

이 예시는 AWS SDK for JavaScript v3 개발자 안내서에서도 확인할 수 있습니다.

이 예제에서 사용되는 서비스

- DynamoDB
- Amazon SNS

Kotlin

SDK for Kotlin

Amazon DynamoDB Kotlin API를 사용하여 데이터를 제출하고 Amazon SNS Kotlin API를 사용하여 문자 메시지를 보내는 기본 Android 애플리케이션을 생성하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 의 전체 예제를 참조하십시오 GitHub.

이 예제에서 사용되는 서비스

- DynamoDB
- Amazon SNS

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오AWS SDK와 함께 Amazon SNS 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

메시지를 번역하는 게시 및 구독 애플리케이션 구축

다음 코드 예제에서는 구독 및 게시 기능이 있고 메시지를 번역하는 애플리케이션을 생성하는 방법을 보여줍니다.

.NET

AWS SDK for .NET

Amazon Simple Notification Service .NET API를 사용하여 구독 및 게시 기능이 있는 웹 애플리케이션을 생성하는 방법을 보여줍니다. 또한 이 예제 애플리케이션은 메시지를 번역합니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 의 전체 예제를 참조하십시오 GitHub.

이 예시에서 사용되는 서비스

- Amazon SNS
- Amazon Translate

Java

SDK for Java 2.x

Amazon Simple Notification Service Java API를 사용하여 구독 및 게시 기능이 있는 웹 애플리케이션을 생성하는 방법을 보여줍니다. 또한 이 예제 애플리케이션은 메시지를 번역합니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 의 전체 예제를 참조하십시오 GitHub.

Java Async API를 사용하는 예제를 설정하고 실행하는 방법에 대한 전체 소스 코드와 지침은 의 전체 예제를 참조하십시오. GitHub

이 예시에서 사용되는 서비스

- Amazon SNS
- Amazon Translate

Kotlin

SDK for Kotlin

Amazon SNS Kotlin API를 사용하여 구독 및 게시 기능이 있는 애플리케이션을 생성하는 방법을 보여줍니다. 또한 이 예제 애플리케이션은 메시지를 번역합니다.

전체 소스 코드와 웹 앱을 만드는 방법에 대한 지침은 전체 예제를 참조하십시오. GitHub

전체 소스 코드와 네이티브 Android 앱을 만드는 방법에 대한 지침은 전체 예제를 참조하십시오 GitHub.

이 예시에서 사용되는 서비스

- Amazon SNS
- Amazon Translate

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오AWS SDK와 함께 Amazon SNS 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

사용자가 레이블을 사용하여 사진을 관리할 수 있는 사진 자산 관리 애플리케이션 만들기

다음 코드 예제에서는 사용자가 레이블을 사용하여 사진을 관리할 수 있는 서버리스 애플리케이션을 생성하는 방법을 보여 줍니다.

.NET

AWS SDK for .NET

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 의 전체 예제를 참조하십시오 GitHub.

이 예제의 출처에 대한 자세한 내용은 AWS 커뮤니티의 게시물을 참조하십시오.

이 예시에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDK for C++

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 의 전체 예제를 참조하십시오 GitHub.

이 예제의 출처에 대한 자세한 내용은 AWS 커뮤니티의 게시물을 참조하십시오.

이 예시에서 사용되는 서비스

- API Gateway

- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDK for Java 2.x

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 의 전체 예제를 참조하십시오 GitHub.

이 예제의 출처에 대한 자세한 내용은 AWS 커뮤니티의 게시물을 참조하십시오.

이 예시에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

JavaScript (v3) 용 SDK

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 의 전체 예제를 참조하십시오. GitHub

이 예제의 출처에 대한 자세한 내용은 AWS 커뮤니티의 게시물을 참조하십시오.

이 예시에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDK for Kotlin

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 의 전체 예제를 참조하십시오 GitHub.

이 예제의 출처에 대한 자세한 내용은 AWS 커뮤니티의 게시물을 참조하십시오.

이 예시에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDK for PHP

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 의 전체 예제를 참조하십시오 GitHub.

이 예제의 출처에 대한 자세한 내용은 AWS 커뮤니티의 게시물을 참조하십시오.

이 예시에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDK for Rust

Amazon Rekognition을 사용하여 이미지에서 레이블을 감지하고 나중에 검색할 수 있도록 저장하는 사진 자산 관리 애플리케이션을 개발하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 의 전체 예제를 참조하십시오 GitHub.

이 예제의 출처에 대한 자세한 내용은 AWS 커뮤니티의 게시물을 참조하십시오.

이 예시에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오AWS SDK와 함께 Amazon SNS 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

Amazon Textract 탐색기 애플리케이션 생성

다음 코드 예제에서는 대화형 애플리케이션을 통해 Amazon Textract 출력을 탐색하는 방법을 보여줍니다.

JavaScript

JavaScript (v3) 용 SDK

를 사용하여 Amazon AWS SDK for JavaScript Textract를 사용하여 문서 이미지에서 데이터를 추출하고 대화형 웹 페이지에 표시하는 React 애플리케이션을 구축하는 방법을 보여 줍니다. 이 예제는 웹 브라우저에서 실행되며 자격 증명을 위해 인증된 Amazon Cognito 자격 증명이 필요합니다. 이 애플리케이션은 스토리지로 Amazon Simple Storage Service (Amazon S3)를 사용하고 알림을 위해 Amazon Simple Notification Service(Amazon SNS) 주제를 구독하는 Amazon Simple Queue Service(Amazon SQS) 대기열을 폴링합니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 전체 예제를 참조하십시오. GitHub

이 예시에서 사용되는 서비스

- Amazon Cognito 자격 증명
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDK for Python(Boto3)

AWS SDK for Python (Boto3) with Amazon Textract를 사용하여 문서 이미지에서 텍스트, 양식 및 표 요소를 감지하는 방법을 보여 줍니다. 입력 이미지와 Amazon Textract 출력은 탐지된 요소를 탐색할 수 있는 Tkinter 애플리케이션에 표시됩니다.

- 문서 이미지를 Amazon Textract에 제출하고 감지된 요소의 출력을 탐색합니다.
- Amazon Textract로 직접, 또는 Amazon Simple Storage Service (Amazon S3) 버킷을 통해 이미지를 제출합니다.
- 비동기식 API를 사용하여 작업이 완료되면 Amazon Simple Notification Service(Amazon SNS) 주제에 알림을 게시하는 작업을 시작합니다.
- Amazon Simple Queue Service(Amazon SQS) 대기열에서 작업 완료 메시지를 폴링하고 결과를 표시합니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 에서 전체 예제를 참조하십시오. GitHub

이 예시에서 사용되는 서비스

- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오AWS SDK와 함께 Amazon SNS 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

Amazon Rekognition에서 SDK를 사용하여 동영상 속 사람과 물체를 감지합니다. AWS

다음 코드 예제에서는 Amazon Rekognition을 사용하여 동영상에서 사람과 객체를 감지하는 방법을 보여줍니다.

Python

SDK for Python(Boto3)

Amazon Rekognition을 사용하여 비동기식 감지 작업을 시작해 동영상의 얼굴, 객체 및 사람을 감지할 수 있습니다. 또한 이 예제에서는 작업이 완료되고 주제에 대한 Amazon Simple Queue Service(Amazon SQS) 대기열을 구독할 때 Amazon Simple Notification Service(Amazon SNS) 주제를 알리도록 Amazon Rekognition을 구성합니다. 대기열이 작업에 대한 메시지를 받으면 작업이 검색되고 결과가 출력됩니다.

이 예제는 에서 가장 잘 볼 수 GitHub 있습니다. 전체 소스 코드와 설정 및 실행 방법에 대한 지침은 의 전체 예제를 참조하십시오 GitHub.

이 예시에서 사용되는 서비스

- Amazon Rekognition
- Amazon SNS
- Amazon SQS

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오AWS SDK와 함께 Amazon SNS 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

SDK를 사용하여 Amazon SNS 메시지를 Amazon SQS 대기열에 게시합니다. AWS

다음 코드 예제는 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.

Java

SDK for Java 2.x

Amazon Simple Notification Service(Amazon SNS) 및 Amazon Simple Queue Service(Amazon SQS)에서 주제 및 대기열을 사용한 메시징을 보여줍니다.

Amazon SNS 및 Amazon SQS의 주제 및 대기열을 사용한 메시징을 보여주는 전체 소스 코드 및 지침은 에서 전체 예제를 참조하십시오. GitHub

이 예시에서 사용되는 서비스

- Amazon SNS
- Amazon SQS

Kotlin

SDK for Kotlin

Amazon Simple Notification Service(Amazon SNS) 및 Amazon Simple Queue Service(Amazon SQS)에서 주제 및 대기열을 사용한 메시징을 보여줍니다.

Amazon SNS 및 Amazon SQS의 주제 및 대기열을 사용한 메시징을 보여주는 전체 소스 코드 및 지침은 에서 전체 예제를 참조하십시오. GitHub

이 예시에서 사용되는 서비스

- Amazon SNS

- Amazon SQS

AWS SDK 개발자 안내서 및 코드 예제의 전체 목록은 을 참조하십시오. AWS SDK와 함께 Amazon SNS 사용 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

API Gateway를 사용하여 Lambda 함수 호출

다음 코드 예제는 Amazon API Gateway에서 호출하는 AWS Lambda 함수를 생성하는 방법을 보여줍니다.

Java

SDK for Java 2.x

Lambda Java AWS Lambda 런타임 API를 사용하여 함수를 생성하는 방법을 보여 줍니다. 이 예제는 다양한 AWS 서비스를 호출하여 특정 사용 사례를 수행합니다. 이 예제에서는 Amazon API Gateway에서 호출한 Lambda 함수를 생성하여 작업 기념일에 대한 Amazon DynamoDB 테이블을 스캔하고 Amazon Simple Notification Service(Amazon SNS)를 사용하여 직원에게 1주년 기념일을 축하하는 문자 메시지를 전송하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 전체 예제를 참조하십시오. GitHub

이 예시에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

JavaScript

JavaScript (v3) 용 SDK

JavaScript Lambda AWS Lambda 런타임 API를 사용하여 함수를 생성하는 방법을 보여 줍니다. 이 예제는 다양한 AWS 서비스를 호출하여 특정 사용 사례를 수행합니다. 이 예제에서는 Amazon API Gateway에서 호출한 Lambda 함수를 생성하여 작업 기념일에 대한 Amazon DynamoDB 테이블을 스캔하고 Amazon Simple Notification Service(Amazon SNS)를 사용하여 직원에게 1주년 기념일을 축하하는 문자 메시지를 전송하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 전체 예제를 참조하십시오. GitHub

이 예시는 AWS SDK for JavaScript v3 개발자 안내서에서도 확인할 수 있습니다.

이 예제에서 사용되는 서비스

- API Gateway
- DynamoDB
- Lambda
- Amazon SNS

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오AWS SDK와 함께 Amazon SNS 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

예약된 이벤트를 사용하여 Lambda 함수 호출

다음 코드 예제는 Amazon EventBridge 예약 이벤트에 의해 호출되는 AWS Lambda 함수를 생성하는 방법을 보여줍니다.

Java

SDK for Java 2.x

AWS Lambda 함수를 호출하는 Amazon EventBridge 예약 이벤트를 생성하는 방법을 보여 줍니다. cron 표현식을 사용하여 Lambda 함수가 호출되는 시기를 EventBridge 스케줄링하도록 구성합니다. 이 예제에서는 Lambda Java 런타임 API를 사용하여 Lambda 함수를 생성합니다. 이 예제에서는 다양한 AWS 서비스를 호출하여 특정 사용 사례를 수행합니다. 이 예제에서는 1주년 기념일에 직원에게 축하하는 모바일 문자 메시지를 전송하는 앱을 생성하는 방법을 보여 줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 전체 예제를 참조하십시오. GitHub

이 예제에서 사용되는 서비스

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

JavaScript (v3) 용 SDK

AWS Lambda 함수를 호출하는 Amazon EventBridge 예약 이벤트를 생성하는 방법을 보여 줍니다. cron 표현식을 사용하여 Lambda 함수가 호출되는 시기를 EventBridge 스케줄링하도록 구성합니다. 이 예시에서는 Lambda 런타임 API를 사용하여 Lambda 함수를 생성합니다. JavaScript 이 예제는 다양한 AWS 서비스를 호출하여 특정 사용 사례를 수행합니다. 이 예제에서는 1주년 기념일에 직원에게 축하하는 모바일 문자 메시지를 전송하는 앱을 생성하는 방법을 보여줍니다.

전체 소스 코드와 설정 및 실행 방법에 대한 지침은 전체 예제를 참조하십시오. GitHub

이 예시는 AWS SDK for JavaScript v3 개발자 안내서에서도 확인할 수 있습니다.

이 예제에서 사용되는 서비스

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

AWS SDK 개발자 가이드 및 코드 예제의 전체 목록은 을 참조하십시오AWS SDK와 함께 Amazon SNS 사용. 이 주제에는 시작하기에 대한 정보와 이전 SDK 버전에 대한 세부 정보도 포함되어 있습니다.

Amazon SNS 보안

이 섹션에서는 Amazon SNS 보안, 인증 및 액세스 제어, Amazon SNS 액세스 정책 언어에 대한 정보를 제공합니다.

주제

데이터 보호

AWS 공동 책임 모델은 Amazon Simple Notification Service의 데이터 보호에 적용됩니다. 이 모델에서 설명하는 것처럼 AWS는 모든 AWS 클라우드를 실행하는 글로벌 인프라를 보호할 책임이 있습니다. 이 인프라에서 호스팅되는 콘텐츠에 대한 제어를 유지하는 것은 사용자의 책임입니다. 이 콘텐츠에는 사용하는 AWS 서비스에 대한 보안 구성 및 관리 작업이 포함됩니다. 데이터 프라이버시에 대한 자세한 내용은 데이터 프라이버시 FAQ를 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 Identity and Access Management(IAM)를 사용하여 개별 사용자 계정을 설정하는 것이 좋습니다. 이러한 방식에서는 각 사용자에게 자신의 직무를 충실히 이행하는 데 필요한 권한만 부여됩니다. 또한 다음과 같은 방법으로 데이터를 보호하는 것이 좋습니다.

- 각 계정마다 멀티 팩터 인증(MFA)을 사용합니다.
- SSL/TLS를 사용하여 AWS 리소스와 통신합니다. TLS 1.2 이상을 권장합니다.
- 로 API 및 사용자 활동 로깅을 설정합니다AWS CloudTrail
- AWS 암호화 솔루션을 AWS 서비스 내의 모든 기본 보안 컨트롤과 함께 사용합니다.
- Amazon S3에 저장된 개인 데이터를 검색하고 보호하는 데 도움이 되는 Amazon Macie와 같은 고급 관리형 보안 서비스를 사용합니다.

- 명령줄 인터페이스 또는 API를 통해 AWS에 액세스할 때 FIPS 140-2 검증된 암호화 모듈이 필요한 경우 FIPS 엔드포인트를 사용합니다. 사용 가능한 FIPS 엔드포인트에 대한 자세한 내용은 Federal Information Processing Standard(FIPS) 140-2를 참조하세요.
- 메시지 대이터 보호
  - 메시지 데이터 보호는 Amazon SNS 새로운 주요 기능임
  - MDP를 사용하여 메시지에서 기밀 또는 중요한 정보 스캔
  - 주제를 통해 이동하는 모든 콘텐츠에 메시지 감사 기능 제공
  - 주제에 게시된 메시지와 주제에서 전달된 메시지에 대한 콘텐츠 액세스 제어 제공

> **Important**
>
> 고객의 이메일 주소와 같은 기밀 정보나 중요한 정보는 태그나 [이름(Name)] 필드와 같은 자유 양식 필드에 입력하지 않는 것이 좋습니다. 여기에는 콘솔, API, AWS CLI, AWS SDK를 사용하여 Amazon SNS 또는 기타 Amazon Web Services로 작업하는 경우가 포함됩니다. 이름에 사용되는 태그 또는 자유 형식 필드에 입력하는 모든 데이터는 청구 또는 진단 로그에 사용될 수 있습니다. 외부 서버에 URL을 제공할 때 해당 서버에 대한 요청을 검증하기 위해 자격 증명 정보를 URL에 포함시켜서는 안됩니다.

다음 섹션에서는 Amazon SNS에서의 데이터 보호에 대한 추가 정보를 제공합니다.

주제

- 데이터 암호화
- 인터네트워크 트래픽 개인 정보 보호
- 메시지 데이터 보호 보안

데이터 암호화

데이터 보호란 전송 중(Amazon SNS 안팎으로 데이터가 이동 중)과 유휴 시(Amazon SNS 데이터 센터의 디스크에 데이터가 저장된 동안) 데이터를 보호하는 것을 말합니다. Secure Sockets Layer(SSL) 또는 클라이언트 측 암호화를 사용하여 전송 중인 데이터를 보호할 수 있습니다. 기본적으로 Amazon SNS는 디스크 암호화를 사용하여 메시지와 파일을 저장합니다. 메시지를 데이터 센터의 암호화된 파일 시스템에 저장하기 전에 Amazon SNS에 메시지를 암호화하도록 요청하여 저장된 데이터를 보호할 수 있습니다. Amazon SNS는 최적화된 데이터 암호화를 위해 SSE를 사용할 것을 권장합니다.

주제

저장 시 암호화

서버 측 암호화 (SSE) 를 사용하면 () 에서 AWS Key Management Service 관리되는 키를 사용하여 Amazon SNS 주제의 메시지 콘텐츠를 보호함으로써 암호화된 주제에 민감한 데이터를 저장할 수 있습니다. AWS KMS

Amazon SNS가 메시지를 수신하면 SSE가 메시지를 즉시 암호화합니다. 메시지는 암호화된 형태로 저장되며 전송 시에만 복호화됩니다.

- AWS Management Console 또는 AWS SDK for Java(`CreateTopic` 및 `SetTopicAttributes` API 작업을 사용하여 `KmsMasterKeyId` 속성 설정)를 사용하여 SSE를 관리하는 자세한 내용은 Amazon SNS 주제에 대한 서버 측 암호화(SSE) 사용에서 확인하세요.
- AWS CloudFormation(`AWS::SNS::Topic` 리소스를 사용하여 `KmsMasterKeyId` 속성 설정)을 사용하여 암호화된 주제를 생성하는 방법에 대한 자세한 내용은 AWS CloudFormation 사용 설명서를 참조하세요.

> **Important**
>
> SSE를 사용할 수 있는 주제에 대한 모든 요청에서는 HTTPS 및 서명 버전 4를 사용해야 합니다.
> 기타 서비스와 암호화 주제의 호환성에 관한 정보는 서비스 설명서를 참조하세요.
> Amazon SNS는 대칭 암호화 KMS 키만 지원합니다. 다른 유형의 KMS 키를 사용하여 서비스 리소스를 암호화할 수 없습니다. KMS 키가 대칭인지 암호화 키인지 확인하는 것과 관련된 도움말은 비대칭 KMS 키 식별을 참조하세요.

AWS KMS는 클라우드에 맞게 확장된 키 관리 시스템을 제공하기 위해 안전하고 가용성이 높은 하드웨어 및 소프트웨어를 결합합니다. Amazon SNS를 AWS KMS에서 사용하는 경우, 메시지 데이터를 암호화하는 데이터 키 또한 암호화되며 데이터 키가 보호하는 데이터와 함께 저장됩니다.

AWS KMS를 사용하면 다음과 같은 이점이 있습니다.

- AWS KMS key를 직접 생성하고 관리할 수 있습니다.
- 또한 각 계정 및 리전에 고유한 Amazon SNS용 AWS 관리형 KMS 키를 사용할 수도 있습니다.
- AWS KMS 보안 표준은 암호화 관련 규정 준수 요구 사항을 충족하는 데 도움이 될 수 있습니다.

자세한 정보는 AWS Key Management Service 개발자 안내서.의 AWS Key Management Service란 무엇입니까?를 참조하세요.

주제

- 암호화 범위
- 주요 용어

암호화 범위

SSE는 Amazon SNS 주제에서 메시지의 본문을 암호화합니다.

SSE는 다음을 암호화하지 않습니다.

- 주제 메타데이터(주제 이름 및 속성)
- 메시지 메타데이터(주제, 메시지 ID, 타임스탬프 및 속성)
- 데이터 보호 정책
- 주제별 지표

> **Note**
>
> - 주제 암호화가 활성화된 후 전송되는 경우에만 메시지가 암호화됩니다. Amazon SNS 백로그된 메시지를 암호화하지 않습니다.
> - 암호화된 메시지는 해당 주제 암호화가 비활성화된 경우에만 암호화된 상태를 유지합니다.

주요 용어

다음 주요 용어 설명은 SSE 기능을 보다 정확하게 이해하는 데 도움이 될 수 있습니다. 자세한 내용은 Amazon Simple Notification Service API 참조를 참조하세요.

데이터 키

Amazon SNS 메시지 내용의 암호화를 담당하는 데이터 암호화 키(DEK)입니다.

자세한 정보는 AWS Key Management Service 개발자 안내서의 데이터 키 및 AWS Encryption SDK 개발자 안내서의 봉투 암호화를 참조하세요.

AWS KMS key ID

사용자 계정이나 다른 계정에 있는 AWS KMS key 또는 사용자 지정 AWS KMS의 별칭, 별칭 ARN, 키 ID 또는 키 ARN입니다. Amazon SNS용 AWS 관리형 AWS KMS의 별칭이 항상 `alias/aws/sns`인 반면, 사용자 지정 AWS KMS의 별칭은 예를 들어 `alias/MyAlias`일 수 있습니다. 이러한 AWS KMS 키를 사용하여 Amazon SNS 주제의 메시지를 보호할 수 있습니다.

> **Note**
>
> 다음 사항에 유의하세요:
>
> - 주제에 대한 Amazon SNS용 AWS 관리형 KMS를 지정하는 데 AWS Management Console을 처음으로 사용하면 AWS KMS에서 Amazon SNS용 AWS 관리형 KMS를 생성합니다.
> - 또는 SSE가 활성화된 주제에서 `Publish` 작업을 처음으로 사용하는 경우에는 AWS KMS에서 Amazon SNS용 AWS 관리형 KMS를 생성합니다.

AWS KMS 키를 생성하고, AWS KMS 키 사용 방법을 제어하는 정책을 정의하고, AWS KMS 콘솔 또는 `CreateKey` AWS KMS 작업의 AWS KMS keys 섹션을 참조하여 AWS KMS 사용을 감사할 수 있습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서의 AWS KMS keys 및 키 생성을 참조하세요. AWS KMS식별자의 더 많은 예는 API 참조를 참조하십시오 KeyId. AWS Key Management Service AWS KMS 식별자 찾기에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 키 ID 및 ARN 찾기를 참조하세요.

> **Important**
>
> AWS KMS 사용에 따르는 추가 요금이 있습니다. 자세한 정보는 AWS KMS 비용 추정 및 AWS Key Management Service 요금을 참조하세요.

키 관리

다음 섹션에서는 AWS Key Management Service(AWS KMS)의 관리형 키 작업에 대해 설명합니다. 다음에 대한 자세한 정보

> **Note**
>
> Amazon SNS는 대칭 암호화 KMS 키만 지원합니다. 다른 유형의 KMS 키를 사용하여 서비스 리소스를 암호화할 수 없습니다. KMS 키가 대칭인지 암호화 키인지 확인하는 것과 관련된 도움말은 비대칭 KMS 키 식별을 참조하세요.

주제

- AWS KMS 비용 추정
- AWS KMS 권한 구성
- AWS KMS 오류

AWS KMS 비용 추정

요금을 예측하고 AWS 청구 내역을 효과적으로 이해하려면 Amazon SNS의 AWS KMS key 사용 빈도를 알아보면 됩니다.

> **Note**
>
> 다음 공식으로 예상되는 비용을 거의 정확하게 짐작할 수 있지만, Amazon SNS의 분산 특성상 실제 비용은 더 높을 수 있습니다.

주제당 API 요청 수(R)를 계산하려면 다음 수식을 사용하세요.

```
R = B / D * (2 * P)
```

여기서 B는 청구 기간(초)입니다.

D는 데이터 키 재사용 기간입니다(초 단위—Amazon SNS는 최대 5 분 동안 데이터 키를 재사용함).

P는 Amazon SNS 주제로 보내는 게시 보안 주체의 수입니다.

다음은 계산 예제입니다. 정확한 요금 정보는 AWS Key Management Service 요금을 참조하세요.

예 1: 게시자 1명 및 주제 1개에 대한 AWS KMS API 호출 수 계산

이 예에서는 다음과 같이 가정합니다.

- 청구 기간은 1월 1일부터 31일까지입니다(2,678,400초).
- 데이터 키 재사용 기간은 5분(300초)입니다.
- 1개의 주제가 있습니다.
- 1개의 게시 보안 주체가 있습니다.

```
2,678,400 / 300 * (2 * 1) = 17,856
```

예 2: 여러 게시자 및 주제 2개에 대한 AWS KMS API 호출 수 계산

이 예에서는 다음과 같이 가정합니다.

- 청구 기간은 2월 1일부터 28일까지입니다(2,419,200초).
- 데이터 키 재사용 기간은 5분(300초)입니다.
- 2개의 주제가 있습니다.
- 첫 번째 주제에는 3개의 게시 보안 주체가 있습니다.
- 두 번째 주제에는 5개의 게시 보안 주체가 있습니다.

```
(2,419,200 / 300 * (2 * 3)) + (2,419,200 / 300 * (2 * 5)) = 129,024
```

AWS KMS 권한 구성

SSE를 사용하려면 먼저 AWS KMS key 정책을 구성하여 주제 암호화 및 메시지 암호화 및 해독을 허용하도록 해야 합니다. AWS KMS 권한에 대한 예제 및 자세한 정보는 AWS Key Management Service 개발자 안내서의 AWS KMS API 권한: 작업 및 리소스 참조를 참조하세요. Amazon SNS 주제에 서버 측 암호화를 설정하는 방법에 대한 자세한 내용은 Amazon SNS 주제에 서버 측 암호화 설정 섹션을 참조하세요.

> **Note**
>
> IAM 정책을 사용하여 대칭 암호화 KMS 키 권한을 관리할 수도 있습니다. 자세한 정보는 AWS KMS에 IAM 정책 사용을 참조하세요.

> Amazon SNS와 송수신하기 위한 전역 권한을 구성해도 되지만, AWS KMS는 IAM 정책의 `Resource` 섹션에 구체적인 리전별 KMS의 전체 ARN을 명시적으로 지정할 것을 요구합니다.

또한 AWS KMS key의 키 정책에서 필수 권한을 허용하는지 확인해야 합니다. 이렇게 하려면 Amazon SNS에서 암호화된 메시지를 생산하고 소비하는 보안 주체를 KMS 키 정책에서 사용자로 지정해야 합니다.

또는 Amazon SNS에서 암호화된 메시지를 수신하기 위해 게시 및 구독하는 보안 주체에 할당된 IAM 정책에서 필요한 AWS KMS 작업과 KMS ARN을 지정할 수 있습니다. 자세한 정보는 AWS Key Management Service 개발자 안내서의 AWS KMS에 대한 액세스 관리를 참조하세요.

Amazon SNS 주제에 대한 고객 관리형 키를 선택하고 조건 키 `kms:ResourceAliases`와 함께 IAM 정책 또는 KMS 키 정책을 사용하여 KMS 키에 대한 액세스를 제어하도록 별칭을 사용하는 경우 선택한 고객 관리형 키에도 별칭이 연결되어 있어야 합니다. 별칭을 사용하여 KMS 키에 대한 액세스를 제어하는 방법에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 별칭을 사용하여 KMS 키에 대한 액세스 제어를 참조하세요.

사용자가 SSE를 사용하여 주제에 메시지 보내기 허용

게시자에게 AWS KMS key에 대한 `kms:GenerateDataKey*` 및 `kms:Decrypt` 권한이 있어야 합니다.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "kms:GenerateDataKey*",
      "kms:Decrypt"
    ],
    "Resource": "arn:aws:kms:us-
east-2:123456789012:key/1234abcd-12ab-34cd-56ef-1234567890ab"
  }, {
    "Effect": "Allow",
    "Action": [
      "sns:Publish"
    ],
    "Resource": "arn:aws:sns:*:123456789012:MyTopic"
  }]
}
```

AWS 서비스 및 암호화된 주제의 이벤트 소스 간 호환성 활성화

여러 AWS 서비스는 Amazon SNS 주제에 이벤트를 게시합니다. 이러한 이벤트 소스가 암호화된 주제와 연동되도록 하려면 다음 단계를 수행해야 합니다.

1. 고객 관리형 키를 사용합니다. 자세한 내용은 AWS Key Management Service 개발자 안내서에서 키 생성을 참조하세요.
2. AWS 서비스에서 `kms:GenerateDataKey*` 및 `kms:Decrypt` 권한을 갖도록 하려면 KMS 정책에 다음 문을 추가합니다.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "Service": "service.amazonaws.com"
    },
    "Action": [
      "kms:GenerateDataKey*",
      "kms:Decrypt"
    ],
    "Resource": "*"
  }]
}
```

이벤트 소스	서비스 보안 주체
Amazon CloudWatch	cloudwatch.amazonaws.com
Amazon CloudWatch Events	events.amazonaws.com
AWS CodeCommit	codecommit.amazonaws.com
AWS CodeStar	codestar-notifications.amazonaws.com
AWS Database Migration Service	dms.amazonaws.com
AWS Directory Service	ds.amazonaws.com
Amazon DynamoDB	dynamodb.amazonaws.com

이벤트 소스	서비스 보안 주체
Amazon Inspector	`inspector.amazonaws.com`
Amazon Redshift	`redshift.amazonaws.com`
Amazon RDS	`events.rds.amazonaws.com`
Amazon S3 Glacier	`glacier.amazonaws.com`
Amazon Simple Email Service	`ses.amazonaws.com`
Amazon Simple Storage Service	`s3.amazonaws.com`
AWS Snowball	`importexport.amazonaws.com`
AWS Systems Manager Incident Manager	AWS Systems Manager Incident Manager는 다음 2가지 서비스 원칙으로 구성됩니다. `ssm-incidents.amazonaws.com` ; `ssm-contacts.amazonaws.com`

> **Note**
>
> 일부 Amazon SNS 이벤트 소스는 AWS KMS key 정책에서 IAM 역할(서비스 보안 주체 아님)을 제공해야 합니다.
>
> - Amazon EC2 Auto Scaling
> - Amazon Elastic Transcoder
> - AWS CodePipeline
> - AWS Config
> - AWS Elastic Beanstalk
> - AWS IoT
> - EC2 Image Builder

3. `aws:SourceAccount` 및 `aws:SourceArn` 조건 키를 KMS 리소스 정책에 추가하여 혼동된 대리자 공격으로부터 KMS 키를 더 보호합니다. 각 경우에 대한 정확한 세부 정보는 서비스별 설명서 목록(위)을 참조하세요.

> **Important**
>
> EventBridge-암호화된 주제에는 AWS KMS 정책에 `aws:SourceAccount` 및 `aws:SourceArn` 추가가 지원되지 않습니다.

```
{
  "Effect": "Allow",
  "Principal": {
    "Service": "service.amazonaws.com"
  },
  "Action": [
    "kms:GenerateDataKey*",
    "kms:Decrypt"
  ],
  "Resource": "*",
  "Condition": {
    "StringEquals": {
      "aws:SourceAccount": "customer-account-id"
    },
    "ArnLike": {
      "aws:SourceArn": "arn:aws:service:region:customer-account-id:resource-
type:customer-resource-id"
    }
  }
}
```

4. KMS를 사용하여 주제의 SSE를 활성화합니다.
5. 암호화된 주제의 ARN을 이벤트 소스에 제공합니다.

AWS KMS 오류

Amazon SNS 및 AWS KMS로 작업할 때 오류가 발생할 수 있습니다. 아래 목록에서 이러한 오류와 문제 해결 방법을 설명합니다.

KMSAccessDeniedException

암호화 텍스트가 존재하지 않는 키 또는 액세스 권한이 없는 키를 참조합니다.

HTTP 상태 코드: 400

KMSDisabledException

지정한 KMS가 활성화되지 않아서 요청이 거부되었습니다.

HTTP 상태 코드: 400

KMSInvalidStateException

지정한 리소스의 상태가 이 요청에 유효하지 않아서 요청이 거부되었습니다. 자세한 내용은 AWS Key Management Service 개발자 안내서의 AWS KMS keys의 키 상태를 참조하세요.

HTTP 상태 코드: 400

KMSNotFoundException

지정한 엔터티 또는 리소스를 찾을 수 없으므로 요청이 거부되었습니다.

HTTP 상태 코드: 400

KMSOptInRequired

AWS 액세스 키 ID는 서비스에 대한 구독이 필요합니다.

HTTP 상태 코드: 403

KMSThrottlingException

요청 제한 때문에 요청이 거부되었습니다. 제한에 대한 자세한 정보는 AWS Key Management Service 개발자 안내서의 할당량을 참조하세요.

HTTP 상태 코드: 400

Amazon SNS 주제에 대한 서버 측 암호화(SSE) 사용

서버 측 암호화(SSE)를 사용하면 암호화된 주제에서 민감한 데이터를 저장할 수 있습니다. SSE는 AWS Key Management Service(AWS KMS)에서 관리되는 키를 사용하여 Amazon SNS 주제의 메시지 내용을 보호합니다. Amazon SNS에 서버 측 암호화 사용에 대한 자세한 정보는 저장 시 암호화 섹션을 참조하세요. AWS KMS 키 생성에 대한 자세한 내용은 AWS Key Management Service 개발자 안내서의 키 생성을 참조하세요.

> **Important**
>
> SSE를 사용할 수 있는 주제에 대한 모든 요청에서는 HTTPS 및 서명 버전 4를 사용해야 합니다.

AWS Management Console을 사용하여 Amazon SNS 주제에 대해 서버 측 암호화(SSE) 활성화

1. Amazon SNS 콘솔에 로그인합니다.
2. 탐색 창에서 주제(Topics)를 선택합니다.
3. 주제 페이지에서 주제를 선택하고 작업, 편집을 선택합니다.
4. 암호화 섹션을 확장하고 다음을 수행합니다.

   a. 암호화 활성을 선택합니다.

   b. AWS KMS 키를 지정합니다. 자세한 내용은 주요 용어 섹션을 참조하세요.

      각 KMS 유형에 대해 설명(Description), 계정(Account), KMS ARN이 표시됩니다.

> **Important**
>
> 해당 KMS의 소유자가 아니거나 `kms:ListAliases` 및 `kms:DescribeKey` 권한이 없는 계정으로 로그인하는 경우 Amazon SNS 콘솔에서 해당 KMS에 대한 정보를 볼 수 없습니다.
> KMS의 소유자에게 이 권한을 부여해 달라고 요청해야 합니다. 자세한 정보는 AWS Key Management Service 개발자 안내서의 AWS KMS API 권한: 작업 및 리소스 참조를 참조하세요.

- Amazon SNS용 AWS 관리형 KMS(기본값) alias/aws/sns는 기본적으로 선택됩니다.

> **Note**
>
> 다음 사항에 유의하세요.
>
> - 주제에 대한 Amazon SNS용 AWS 관리형 KMS를 지정하는 데 AWS Management Console을 처음으로 사용하면 AWS KMS에서 Amazon SNS용 AWS 관리형 KMS를 생성합니다.
> - 또는 SSE가 활성화된 주제에서 `Publish` 작업을 처음으로 사용하는 경우에는 AWS KMS에서 Amazon SNS용 AWS 관리형 KMS를 생성합니다.

- AWS 계정에서 사용자 지정 KMS를 사용하려면 KMS 키 필드를 선택한 다음 목록에서 사용자 지정 KMS를 선택합니다.

> **Note**
>
> 사용자 지정 KMS 생성에 대한 지침은 AWS Key Management Service 개발자 안내서의 키 생성을 참조하세요.

- AWS 계정 또는 다른 AWS 계정에서 사용자 지정 KMS ARN을 사용하려면 KMS 키 필드에 입력합니다.

5. Save changes(변경 사항 저장)를 선택합니다.

   주제에 대해 SSE가 활성화되고 ***MyTopic*** 페이지가 표시됩니다.

   주제의 암호화 상태, AWS 계정, 고객 마스터 키(CMK), CMK ARN 및 설명이 암호화 탭에 표시됩니다.

Amazon SNS 주제에 서버 측 암호화 설정

KMS 키를 생성할 때는 다음 KMS 키 정책을 사용합니다.

```
{
    "Effect": "Allow",
    "Principal": {
        "Service": "service.amazonaws.com"
     },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*",
    "Condition": {
        "ArnLike": {
            "aws:SourceArn": "arn:aws:service:region:customer-account-id:resource-
type/customer-resource-id"
        },
        "StringEquals": {
            "kms:EncryptionContext:aws:sns:topicArn":
 "arn:aws:sns:your_region:customer-account-id:your_sns_topic_name"
        }
    }
}
```

암호화된 Amazon SQS 대기열 구독이 포함된 Amazon SNS 주제에 대해 서버 측 암호화(SSE) 사용

주제에 대해 서버 측 암호화(SSE)를 활성화하여 데이터를 보호할 수 있습니다. Amazon SNS가 암호화된 Amazon SQS 대기열로 메시지를 전송하도록 허용하려면 Amazon SQS 대기열과 연결된 고객 관리형 키에는 AWS KMS API 작업 `GenerateDataKey` 및 `Decrypt`에 Amazon SNS 서비스 보안 주체에 액세스 권한을 부여하는 정책 명령문이 있어야 합니다. SSE 사용에 관한 자세한 정보는 저장 시 암호화에서 확인하세요.

다음 페이지에서는 AWS Management Console을 사용하여 암호화된 Amazon SQS 대기열이 구독하는 Amazon SNS 주제에 대해 SSE를 사용하는 방법을 보여줍니다.

1단계: 사용자 지정 KMS 키 생성

1. 최소한 `AWSKeyManagementServicePowerUser` 정책이 있는 사용자로 AWS KMS 콘솔 에 로그인합니다.
2. Create key(키 생성)를 선택합니다.
3. 대칭 암호화 KMS 키를 생성하려면 키 유형(Key type)에 대칭(Symmetric)을 선택합니다.

   AWS KMS 콘솔에서 비대칭 KMS 키를 생성하는 방법에 대한 자세한 내용은 비대칭 KMS 키 생성(콘솔)을 참조하세요.
4. 키 사용(Key usage)에서 암호화 및 해독(Encrypt and decrypt) 옵션이 선택됩니다.

   MAC 코드를 생성 및 확인하는 KMS 키를 생성하는 방법에 대한 자세한 내용은 HMAC KMS 키 생성을 참조하세요.

   고급 옵션에 대한 자세한 내용은 특수 용도 키를 참조하세요.
5. 다음(Next)을 선택합니다.
6. KMS 키의 별칭을 입력합니다. 별칭은 **`aws/`**로 시작할 수 없습니다. **`aws/`** 접두사는 Amazon Web Services에서 계정에서 AWS 관리형 키를 나타내기 위해 예약한 것입니다.

> **Note**
>
> 별칭을 추가, 삭제 또는 업데이트하면 KMS 키에 대한 권한을 허용하거나 거부할 수 있습니다. 자세한 내용은 AWS KMS의 ABAC 및 별칭을 사용하여 KMS 키에 대한 액세스 제어를 참조하세요.

별칭은 KMS 키를 식별하는 데 사용할 수 있는 표시 이름입니다. 보호하고자 하는 데이터의 유형 또는 KMS 키와 함께 사용할 애플리케이션을 나타내는 별칭을 선택하는 것이 좋습니다.

AWS Management Console에서 KMS 키를 생성할 때 별칭이 필요합니다. 이들은 CreateKey 작업을 사용할 때 선택 사항입니다.

7. (선택 사항) KMS 키에 대한 설명을 입력합니다.

   키 상태가 `Pending Deletion` 또는 `Pending Replica Deletion`이 아닌 한 지금 설명을 추가하거나 언제든지 설명을 업데이트할 수 있습니다. 기존 고객 관리 키의 설명을 추가, 변경 또는 삭제하려면 AWS Management Console에서 설명을 편집하거나 UpdateKeyDescription 작업을 사용합니다.

8. (선택 사항) 태그 키와 태그 값(선택)을 입력합니다. KMS 키에 두 개 이상의 태그를 추가하려면 태그 추가(Add tag)를 선택합니다.

   > **Note**
   >
   > KMS 키에 태그를 지정하거나 해제하면 KMS 키에 대한 권한을 허용하거나 거부할 수 있습니다. 자세한 내용은 AWS KMS의 ABAC 및 태그를 사용하여 KMS 키에 대한 액세스 제어를 참조하세요.

   AWS 리소스에 태그를 추가하면 AWS에서 사용 내역 및 비용을 태그별로 집계한 비용 할당 보고서를 생성합니다. KMS 키에 대한 액세스를 제어하는 데에도 태그를 사용할 수 있습니다. KMS 키 태그 지정에 대한 자세한 내용은 키 태그 지정 및 AWS KMS의 ABAC를 참조하세요.

9. 다음(Next)을 선택합니다.

10. KMS 키를 관리할 수 있는 IAM 사용자 및 역할을 선택합니다.

    > **Note**
    >
    > 이 키 정책은 AWS 계정에 이 KMS 키에 대한 완전한 제어 권한을 부여합니다. 계정 관리자가 IAM 정책을 사용하여 KMS 키를 관리할 수 있는 권한을 다른 보안 주체에게 부여하도록 허용합니다. 자세한 내용은 기본 키 정책을 참조하세요.

> IAM 모범 사례는 장기 보안 인증 정보가 있는 IAM 사용자의 사용을 장려하지 않습니다. 가능할 경우, 임시 보안 인증 정보를 제공하는 IAM 역할을 사용하세요. 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

11. (선택 사항) 선택한 IAM 사용자와 역할이 페이지 하단의 키 삭제 섹션에서 이 KMS 키를 삭제하지 못하도록 하려면 키 관리자가 이 키를 삭제하도록 허용(Allow key administrators to delete this key) 확인란의 선택을 취소합니다.
12. 다음(Next)을 선택합니다.
13. 암호화 작업에서 키를 사용할 수 있는 IAM 사용자 및 역할을 선택합니다. 다음(Next)을 선택합니다.
14. Review and edit key policy(키 정책 검토 및 편집) 페이지에서 다음 문을 키 정책에 추가한 다음 완료를 선택합니다.

```
{
    "Sid": "Allow Amazon SNS to use this key",
    "Effect": "Allow",
    "Principal": {
        "Service": "sns.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey*"
    ],
    "Resource": "*"
}
```

새 고객 관리형 키가 키 목록에 나타납니다.

2단계: 암호화된 Amazon SNS 주제 생성

1. Amazon SNS 콘솔에 로그인합니다.
2. 탐색 창에서 주제(Topics)를 선택합니다.
3. 주제 생성을 선택합니다.
4. 새로운 주제 생성 페이지에서 이름에 주제 이름을 입력하고(예: `MyEncryptedTopic`) 주제 생성을 선택합니다.
5. 암호화 섹션을 확장하고 다음을 수행합니다.

a. Enable server-side encryption(서버 측 암호화 활성화)을 선택합니다.

b. 고객 관리형 키를 지정합니다. 자세한 내용은 주요 용어 섹션을 참조하세요.

각 고객 관리 키 유형에 대해 설명, 계정 및 고객 관리형 키 ARN이 표시됩니다.

> **Important**
>
> 해당 고객 관리형 키의 소유자가 아니거나 `kms:ListAliases` 및 `kms:DescribeKey` 권한이 없는 계정으로 로그인하는 경우 Amazon SNS 콘솔에서 해당 고객 관리형 키에 대한 정보를 볼 수 없습니다.
> 고객 관리형 키의 소유자에게 이 권한을 부여해 달라고 요청해야 합니다. 자세한 정보는 AWS Key Management Service 개발자 안내서의 AWS KMS API 권한: 작업 및 리소스 참조를 참조하세요.

c. 고객 관리형 키에서 이전에 생성한 MyCustomKey를 선택한 다음, 서버 측 암호화 활성화를 선택합니다.

6. Save changes(변경 사항 저장)를 선택합니다.

주제에 대해 SSE가 활성화되고 MyTopic 페이지가 표시됩니다.

주제의 암호화 상태, AWS 계정, 고객 관리형 키, 고객 관리형 키 ARN 및 설명이 암호화 탭에 표시됩니다.

암호화된 새 주제가 주제 목록에 나타납니다.

3단계: 암호화된 Amazon SQS 대기열 생성 및 구독

1. Amazon SQS 콘솔에 로그인합니다.
2. 새로운 대기열 생성을 선택합니다.
3. 새로운 대기열 생성 페이지에서 다음을 수행합니다.

a. 대기열 이름을 입력합니다(예: `MyEncryptedQueue1`).

b. 표준 대기열을 선택하고 대기열 구성을 선택합니다.

c. SSE 사용을 선택합니다.

d. AWS KMS key의 경우 이전에 생성한 MyCustomKey를 선택한 다음, 대기열 생성을 선택합니다.

4. 두 번째 대기열(예 MyEncryptedQueue2)을 만들려면 프로세스를 반복합니다.

   암호화된 새 대기열이 대기열 목록에 나타납니다.

5. Amazon SQS 콘솔에서 MyEncryptedQueue1 및 MyEncryptedQueue2를 선택하고, 대기열 작업, 대기열에서 SNS 주제 구독을 선택합니다.
6. 주제 구독 대화 상자의 주제 선택에서 MyEncryptedTopic을 선택한 다음 구독을 선택합니다.

   암호화된 주제에 대한 암호화된 대기열의 구독은 주제 구독 결과 대화 상자에 표시됩니다.

7. 확인(OK)을 선택합니다.

4단계: 암호화된 주제에 메시지 게시

1. Amazon SNS 콘솔에 로그인합니다.
2. 탐색 창에서 주제(Topics)를 선택합니다.
3. 주제 목록에서 MyEncryptedTopic을 선택하고 메시지 게시를 선택합니다.
4. 메시지 게시 페이지에서 다음을 수행합니다.

   a. (선택 사항) 메시지 세부 정보 섹션에서 제목을 입력합니다(예: `Testing message publishing`).
   b. 메시지 본문 섹션에서 메시지 본문을 입력합니다(예: `My message body is encrypted at rest.`).
   c. 메시지 게시를 선택합니다.

메시지가 구독 암호화 대기열에 게시됩니다.

5단계: 메시지 전송 확인

1. Amazon SQS 콘솔에 로그인합니다.
2. 대기열 목록에서 MyEncryptedQueue1을 선택한 다음 Send and receive messages(메시지 보내기 및 받기)를 선택합니다.
3. Send and receive messages in MyEncryptedQueue1(MyEncryptedQueue1에서 메시지 보내기 및 받기) 페이지에서 Poll for messages(메시지폴링)를 선택합니다.

   이전에 보낸 메시지가 표시됩니다.

4. 메시지를 보려면 추가 정보를 선택합니다.
5. 모두 마친 후에는 닫기를 선택합니다.

6. MyEncryptedQueue2에 대해 이 프로세스를 반복합니다.

인터네트워크 트래픽 개인 정보 보호

Amazon SNS용 Amazon Virtual Private Cloud(Amazon VPC) 엔드포인트는 VPC 내의 논리적 엔터티로서, Amazon SNS에만 연결을 허용합니다. VPC는 Amazon SNS로 요청을 라우팅하고, 응답을 다시 VPC로 라우팅합니다. 다음 섹션에서는 VPC 엔드포인트 작업 및 VPC 엔드포인트 정책 생성에 대해 설명합니다.

Amazon Virtual Private Cloud(Amazon VPC)를 사용하여 AWS 리소스를 호스트하는 경우, VPC와 Amazon SNS 간에 프라이빗 연결을 설정할 수 있습니다. 이 연결이 있으면 퍼블릭 인터넷을 통해 메시지를 전송하지 않고 Amazon SNS 주제에 메시지를 게시할 수 있습니다.

Amazon VPC란 사용자가 정의한 가상 네트워크에서 AWS 리소스를 시작할 때 사용할 수 있는 AWS 서비스입니다. VPC가 있으면 IP 주소 범위, 서브넷, 라우팅 테이블, 네트워크 게이트웨이 등 네트워크 설정을 제어할 수 있습니다. VPC를 Amazon SNS에 연결하려면 인터페이스 VPC 엔드포인트를 정의하세요. 이 유형의 엔드포인트를 사용하여 VPC를 AWS 서비스에 연결할 수 있습니다. 이 엔드포인트를 이용하면 인터넷 게이트웨이나 NAT(네트워크 주소 변환) 인스턴스 또는 VPN 연결 없이도 Amazon SNS에 안정적이고 확장 가능하게 연결됩니다. 자세한 정보는 Amazon VPC 사용 설명서의 VPC 엔드포인트를 참조하세요.

본 섹션의 내용은 Amazon VPC의 사용자를 대상으로 작성한 것입니다. 자세한 내용과 VPC 생성을 시작하는 방법은 Amazon VPC 사용 설명서의 Amazon VPC 시작하기를 참조하세요.

> **Note**
>
> VPC 엔드포인트에서는 프라이빗 IP 주소에 Amazon SNS 주제를 구독할 수 없습니다.

주제

- Amazon SNS용 Amazon VPC 엔드포인트 생성
- Amazon SNS용 Amazon VPC 엔드포인트 정책 생성
- Amazon VPC에서 Amazon SNS 메시지 게시

Amazon SNS용 Amazon VPC 엔드포인트 생성

Amazon VPC에서 Amazon SNS 주제에 메시지를 게시하려면 인터페이스 VPC 엔드포인트를 생성해야 합니다. 그런 다음 VPC로 관리하는 네트워크 안에 트래픽을 유지하면서 메시지를 해당 주제에 게시할 수 있습니다.

다음 정보를 사용하여 엔드포인트를 생성하고 VPC와 Amazon SNS 간 연결을 테스트합니다. 또는 처음부터 새로 시작하는 경우 이에 대한 안내를 받으려면 Amazon VPC에서 Amazon SNS 메시지 게시에서 확인하세요.

엔드포인트 생성

,, AWS SDK AWS Management Console, Amazon SNS API 또는 를 사용하여 VPC에 아마존 SNS 엔드포인트를 생성할 수 있습니다. AWS CLI AWS CloudFormation

Amazon VPC 콘솔 또는 AWS CLI를 사용한 엔드포인트 생성 및 구성에 대한 자세한 내용은 Amazon VPC 사용 설명서의 인터페이스 엔드포인트 생성을 참조하십시오.

> **Important**
>
> Amazon Virtual Private Cloud는 HTTPS Amazon SNS 엔드포인트에서만 사용할 수 있습니다. 엔드포인트를 생성할 때 VPC를 연결할 서비스로 Amazon SNS를 지정해야 합니다. Amazon VPC 콘솔에서는 리전에 따라 서비스 이름이 다릅니다. 예를 들어 미국 동부(버지니아 북부)를 선택한 경우 서비스 이름은 com.amazonaws.us-east-1.sns입니다.
> Amazon VPC에서 메시지를 보내도록 Amazon SNS를 구성할 때는 프라이빗 DNS를 활성화하고 엔드포인트를 `sns.us-east-2.amazonaws.com` 형식으로 지정해야 합니다.
> 프라이빗 DNS는 `queue.amazonaws.com` 또는 `us-east-2.queue.amazonaws.com` 같은 레거시 엔드포인트를 지원하지 않습니다.

를 사용하여 AWS CloudFormation엔드포인트를 생성하고 구성하는 방법에 대한 자세한 내용은 사용 AWS CloudFormation 설명서의 *AWS::EC2::VPCEndpoint*리소스를 참조하십시오.

VPC와 Amazon SNS 간의 연결 테스트

Amazon SNS에 대한 엔드포인트를 생성한 후에는 VPC에서 Amazon SNS 주제로 메시지를 게시할 수 있습니다. 이 연결을 테스트하려면 다음을 수행합니다.

1. VPC에 있는 Amazon EC2 인스턴스에 연결합니다. 연결에 대한 자세한 내용은 Amazon EC2 설명서의 Linux 인스턴스에 연결 또는 Windows 인스턴스에 연결을 참조하세요.

예를 들어, SSH 클라이언트를 사용하여 Linux 인스턴스에 연결하려면 터미널에서 다음 명령을 실행합니다.

```
$ ssh -i ec2-key-pair.pem ec2-user@instance-hostname
```

위치:

- ec2-key-pair.pem은 인스턴스를 생성할 때 Amazon EC2에서 제공한 키 페어가 들어 있는 파일입니다.
- instance-hostname은 해당 인스턴스의 퍼블릭 호스트 이름입니다. Amazon EC2 콘솔에서 호스트 이름을 가져오려면 인스턴스를 선택하고 원하는 인스턴스를 선택한 다음 퍼블릭 DNS(IPv4)의 값을 찾습니다.

2. 인스턴스에서 AWS CLI를 사용하여 Amazon SNS publish 명령을 사용합니다. 다음 명령으로 간단한 메시지를 주제에 게시할 수 있습니다.

```
$ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"
```

위치:

- AWS aws-region은 주제가 위치한 지역입니다.
- sns-topic-arn주제의 Amazon 리소스 이름 (ARN) 입니다. Amazon SNS 콘솔에서 ARN을 가져오려면 주제를 선택하고 주제를 찾은 다음 ARN 열에서 값을 찾습니다.

메시지가 Amazon SNS에 수신되면 터미널은 다음과 같은 메시지 ID를 인쇄합니다.

```
{
   "MessageId": "6c96dfff-0fdf-5b37-88d7-8cba910a8b64"
}
```

Amazon SNS용 Amazon VPC 엔드포인트 정책 생성

Amazon SNS에 대한 Amazon VPC 엔드포인트 정책을 생성하여 다음을 지정할 수 있습니다.

- 태스크를 수행할 수 있는 보안 주체.
- 수행할 수 있는 작업입니다.
- 태스크를 수행할 있는 리소스.

자세한 정보는 Amazon VPC 사용 설명서의 VPC 엔드포인트를 통해 서비스에 대한 액세스 제어를 참조하세요.

다음 예제 VPC 엔드포인트 정책에서는 IAM 사용자 `MyUser`가 Amazon SNS 주제 `MyTopic`으로 게시할 수 있도록 지정합니다.

```
{
  "Statement": [{
    "Action": ["sns:Publish"],
    "Effect": "Allow",
    "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic",
    "Principal": {
      "AWS": "arn:aws:iam:123456789012:user/MyUser"
    }
  }]
}
```

다음 작업은 거부됩니다.

- `sns:Subscribe` 및 `sns:Unsubscribe`와 같은 다른 Amazon SNS API 작업.
- 다른 IAM 사용자 및 규칙이 이 VPC 엔드포인트를 사용.
- `MyUser`다른 Amazon SNS 주제에 게시하는 .

> **Note**
>
> IAM 사용자는 VPC 외부에서도 다른 Amazon SNS API 작업을 계속 사용할 수 있습니다.

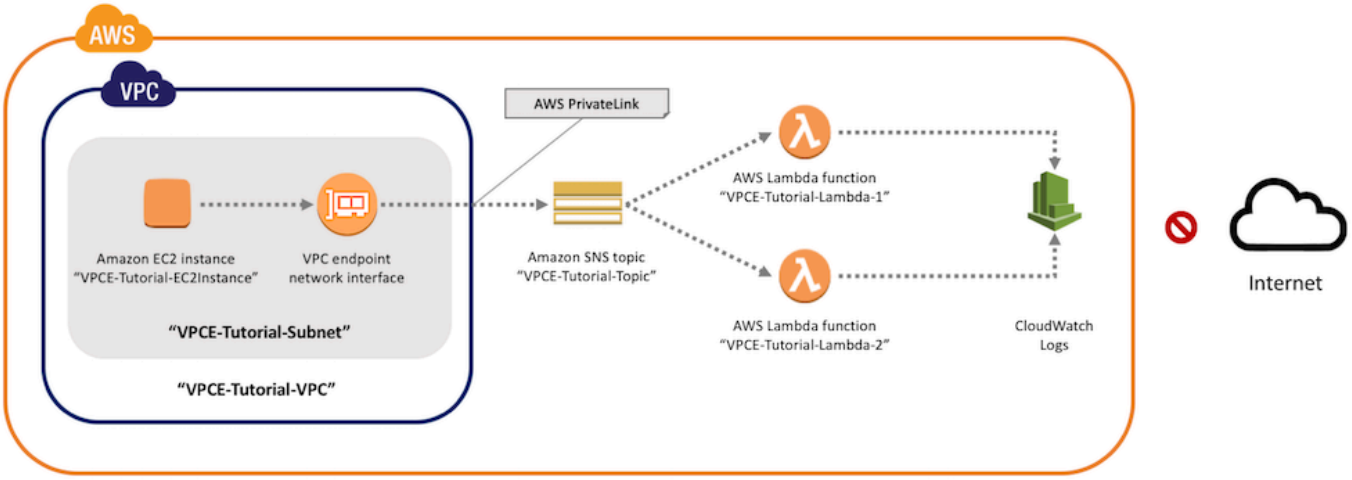
Amazon VPC에서 Amazon SNS 메시지 게시

이 섹션에서는 프라이빗 네트워크에서 메시지를 안전하게 유지하면서 Amazon SNS 주제에 게시하는 방법을 설명합니다. Amazon Virtual Private Cloud(Amazon VPC)에서 호스팅되는 Amazon EC2 인스턴스에서 메시지를 게시합니다. 이 메시지는 퍼블릭 인터넷으로 이동하지 않고 AWS 네트워크 내에 머물러 있습니다. VPC에서 메시지를 비공개로 게시하면 애플리케이션과 Amazon SNS 간 트래픽의 보안을 강화할 수 있습니다. 이 보안은 고객에 관한 개인 식별 정보(PII)를 게시할 때 또는 애플리케이션에 시장 규정이 적용될 때 중요합니다. 예를 들어 미국 건강 보험 이전 및 책임법(HIPAA)을 준수해야 하는 의료 서비스 시스템이나 지불 카드 산업 데이터 보안 표준(PCI DSS)을 준수해야 하는 금융 시스템을 보유한 경우 비공개로 게시하는 것이 도움이 됩니다.

일반적인 단계는 다음과 같습니다.

- AWS CloudFormation 템플릿을 사용하여 AWS 계정에서 임시 프라이빗 네트워크를 자동으로 생성합니다.
- VPC를 Amazon SNS와 연결하는 VPC 엔드포인트를 생성합니다.
- Amazon EC2 인스턴스에 로그인하여 Amazon SNS 주제에 메시지를 비공개로 게시합니다.
- 메시지가 성공적으로 전송되었는지 확인합니다.
- 이 프로세스에서 생성한 리소스를 삭제하여 AWS 계정에 남아 있지 않게 합니다.

아래 다이어그램은 이 단계들을 완료하면 AWS 계정에 생성되는 프라이빗 네트워크를 나타낸 것입니다.



이 네트워크는 Amazon EC2 인스턴스를 포함하는 VPC로 구성됩니다. 인스턴스는 인터페이스 VPC 엔드포인트를 통해 Amazon SNS에 연결합니다. 이러한 엔드포인트 유형은 AWS PrivateLink에서 제공하는 서비스에 연결됩니다. 이 연결이 설정되면 네트워크가 퍼블릭 인터넷에서 연결이 끊어져 있더라도 Amazon EC2 인스턴스에 로그인하여 Amazon SNS 주제에 메시지를 게시할 수 있습니다. 해당 주제는 수신한 메시지를 두 가지 구독 AWS Lambda 함수로 분산합니다. 이 두 함수는 수신한 메시지를 Amazon CloudWatch Logs에 로그합니다.

이 단계를 완료하는 데 약 20분 정도 소요됩니다.

주제

- 시작하기 전에
- 1단계: Amazon EC2 키 페어 생성
- 2단계: AWS 리소스 생성

시작하기 전에

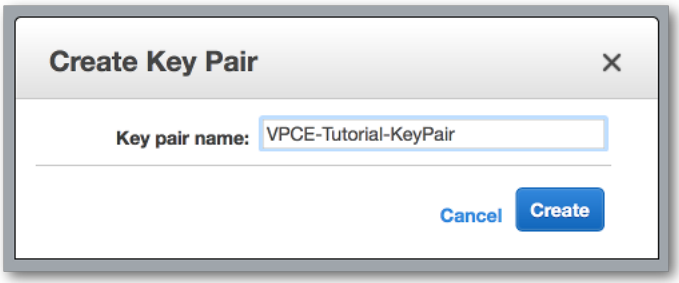
시작하기 전에 Amazon Web Services(AWS) 계정에 로그인합니다. 가입하면 AWS에서 Amazon SNS 및 Amazon VPC를 포함한 모든 서비스에 계정이 자동으로 등록됩니다. 계정을 아직 만들지 않은 경우 https://aws.amazon.com/으로 이동한 다음 무료 계정 생성을 선택합니다.

1단계: Amazon EC2 키 페어 생성

키 페어는 Amazon EC2 인스턴스에 로그인하는 데 사용됩니다. 키 페어는 로그인 정보 암호화에 사용되는 퍼블릭 키와 로그인 정보 암호화 해제에 사용되는 프라이빗 키로 구성됩니다. 키 페어를 생성할 때 프라이빗 키의 사본을 다운로드합니다. 이후에 키 페어는 Amazon EC2 인스턴스에 로그인하는 데 사용됩니다. 로그인하려면 키 페어의 이름을 지정하고 프라이빗 키를 제공해야 합니다.

키 페어를 생성하려면

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 왼쪽의 탐색 메뉴에서 Network & Security 섹션을 찾습니다. 그다음에 키 페어를 선택합니다.
3. 키 페어 생성(Create Key Pair)을 선택합니다.
4. Create Key Pair(키 페어 생성) 창에서 Key pair name(키 페어 이름)에 **`VPCE-Tutorial-KeyPair`**를 입력합니다. 그다음에 생성을 선택합니다.



5. 브라우저에서 프라이빗 키 파일이 자동으로 다운로드됩니다. 안전한 장소에 저장합니다. Amazon EC2는 파일에 `.pem` 확장자를 부여합니다.
6. (선택 사항) Mac 또는 Linux 컴퓨터에서 SSH 클라이언트를 사용하여 인스턴스에 연결하는 경우 사용자만 프라이빗 키 파일을 읽을 수 있도록 다음과 같이 chmod 명령으로 프라이빗 키에 대한 권한을 설정합니다.

   a. 다음과 같이 터미널을 열고 프라이빗 키가 포함된 디렉터리로 이동합니다.

   ```
   $ cd /filepath_to_private_key/
   ```

   b. 다음 명령을 사용하여 권한을 설정합니다.

   ```
   $ chmod 400 VPCE-Tutorial-KeyPair.pem
   ```

2단계: AWS 리소스 생성

인프라를 설정하려면 AWS CloudFormation 템플릿을 사용하세요. 템플릿은 Amazon EC2 인스턴스 및 Amazon SNS 주제와 같은 AWS 리소스를 구축하기 위한 블루프린트 역할을 수행하는 파일입니다. 이 프로세스는 템플릿은 GitHub에서 다운로드할 수 있습니다.

사용자는 AWS CloudFormation에 템플릿을 제공하고 AWS CloudFormation에서는 AWS 계정에 스택으로 필요한 리소스를 프로비저닝합니다. 스택이란 사용자가 하나의 단위로 관리하는 리소스 모음입니다. 이 단계를 완료한 후 AWS CloudFormation을 사용하여 스택에 있는 모든 리소스를 한 번에 삭제할 수 있습니다. 이 리소스는 사용자가 원하지 않는 한 AWS 계정에 남아 있지 않습니다.

이 프로세스에서 사용할 스택으로 다음과 같은 리소스가 있습니다.

- VPC 및 이에 연결된 네트워킹 리소스(예: 서브넷, 보안 그룹, 인터넷 게이트웨이, 라우팅 테이블)
- VPC의 서브넷으로 시작되는 Amazon EC2 인스턴스.
- Amazon SNS 주제.
- 두 가지 AWS Lambda 함수. 이러한 함수는 Amazon SNS 주제에 게시되는 메시지를 수신하고 CloudWatch Logs에서 이벤트를 로그합니다.
- Amazon CloudWatch 지표 및 로그.
- Amazon EC2 인스턴스가 Amazon SNS를 사용하도록 허용하는 IAM 역할과 Lambda 함수가 CloudWatch 로그에 쓸 수 있도록 허용하는 IAM 역할.

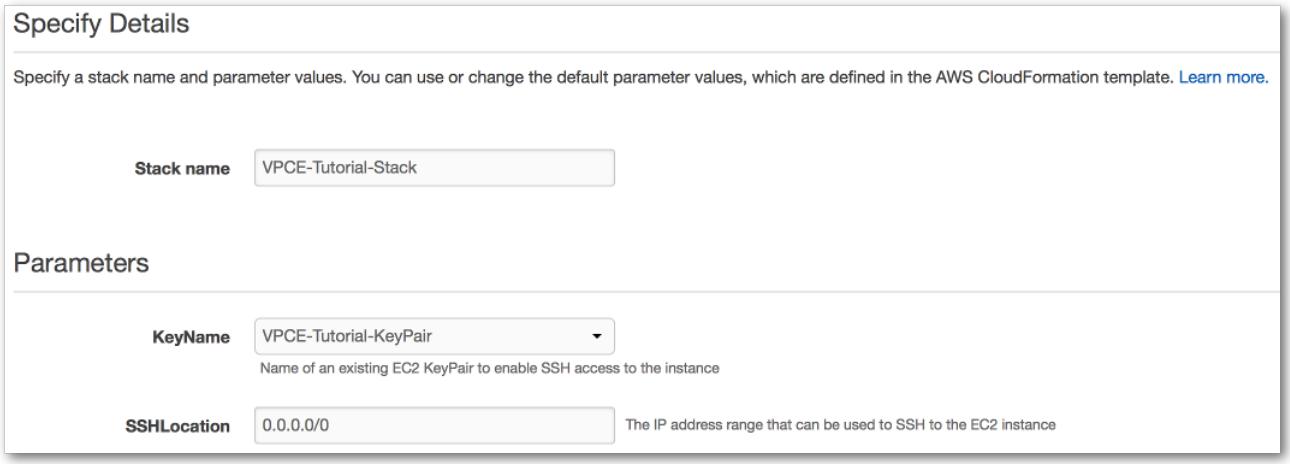
AWS 리소스를 생성하려면

1. GitHub 웹 사이트에서 템플릿 파일을 다운로드합니다.
2. AWS CloudFormation 콘솔에 로그인합니다.
3. 스택 생성을 선택합니다.
4. 템플릿 선택 페이지에서 Amazon S3에 템플릿 업로드를 선택하고 파일을 선택한 후 다음을 선택합니다.
5. 다음과 같이 세부 정보 지정 페이지에서 스택과 키 이름을 지정합니다.

   a. 스택 이름에는 **`VPCE-Tutorial-Stack`**을 입력합니다.

   b. KeyName에서 VPCE-Tutorial-KeyPair를 선택합니다.

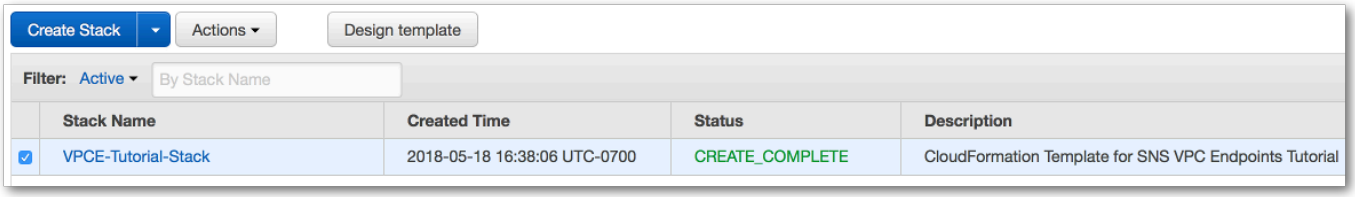
   c. SSHLocation에서 기본값 **`0.0.0.0/0`**을 그대로 둡니다.



   d. 다음을 선택합니다.

6. 옵션 페이지에서 모든 기본값을 그대로 두고 다음을 선택합니다.
7. 검토 페이지에서 스택 세부 정보를 확인합니다.
8. 기능에서 AWS CloudFormation이 사용자 지정 이름으로 IAM 리소스를 생성할 수 있음을 인정합니다.
9. Create를 선택합니다.

   AWS CloudFormation 콘솔에서 Stacks(스택) 페이지를 엽니다. VPCE-Tutorial-Stack은 CREATE_IN_PROGRESS 상태입니다. 몇 분 후 생성 프로세스가 완료되면 상태가 CREATE_COMPLETE으로 바뀝니다.



> **Tip**
>
> Refresh(새로 고침) 버튼을 선택하여 최신 스택 상태를 확인합니다.

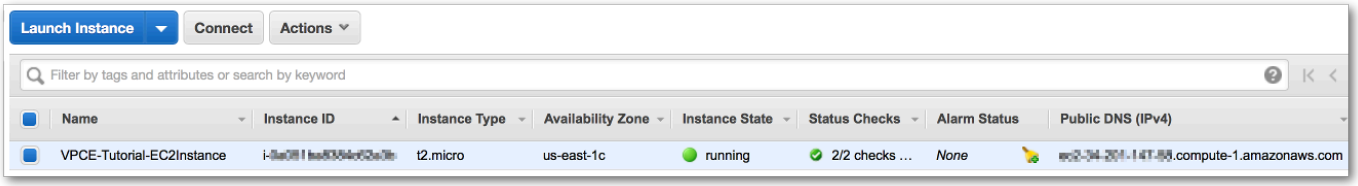
3단계: Amazon EC2 인스턴스에 인터넷 액세스 권한이 없는지 확인

이전 단계에서 VPC로 시작된 Amazon EC2 인스턴스에 인터넷 액세스 권한이 없습니다. 발신 트래픽을 허용하지 않고 Amazon SNS에 메시지를 게시할 수 없습니다. 인스턴스에 로그인하여 이를 확인합니다. 그런 다음 퍼블릭 엔드포인트에 연결을 시도하고 Amazon SNS에 메시지를 게시해 보세요.

이 시점에서는 게시 시도가 실패합니다. 나중 단계에서 Amazon SNS용 VPC 엔드포인트를 생성하고 나면 게시 시도가 성공합니다.

Amazon EC2 인스턴스에 연결하려면

1. https://console.aws.amazon.com/ec2/에서 Amazon EC2 콘솔을 엽니다.
2. 왼쪽의 탐색 메뉴에서 인스턴스 섹션을 찾습니다. 그런 다음 인스턴스를 선택합니다.
3. 인스턴스 목록에서 VPCE-Tutorial-EC2Instance를 선택합니다.
4. 퍼블릭 DNS(IPv4) 열에 있는 호스트 이름을 복사합니다.



5. 터미널을 엽니다. 키 페어가 포함된 디렉터리에서 다음 명령을 사용하여 인스턴스에 연결합니다. 여기에서 *instance-hostname*은 Amazon EC2 콘솔에서 복사한 호스트 이름입니다.

```
$ ssh -i VPCE-Tutorial-KeyPair.pem ec2-user@instance-hostname
```

인스턴스가 인터넷에 연결되어 있지 않음을 확인하려면

- 다음과 같이 터미널에서 amazon.com과 같은 퍼블릭 엔드포인트에 연결을 시도합니다.

```
$ ping amazon.com
```

연결 시도가 실패하므로 언제든지 취소할 수 있습니다(Windows에서는 Ctrl + C, macOS에서는 Command + C).

인스턴스가 Amazon SNS에 연결되어 있지 않음을 확인하려면

1. Amazon SNS 콘솔에 로그인합니다.
2. 왼쪽의 탐색 메뉴에서 주제를 선택합니다.
3. 주제 페이지에서 VPCE-Tutorial-Topic이라는 주제의 Amazon 리소스 이름(ARN)을 복사합니다.
4. 다음과 같이 터미널에서 주제에 메시지를 게시해 보세요.

```
$ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"
```

게시 시도가 실패하므로 언제든지 취소할 수 있습니다.

4단계: Amazon SNS용 Amazon VPC 엔드포인트 생성

VPC를 Amazon SNS에 연결하려면 인터페이스 VPC 엔드포인트를 정의하세요. 엔드포인트를 추가한 후에는 VPC의 Amazon EC2 인스턴스에 로그인할 수 있으며 이 인스턴스에서 Amazon SNS API를 사용할 수 있습니다. 주제에 메시지를 게시할 수 있습니다. 메시지는 비공개로 게시됩니다. 메시지는 AWS 네트워크 내에 머물러 있고 퍼블릭 인터넷으로 이동하지 않습니다.

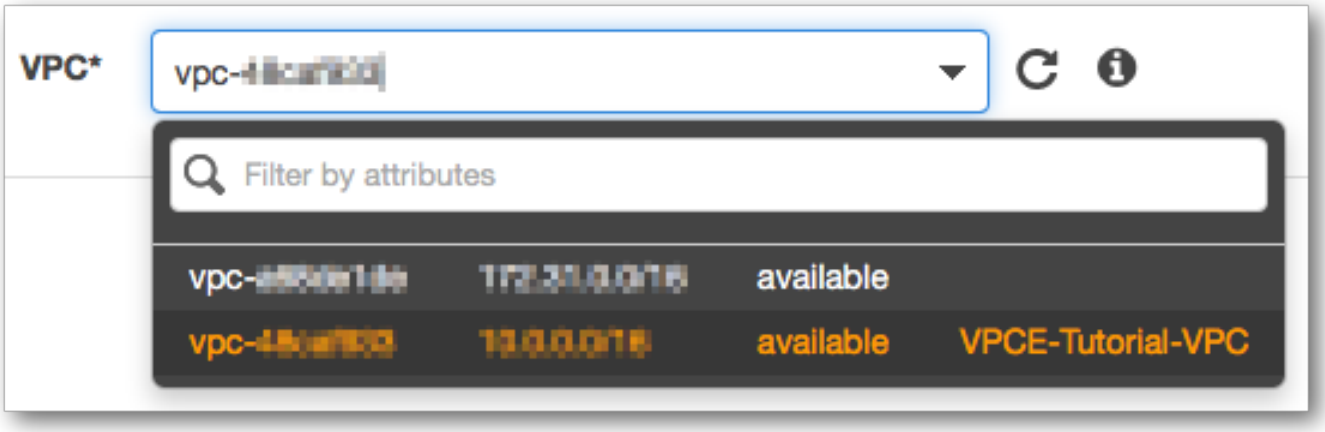
> **Note**
>
> 인스턴스에는 여전히 다른 AWS 서비스 및 인터넷 상의 엔드포인트에 액세스할 수 있는 권한이 없습니다.

엔드포인트를 생성하려면

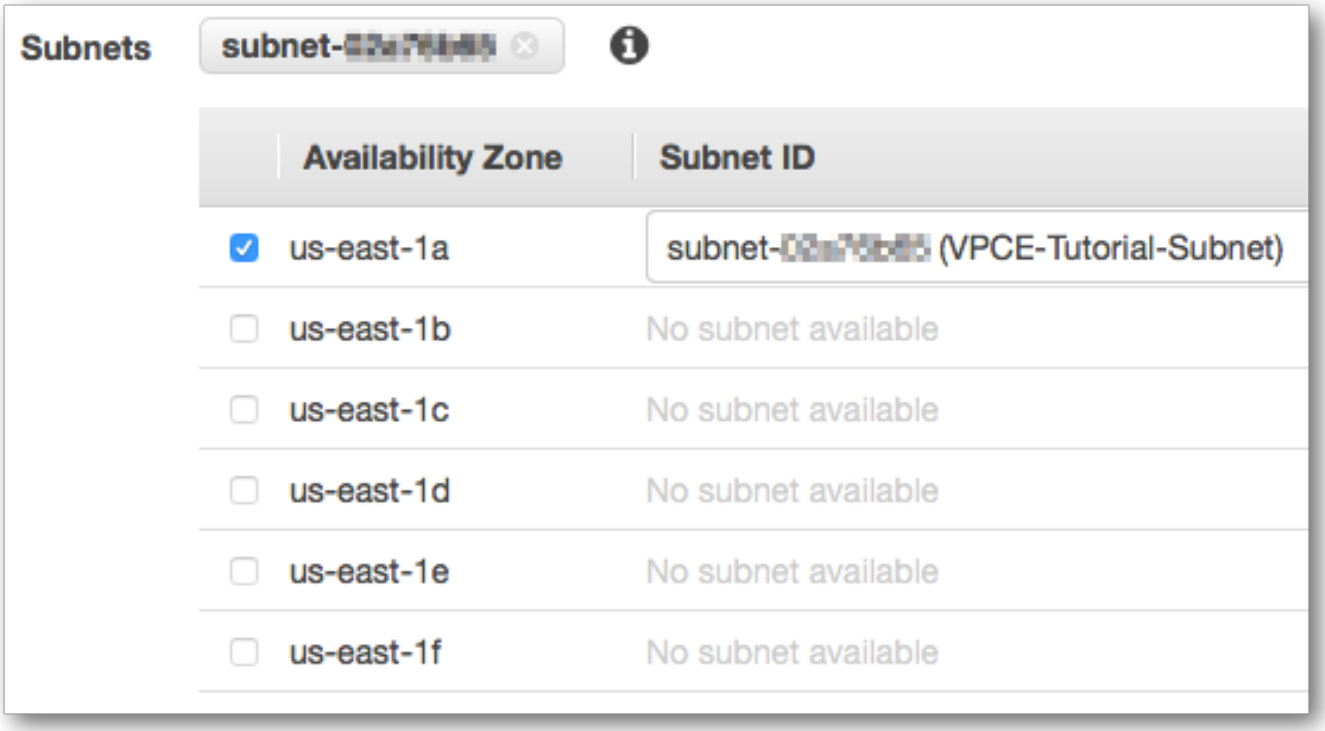
1. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
2. 왼쪽의 탐색 메뉴에서 엔드포인트를 선택합니다.

3. 엔드포인트 생성을 선택합니다.
4. 엔드포인트 생성 페이지의 서비스 범주에서 기본 선택인 AWS 서비스를 그대로 둡니다.
5. 서비스 이름에서 Amazon SNS의 서비스 이름을 선택합니다.

   서비스 이름은 선택한 리전에 따라 달라집니다. 예를 들어 미국 동부(버지니아 북부)를 선택한 경우 서비스 이름은 com.amazonaws.***us-east-1***.sns입니다.
6. VPC에서 이름이 VPCE-Tutorial-VPC인 VPC를 선택합니다.

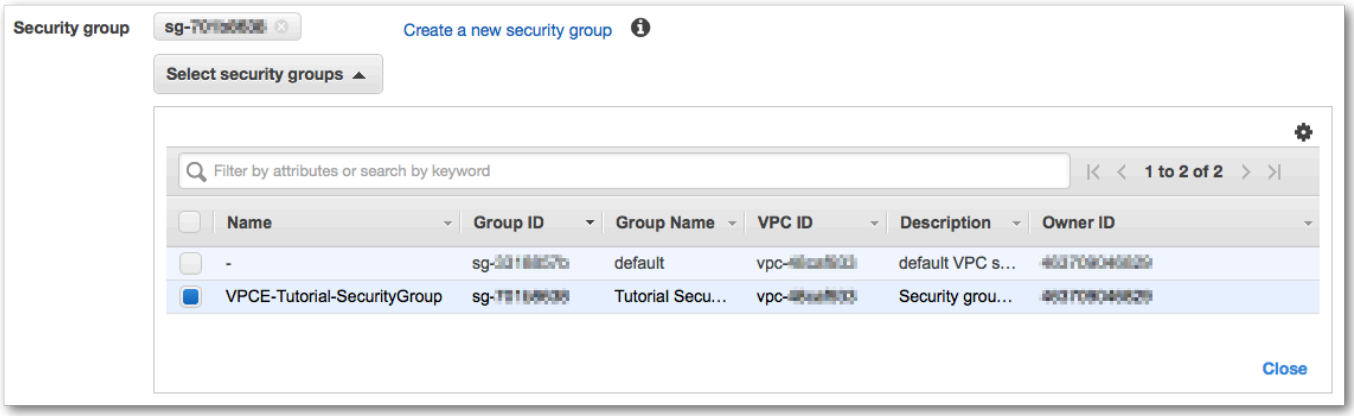


7. 서브넷에서 서브넷 ID에 VPCE-Tutorial-Subnet이 있는 서브넷을 선택합니다.

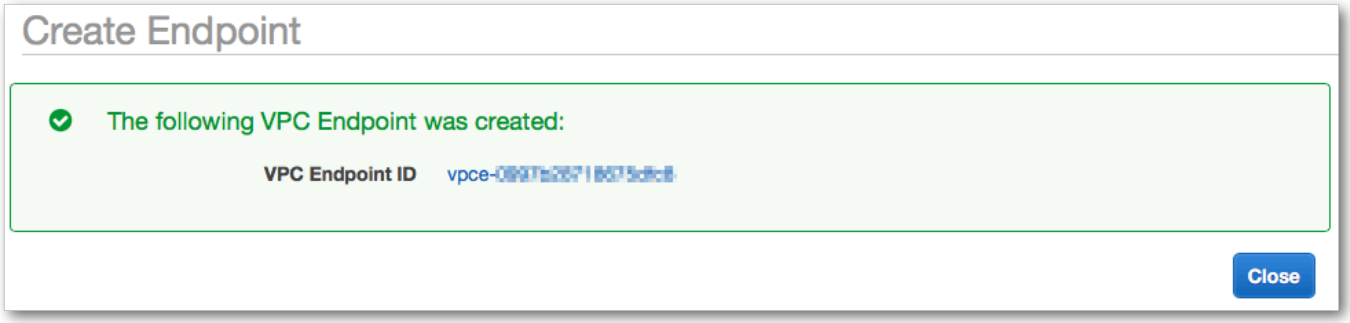


8. 프라이빗 DNS 이름 활성화에서 이 엔드포인트에 대해 활성화를 선택합니다.

9. 보안 그룹에서 Select security group(보안 그룹 선택)을 선택하고 VPCE-Tutorial-SecurityGroup을 선택합니다.

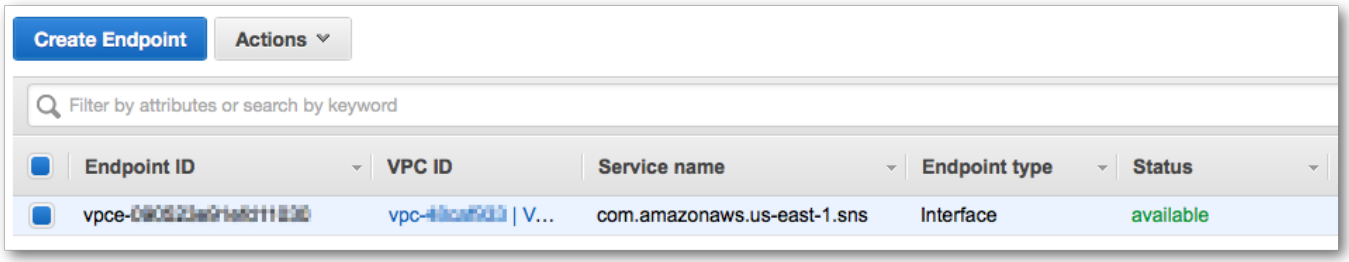


10. 엔드포인트 생성을 선택합니다. Amazon VPC 콘솔은 VPC 엔드포인트가 생성되었음을 확인합니다.



11. 닫기를 선택합니다.

    Amazon VPC 콘솔에서 엔드포인트 페이지가 열립니다. 새 엔드포인트의 상태는 대기 중입니다. 몇 분 후 생성 프로세스가 완료되면 상태가 사용 가능으로 바뀝니다.



5단계: Amazon SNS 주제에 메시지 게시

이제 VPC에 Amazon SNS용 엔드포인트가 포함되므로 Amazon EC2 인스턴스에 로그인하여 주제에 메시지를 게시할 수 있습니다.

메시지를 게시하려면

1. 터미널이 Amazon EC2 인스턴스에 연결되어 있지 않으면 다음과 같이 다시 연결합니다.

```
$ ssh -i VPCE-Tutorial-KeyPair.pem ec2-user@instance-hostname
```

2. Amazon SNS 주제에 메시지를 게시하기 위해 앞서 사용한 명령을 실행합니다. 이번에는 다음과 같이 게시 시도가 성공하고 Amazon SNS는 메시지 ID를 반환합니다.

```
$ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"

{
    "MessageId": "5b111270-d169-5be6-9042-410dfc9e86de"
}
```

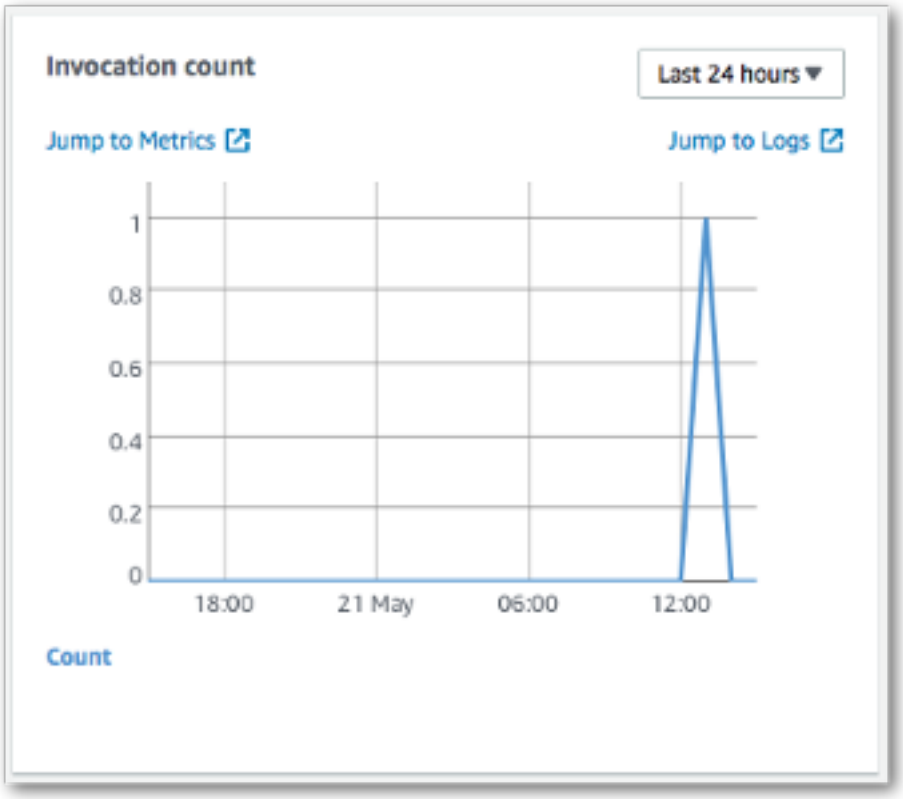
6단계: 메시지 전송 확인

Amazon SNS 주제는 메시지를 수신하면 두 가지 구독 Lambda 함수로 전송하여 메시지를 분산합니다. 이 함수는 메시지를 받으면 이 이벤트를 CloudWatch 로그에 로그합니다. 메시지 전송이 성공했는지 확인하려면 함수가 호출되었는지 확인한 후 CloudWatch 로그가 업데이트되었는지 확인합니다.

Lambda 함수가 호출되었는지 확인하려면

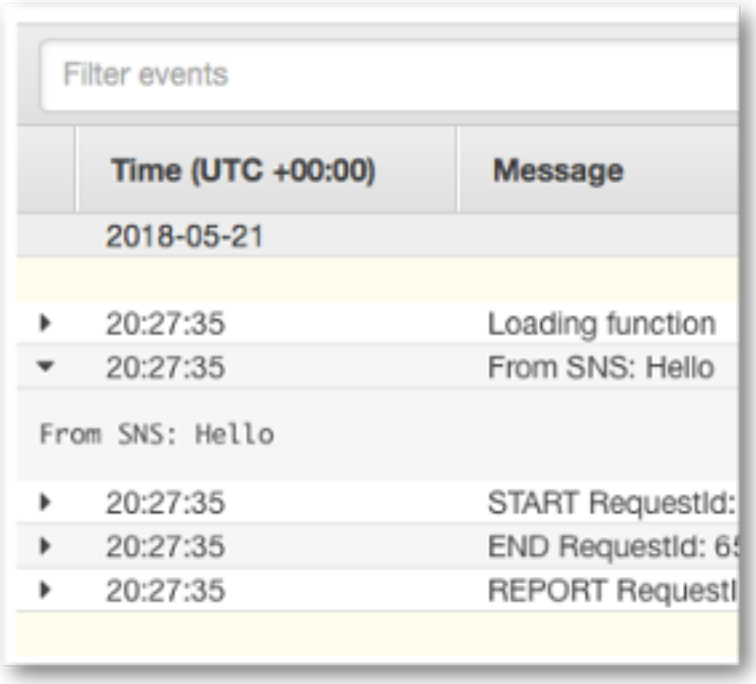
1. https://console.aws.amazon.com/lambda/에서 AWS Lambda 콘솔을 엽니다.
2. 함수 페이지에서 VPCE-Tutorial-Lambda-1을 선택합니다.
3. 모니터링을 선택합니다.
4. 호출 횟수 그래프를 확인합니다. 이 그래프에서는 Lambda 함수가 실행된 횟수를 보여줍니다.

   호출 횟수는 메시지를 주제에 게시한 횟수와 일치합니다.



CloudWatch 로그가 업데이트되었는지 확인하려면

1. https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.
2. 왼쪽 탐색 메뉴에서 로그를 선택합니다.
3. 다음 방법으로 Lambda 함수가 쓴 로그를 확인합니다.
   a. /aws/lambda/VPCE-Tutorial-Lambda-1/ 로그 그룹을 선택합니다.
   b. 로그 스트림을 선택합니다.
   c. 로그에 `From SNS: Hello` 항목이 포함되어 있는지 확인합니다.



d. 콘솔 상단의 로그 그룹을 선택하여 로그 그룹페이지를 표시합니다. 그런 다음 /aws/lambda/VPCE-Tutorial-Lambda-2/ 로그 그룹에 대해 앞서 수행한 단계를 반복합니다.

축하합니다! Amazon SNS용 엔드포인트를 VPC에 추가하는 방식으로 VPC에서 관리하는 네트워크에서 주제에 메시지를 게시할 수 있었습니다. 메시지는 비공개로 게시되어 퍼블릭 인터넷에 노출되지 않았습니다.

7단계: 정리

생성한 리소스를 보관하고 싶지 않으면 지금 삭제할 수 있습니다. 더 이상 사용하지 않는 AWS 리소스를 삭제하면 AWS 계정에 불필요한 요금이 발생하는 것을 방지할 수 있습니다.

먼저 Amazon VPC 콘솔을 사용하여 VPC 엔드포인트를 삭제합니다. 그 다음에 AWS CloudFormation 콘솔에서 스택을 삭제하여 이미 생성한 다른 리소스를 삭제합니다. 스택을 삭제할 때 AWS CloudFormation은 AWS 계정에서 스택의 리소스를 제거합니다.

VPC 엔드포인트를 삭제하려면

1. https://console.aws.amazon.com/vpc/에서 Amazon VPC 콘솔을 엽니다.
2. 왼쪽의 탐색 메뉴에서 엔드포인트를 선택합니다.
3. 생성한 엔드포인트를 선택합니다.
4. 작업을 선택한 다음 엔드포인트 삭제를 선택합니다.

5. 엔드포인트 삭제 창에서 예, 삭제를 선택합니다.

   그러면 엔드포인트 상태가 삭제 중으로 변경됩니다. 삭제가 완료되면 페이지에서 해당 엔드포인트가 제거됩니다.

AWS CloudFormation 스택을 삭제하려면

1. https://console.aws.amazon.com/cloudformation에서 AWS CloudFormation 콘솔을 엽니다.
2. VPCE-Tutorial-Stack 스택을 선택합니다.
3. [Actions]를 선택한 다음 [Delete Stack]을 선택합니다.
4. 스택 삭제 창에서 예, 삭제를 선택합니다.

   그러면 스택 상태가 DELETE_IN_PROGRESS로 변경됩니다. 삭제가 완료되면 페이지에서 해당 스택이 페이지에서 제거됩니다.

관련 리소스

자세한 정보는 다음 리소스를 참조하세요.

- AWS 보안 블로그: AWS PrivateLink를 사용하여 Amazon SNS에 게시된 메시지 보호
- Amazon VPC란 무엇인가?
- VPC 엔드포인트
- Amazon EC2란 무엇입니까?
- AWS CloudFormation 개념

메시지 데이터 보호 보안

- 메시지 데이터 보호는 저장 데이터가 아닌 이동 데이터에 대한 콘텐츠를 감사하고 제어하기 위해 자체 규칙과 정책을 정의하는 데 사용되는 Amazon SNS의 기능입니다.
- 메시지 데이터 보호는 메시지 중심의 엔터프라이즈 애플리케이션에 대한 거버넌스, 규정 준수 및 감사 서비스를 제공하므로 Amazon SNS 주제 소유자가 데이터 수신 및 송신을 제어하고 콘텐츠 흐름을 추적 및 기록할 수 있습니다.
- 페이로드 기반 거버넌스 규칙을 작성하여 승인되지 않은 페이로드 콘텐츠가 메시지 스트림에 들어가는 것을 막을 수 있습니다.

- 개별 구독자에게 다양한 콘텐츠 액세스 권한을 부여하고 전체 콘텐츠 흐름 프로세스를 감사할 수 있습니다.

Amazon SNS의 Identity and Access Management

Amazon SNS에 액세스하려면 AWS가 요청을 인증하는 데 사용할 수 있는 자격 증명이 필요합니다. 이러한 자격 증명에는 AWS 리소스(예: Amazon SNS 주제와 메시지)에 액세스할 수 있는 권한이 있어야 합니다. 다음 섹션에서는 AWS Identity and Access Management(IAM) 및 Amazon SNS를 사용하여 리소스에 대한 액세스를 제어함으로써 리소스를 보호하는 방법에 대해 자세히 설명합니다.

AWS Identity and Access Management(IAM)은 관리자가 AWS 리소스에 대한 액세스를 안전하게 제어할 수 있도록 지원하는 AWS 서비스입니다. IAM 관리자는 어떤 사용자가 Amazon SNS 리소스를 사용할 수 있도록 인증(로그인)되고 권한이 부여(권한 있음)될 수 있는지 제어합니다. IAM은 추가 비용 없이 사용할 수 있는 AWS 서비스입니다.

고객

AWS Identity and Access Management(IAM)를 사용하는 방법은 Amazon SNS에서 수행하는 작업에 따라 달라집니다.

서비스 사용자 – Amazon SNS 서비스를 사용하여 작업을 수행하는 경우 필요한 자격 증명과 권한을 관리자가 제공합니다. 더 많은 Amazon SNS 기능을 사용하여 작업를 수행한다면 추가 권한이 필요할 수 있습니다. 액세스 권한 관리 방식을 이해하면 적절한 권한을 관리자에게 요청할 수 있습니다. Amazon SNS의 기능에 액세스할 수 없다면 Amazon Simple Notification Service ID 및 액세스 문제 해결 단원을 참조하세요.

서비스 관리자 – 회사에서 Amazon SNS 리소스를 책임지고 있다면 Amazon SNS에 대한 완전한 액세스 권한이 있을 것입니다. 서비스 관리자는 서비스 사용자가 액세스해야 하는 Amazon SNS 기능과 리소스를 결정합니다. 그런 다음, IAM 관리자에게 요청을 제출하여 서비스 사용자의 권한을 변경해야 합니다. 이 페이지의 정보를 검토하여 IAM의 기본 개념을 이해하십시오. 회사가 Amazon SNS에서 IAM을 사용하는 방법에 대해 자세히 알아보려면 Amazon SNS가 IAM으로 작동하는 방식 단원을 참조하세요.

IAM 관리자 - IAM 관리자라면 Amazon SNS에 대한 액세스 관리 정책 작성 방법을 자세히 알고 싶을 수도 있습니다. IAM에서 사용할 수 있는 Amazon SNS 자격 증명 기반 정책 예제를 보려면 Amazon Simple Notification Service의 자격 증명 기반 정책 예를 참조하세요.

보안 인증을 통한 인증

인증은 ID 보안 인증을 사용하여 AWS에 로그인하는 방식입니다. AWS 계정 루트 사용자나 IAM 사용자로 또는 IAM 역할을 수임하여 인증(AWS에 로그인)되어야 합니다.

자격 증명 소스를 통해 제공된 보안 인증 정보를 사용하여 페더레이션형 ID로 AWS에 로그인할 수 있습니다. AWS IAM Identity Center (IAM Identity Center) 사용자, 회사의 Single Sign-On 인증, Google 또는 Facebook 보안 인증이 페더레이션형 ID의 예입니다. 페더레이션형 ID로 로그인할 때 관리자가 이전에 IAM 역할을 사용하여 ID 페더레이션을 설정했습니다. 페더레이션을 사용하여 AWS에 액세스하면 간접적으로 역할을 수임합니다.

사용자 유형에 따라 AWS Management Console 또는 AWS 액세스 포털에 로그인할 수 있습니다. AWS에 로그인하는 방법에 대한 자세한 내용은 AWS 로그인사용 설명서의 AWS 계정에 로그인하는 방법을 참조하세요.

AWS에 프로그래밍 방식으로 액세스하는 경우, AWS에서는 보안 인증 정보를 사용하여 요청에 암호화 방식으로 서명할 수 있는 소프트웨어 개발 키트(SDK) 및 명령줄 인터페이스(CLI)를 제공합니다. AWS 도구를 사용하지 않는 경우 요청에 직접 서명해야 합니다. 권장 방법을 사용하여 요청에 직접 서명하는 방법에 대한 자세한 내용은 IAM 사용 설명서의 AWS API 요청에 서명을 참조하세요.

사용하는 인증 방법에 상관없이 추가 보안 정보를 제공해야 할 수도 있습니다. 예를 들어, AWS은(는) 다중 인증(MFA)을 사용하여 계정의 보안을 강화하는 것을 권장합니다. 자세한 내용은 AWS IAM Identity Center사용 설명서의 다중 인증 및 IAM 사용 설명서의 AWS에서 다중 인증(MFA) 사용을 참조하세요.

AWS 계정 루트 사용자

AWS 계정를 생성할 때는 해당 계정의 모든 AWS 서비스 및 리소스에 대해 완전한 액세스 권한이 있는 단일 로그인 ID로 시작합니다. 이 보안 인증은 AWS 계정 루트 사용자라고 하며, 계정을 생성할 때 사용한 이메일 주소와 암호로 로그인하여 액세스합니다. 일상적인 작업에는 루트 사용자를 가급적 사용하지 않는 것이 좋습니다. 루트 사용자 보안 인증 정보를 보호하고 루트 사용자만 수행할 수 있는 작업을 수행하는 데 이 정보를 사용합니다. 루트 사용자로 로그인해야 하는 태스크의 전체 목록은 IAM 사용자 안내서의 루트 사용자 보안 인증이 필요한 태스크 섹션을 참조하세요.

페더레이션 ID

가장 좋은 방법은 관리자 액세스가 필요한 사용자를 포함한 사용자가 보안 인증 공급자와의 페더레이션을 사용하여 임시 보안 인증 정보를 사용하여 AWS 서비스에 액세스하도록 요구합니다.

페더레이션 ID는 엔터프라이즈 사용자 디렉터리, 웹 보안 인증 공급자, AWS Directory Service, Identity Center 디렉터리의 사용자 또는 보안 인증 소스를 통해 제공된 보안 인증을 사용하여 AWS 서비스에 액세스하는 모든 사용자입니다. 페더레이션 인증은 AWS 계정에 액세스할 때 역할을 수임하고 역할은 임시 보안 인증을 제공합니다.

중앙 집중식 액세스 관리를 위해 AWS IAM Identity Center을(를) 사용하는 것이 좋습니다. IAM Identity Center에서 사용자 및 그룹을 생성하거나 모든 AWS 계정 및 애플리케이션에서 사용하기 위해 고유한 보안 인증 소스의 사용자 및 그룹 집합에 연결하고 동기화할 수 있습니다. IAM Identity Center에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서에서 IAM Identity Center란 무엇입니까? 섹션을 참조하세요.

IAM 사용자 및 그룹

IAM 사용자는 단일 개인 또는 애플리케이션에 대한 특정 권한을 가지고 있는 AWS 계정 내 보안 인증입니다. 가능하다면 암호 및 액세스 키와 같은 장기 보안 인증 정보가 있는 IAM 사용자를 생성하는 대신, 임시 보안 인증 정보를 사용하는 것이 좋습니다. 하지만 IAM 사용자의 장기 보안 인증 정보가 필요한 특정 사용 사례가 있는 경우, 액세스 키를 교체하는 것이 좋습니다. 자세한 내용은 IAM 사용 설명서의 장기 보안 인증이 필요한 사용 사례의 경우 정기적으로 액세스 키 교체 섹션을 참조하세요.

IAM 그룹은 IAM 사용자 컬렉션을 지정하는 보안 인증입니다. 귀하는 그룹으로 로그인할 수 없습니다. 그룹을 사용하여 여러 사용자의 권한을 한 번에 지정할 수 있습니다. 그룹을 사용하면 대규모 사용자 집합의 권한을 더 쉽게 관리할 수 있습니다. 예를 들어, IAMAdmins라는 그룹이 있고 이 그룹에 IAM 리소스를 관리할 권한을 부여할 수 있습니다.

사용자는 역할과 다릅니다. 사용자는 한 사람 또는 애플리케이션과 고유하게 연결되지만, 역할은 해당 역할이 필요한 사람이라면 누구나 수임할 수 있습니다. 사용자는 영구적인 장기 보안 인증을 가지고 있지만, 역할은 임시 보안 인증만 제공합니다. 자세한 정보는 IAM 사용 설명서의 IAM 사용자를 만들어야 하는 경우(역할이 아님) 섹션을 참조하세요.

IAM 역할

IAM 역할은 특정 권한을 가지고 있는 AWS 계정계정 내 ID입니다. IAM 사용자와 유사하지만, 특정 개인과 연결되지 않습니다. 역할을 전환하여 AWS Management Console에서 IAM 역할을 임시로 수임할 수 있습니다. AWS CLI 또는 AWS API 태스크를 직접적으로 호출하거나 사용자 지정 URL을 사용하여 역할을 수임할 수 있습니다. 역할 사용 방법에 대한 자세한 정보는 IAM 사용 설명서의 IAM 역할 사용를 참조하세요.

임시 보안 인증이 있는 IAM 역할은 다음과 같은 상황에서 유용합니다.

- 페더레이션 사용자 액세스 - 페더레이션형 ID에 권한을 부여하려면 역할을 생성하고 해당 역할의 권한을 정의합니다. 페더레이션형 ID가 인증되면 이 ID는 역할과 연결되며 역할에 의해 정의된 권한이 부여됩니다. 페더레이션 역할에 대한 자세한 내용은 IAM 사용 설명서의 서드 파티 자격 증명 공급자의 역할 만들기 부분을 참조하세요. IAM Identity Center를 사용하는 경우 권한 세트를 구성합니다. 인증 후 보안 인증 정보에서 액세스할 수 있는 항목을 제어하기 위해 IAM Identity Center는 권한 집합을 IAM의 역할과 연결합니다. 권한 세트에 대한 자세한 내용은 AWS IAM Identity Center 사용 설명서의 권한 세트 섹션을 참조하세요.
- 임시 IAM 사용자 권한 - IAM 사용자 또는 역할은 IAM 역할을 수임하여 특정 태스크에 대한 다양한 권한을 임시로 받을 수 있습니다.
- 크로스 계정 액세스 - IAM 역할을 사용하여 다른 계정의 사용자(신뢰할 수 있는 보안 주체)가 내 계정의 리소스에 액세스하도록 허용할 수 있습니다. 역할은 계정 간 액세스를 부여하는 기본적인 방법입니다. 그러나 일부 AWS 서비스를 사용하면 역할을(프록시로 사용하는 대신) 리소스에 정책을 직접 연결할 수 있습니다. 교차 계정 액세스를 위한 역할과 리소스 기반 정책의 차이점을 알아보려면 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이 섹션을 참조하세요.
- 교차 서비스 액세스 – 일부 AWS 서비스은(는) 다른 AWS 서비스의 기능을 사용합니다. 예를 들어 서비스에서 직접적으로 호출을 수행하면 일반적으로 해당 서비스는 Amazon EC2에서 애플리케이션을 실행하거나 Amazon S3에 객체를 저장합니다. 서비스는 호출하는 보안 주체의 권한을 사용하거나, 서비스 역할을 사용하거나, 또는 서비스 연결 역할을 사용하여 이 작업을 수행할 수 있습니다.
  - 전달 액세스 세션(FAS) - IAM 사용자 또는 역할을 사용하여 AWS에서 작업을 수행하는 사람은 보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 AWS 서비스를 직접 호출하는 보안 주체의 권한과 요청하는 AWS 서비스를 함께 사용하여 다운스트림 서비스에 대한 요청을 수행합니다. FAS 요청은 서비스에서 완료를 위해 다른 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 받은 경우에만 이루어집니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하십시오.
  - 서비스 역할 - 서비스 역할은 서비스가 사용자를 대신하여 태스크를 수행하기 위해 맡는 IAM 역할입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요.
  - 서비스 연결 역할 – 서비스 연결 역할은 AWS 서비스에 연결된 서비스 역할의 한 유형입니다. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 링크 역할은 AWS 계정에 나타나고, 서비스가 소유합니다. IAM 관리자는 서비스 연결 역할의 권한을 볼 수 있지만 편집할 수는 없습니다.
- Amazon EC2에서 실행 중인 애플리케이션 – IAM 역할을 사용하여 EC2 인스턴스에서 실행되고 AWS CLI 또는 AWS API 요청을 수행하는 애플리케이션의 임시 보안 인증을 관리할 수 있습니다. 이는 EC2 인스턴스 내에 액세스 키를 저장할 때 권장되는 방법입니다. EC2 인스턴스에 AWS역할을

할당하고 해당 역할을 모든 애플리케이션에서 사용할 수 있도록 하려면 인스턴스에 연결된 인스턴스 프로파일을 생성합니다. 인스턴스 프로파일에는 역할이 포함되어 있으며 EC2 인스턴스에서 실행되는 프로그램이 임시 보안 인증을 얻을 수 있습니다. 자세한 정보는 IAM 사용 설명서의 IAM 역할을 사용하여 Amazon EC2 인스턴스에서 실행되는 애플리케이션에 권한 부여 섹션을 참조하세요.

IAM 역할을 사용할지 또는 IAM 사용자를 사용할지를 알아보려면 IAM 사용 설명서의 IAM 역할(사용자 대신)을 생성하는 경우를 참조하세요.

정책을 사용한 액세스 관리

정책을 생성하고 AWSID 또는 리소스에 연결하여 AWS내 액세스를 제어합니다. 정책은 ID 또는 리소스와 연결될 때 해당 권한을 정의하는 AWS의 객체입니다. AWS은(는) 보안 주체(사용자, 루트 사용자 또는 역할 세션)가 요청을 보낼 때 이러한 정책을 평가합니다. 정책에서 권한은 요청이 허용되는지 또는 거부되는지를 결정합니다. 대부분의 정책은 AWS에 JSON 설명서로서 저장됩니다. JSON 정책 문서의 구조와 콘텐츠에 대한 자세한 정보는 IAM 사용 설명서의 JSON 정책 개요 섹션을 참조하세요.

관리자는 AWSJSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지를 지정할 수 있습니다.

기본적으로, 사용자와 역할에는 어떠한 권한도 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 수임할 수 있습니다.

IAM 정책은 작업을 수행하기 위해 사용하는 방법과 상관없이 작업에 대한 권한을 정의합니다. 예를 들어, `iam:GetRole`태스크를 허용하는 정책이 있다고 가정합니다. 해당 정책이 있는 사용자는 AWS Management Console, AWS CLI또는 AWSAPI에서 역할 정보를 가져올 수 있습니다.

ID 기반 정책

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 태스크를 수행할 수 있는지를 제어합니다. ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

ID 기반 정책은 인라인 정책 또는 관리형 정책으로 한층 더 분류할 수 있습니다. 인라인 정책은 단일 사용자, 그룹 또는 역할에 직접 포함됩니다. 관리형 정책은 AWS 계정에 속한 다수의 사용자, 그룹 및 역할에 독립적으로 추가할 수 있는 정책입니다. 관리형 정책에는 AWS관리형 정책과 고객 관리형 정책이 포함되어 있습니다. 관리형 정책 또는 인라인 정책을 선택하는 방법을 알아보려면 IAM 사용 설명서의 관리형 정책과 인라인 정책의 선택을 참조하세요.

리소스 기반 정책

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 제어할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 태스크를 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는 AWS 서비스이(가) 포함될 수 있습니다.

리소스 기반 정책은 해당 서비스에 있는 인라인 정책입니다. 리소스 기반 정책에서는 IAM의 AWS관리형 정책을 사용할 수 없습니다.

액세스 제어 목록(ACL)

액세스 제어 목록(ACL)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACLs는 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

Amazon S3, AWS WAF및 Amazon VPC는 ACL을 지원하는 대표적인 서비스입니다. ACL에 대해 자세히 알아보려면 Amazon Simple Storage Service 개발자 안내서의 액세스 제어 목록(ACL) 개요를 참조하세요.

기타 정책 유형

AWS은(는) 비교적 일반적이지 않은 추가 정책 유형을 지원합니다. 이러한 정책 유형은 더 일반적인 정책 유형에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 – 권한 경계는 ID 기반 정책에 따라 IAM 엔터티(IAM 사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 개체에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 개체의 ID 기반 정책과 그 권한 경계의 교집합입니다. `Principal` 필드에서 사용자나 역할을 보안 주체로 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 정보는 IAM 사용 설명서의 IAM 엔터티에 대한 권한 경계 섹션을 참조하세요.
- 서비스 제어 정책(SCP) – SCP는 AWS Organizations에서 조직 또는 조직 단위(OU)에 최대 권한을 지정하는 JSON 정책입니다. AWS Organizations은(는) 기업이 소유하는 여러 개의 AWS 계정을(를) 그룹화하고 중앙에서 관리하기 위한 서비스입니다. 조직에서 모든 기능을 활성화할 경우 서비스 제어 정책(SCP)을 임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 각 AWS 계정 루트 사용자를 비롯하여 멤버 계정의 엔터티에 대한 권한을 제한합니다. 조직 및 SCP에 대한 자세한 정보는 AWS Organizations 사용 설명서의 SCP 작동 방식을 참조하세요.

- 세션 정책 – 세션 정책은 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의 ID 기반 정책의 교차와 세션 정책입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 정보는 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 여러 정책 유형이 관련될 때 AWS가 요청을 허용할지 여부를 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하십시오.

액세스 제어

Amazon SNS에는 AWS Identity and Access Management(IAM) 정책에 사용되는 것과 동일한 언어로 작성된 정책을 사용하는 자체 리소스 기반 권한 시스템이 있습니다. 즉, Amazon SNS 정책 및 IAM 정책과 유사한 정책을 적용할 수 있습니다.

> **Note**
>
> 모든 AWS 계정이 자신의 계정에 속한 사용자에게 그 권한을 위임할 수도 있다는 사실을 이해하고 있어야 합니다. 교차 계정 액세스를 사용하면 추가 사용자를 관리하지 않고도 AWS 리소스 액세스 권한을 공유할 수 있습니다. 교차 계정 액세스 사용에 대한 자세한 정보는 IAM 사용 설명서의 교차 계정 액세스 사용을 참조하세요.

Amazon SNS의 액세스 관리 개요

이 섹션에서는 정책 작성에 액세스 정책 언어를 사용하기 위해 알아두어야 할 기본 개념을 설명합니다. 또한 액세스 정책 언어를 사용한 액세스 제어 방식과 정책 평가 방식의 일반적인 프로세스도 알아봅니다.

주제

- 액세스 제어를 사용할 때
- 주요 개념
- 아키텍처 개요

- 액세스 정책 언어 사용
- 평가 로직
- Amazon SNS 액세스 제어의 예제 사례

액세스 제어를 사용할 때

리소스에 대한 액세스 권한을 부여하거나 거부하는 방식에는 상당한 유연성을 발휘할 수 있습니다. 그러나 일반적인 사용 사례는 매우 간단합니다.

- 다른 AWS 계정에 특정 유형의 주제 작업(예: 게시)에 대한 권한을 부여하려는 경우 자세한 설명은 주제에 대한 AWS 계정 액세스 권한 부여 섹션을 참조하세요.
- 주제에 대한 구독을 HTTPS 프로토콜로만 제한하려는 경우 자세한 설명은 HTTPS로 구독 제한 섹션을 참조하세요.
- Amazon SNS에서 Amazon SQS 대기열에 메시지를 게시하는 것을 허용하려는 경우 자세한 설명은 Amazon SQS 대기열에 메시지를 게시합니다. 섹션을 참조하세요.

주요 개념

다음 섹션에서는 액세스 정책 언어를 사용하기 위해 알아두어야 할 개념을 설명합니다. 이 개념들은 논리적 순서대로 제시됩니다. 즉 가장 먼저 알아야 할 용어가 목록의 맨 위에 있습니다.

주제

- 권한
- 문
- 정책
- Issuer
- 보안 주체
- 작업
- Resource
- 조건과 키
- 요청자
- 평가
- Effect

- 기본 거부
- 허용
- 명시적 거부

권한

권한(permission)은 특정 리소스에 대해 어떤 종류의 액세스를 허용하거나 허용하지 않는 개념입니다. 기본적으로 권한은 "D에 해당된다면 A는 C에게 B를 할 수 있다/없다"의 형식을 띱니다. 예를 들어, Jane(A)은 HTTP 프로토콜을 사용(D)한다면 TopicA(C)에 게시(B)할 권한을 갖습니다. Jane이 TopicA에 게시할 때마다 서비스에서는 그녀에게 권한이 있는지 그리고 그 요청이 권한에 설정된 조건을 충족하는지 확인합니다.

문

문(statement)은 단일 권한에 대한 공식적인 설명이며 액세스 정책 언어로 작성됩니다. 설명은 항상 정책(다음 개념 설명 참조)이라고 부르는 더 광범위한 컨테이너 문서의 일부로 작성합니다.

정책

정책(policy)은 하나 이상의 문에 대한 컨테이너의 역할을 수행하는 문서(액세스 정책 언어에서 작성됨)입니다. 예를 들어, 하나의 정책이 2개의 설명을 포함할 수 있습니다. 그중 하나는 Jane이 이메일 프로토콜을 사용하여 구독할 수 있다고, 다른 하나는 Bob이 Topic A에 게시할 수 없다고 규정합니다. 다음 그림에서 보여 주는 것처럼, 상응하는 시나리오는 2개의 정책을 갖습니다. 하나는 Jane이 이메일 프로토콜을 사용하여 구독할 수 있다고, 다른 하나는 Bob이 Topic A에 게시할 수 없다고 규정합니다.

정책 문서에는 ASCII 문자만 사용할 수 있습니다. `aws:SourceAccount` 및 `aws:SourceOwner`를 사용하여 ASCII가 아닌 문자가 포함된 다른 AWS 서비스의 ARN을 플러그인해야 하는 시나리오를 해결할 수 있습니다. `aws:SourceAccount`과 `aws:SourceOwner` 비교 간 차이점을 참조하세요.

Issuer

발행자(issuer)는 리소스에 대한 권한을 부여하기 위해 정책을 작성하는 사람입니다. 발행자는 그 정의에 따라 항상 리소스 소유자입니다. AWS는 AWS 서비스 사용자가 본인 소유가 아닌 리소스에 대해 정책을 만드는 것을 허용하지 않습니다. John이 리소스 소유자라면 AWS는 John이 그 리소스에 대한 권한을 부여하고자 작성한 정책을 제출할 때 John의 신원을 인증합니다.

보안 주체

보안 주체는 정책에 따라 권한을 받는 사람입니다. 보안 주체는 "A는 D가 적용되는 경우 B에게 B를 할 수 있는 권한이 있습니다."라는 명령문에서 A입니다. 책에서 보안 주체를 "모든 사람"으로 설정할 수 있습니다. 즉, 모든 사람을 나타내는 와일드카드를 지정할 수 있습니다. 예를 들어, 요청자의 신원이 아닌, 요청자의 IP 주소와 같은 다른 식별 특성에 따라 액세스를 제한하려는 경우 이와 같이 할 수 있습니다.

작업

작업은 보안 주체가 수행할 수 있는 권한이 있는 작업입니다. "D에 해당된다면 A는 C에게 B를 할 수 있다" 설명에서 작업은 B입니다. 일반적으로 작업은 AWS에 대한 요청에 포함된 작업일 뿐입니다. 예를 들어, Jane이 `Action=Subscribe`를 사용하여 Amazon SNS에 요청을 전송합니다. 하나의 정책에 하나 이상의 작업을 지정할 수 있습니다.

Resource

리소스는 보안 주체가 액세스를 요청하는 개체입니다. "D에 해당된다면 A는 C에게 B를 할 수 있다" 설명에서 리소스는 C입니다.

조건과 키

조건(conditions)은 권한에 대한 제한 또는 세부 정보입니다. "D에 해당된다면 A는 C에게 B를 할 수 있다" 설명에서 조건은 D입니다. 정책에서 조건을 지정하는 부분이 가장 세부적이고 복잡할 수 있습니다. 일반적인 조건은 다음 항목과 관련 있습니다.

- 날짜와 시간(예: 요청이 특정 일보다 먼저 도착해야 함)
- IP 주소(예: 요청자의 IP 주소가 특정 CIDR 범위에 속해야 함)

키는 액세스 제한의 기준이 되는 구체적 특성입니다. 이를테면 요청의 날짜와 시간이 키가 될 수 있습니다.

조건과 키를 함께 사용하여 제한을 표현합니다. 실제로 제한을 구현하는 방식은 예시를 통해 가장 쉽게 이해할 수 있습니다. 2010년 5월 30일 이전으로 액세스를 제한하려는 경우 `DateLessThan`이라는 조

건을 사용합니다. 호출된 키 `aws:CurrentTime`을 사용하여 값 `2010-05-30T00:00:00Z`로 설정합니다. AWS에서는 사용할 수 있는 조건과 키를 정의합니다. AWS 서비스 자체(예: Amazon SQS 또는 Amazon SNS)에서 서비스별 키를 정의할 수도 있습니다. 자세한 설명은 Amazon SNS API 권한: 작업 및 리소스 참조 섹션을 참조하세요.

요청자

요청자(requester)(requester)는 AWS 서비스에 요청을 보내 특정 리소스에 대한 액세스 권한을 원하는 사람입니다. 요청자는 AWS에 "D에 해당된다면 내가 C에게 B를 하는 것을 허락해달라"는 의미의 요청을 보냅니다.

평가

평가(evaluation)는 AWS 서비스에서 수신한 요청을 적용 가능한 정책에 따라 거부할지 또는 허용할지 결정하는 데 이용하는 프로세스입니다. 평가 로직에 대한 자세한 내용은 평가 로직에서 확인하세요.

Effect

효과(effect)는 정책 설명에서 평가 시점에 반환해야 할 결과입니다. 정책 설명을 작성할 때 이 값을 지정하며, 가능한 값은 거부(deny)와 허용(allow)입니다.

예를 들어, 남극 대륙에서 보내는 모든 요청을 거부하는 정책 설명을 작성할 수 있습니다(효과=요청에서 남극 대륙에 할당된 IP 주소를 사용할 경우 거부). 또는 남극 대륙에서 보내지 않은 모든 요청을 허용하라는 정책 설명을 작성할 수 있습니다(효과=남극 대륙에서 보내지 않은 요청이면 허용). 두 설명이 동일한 기능을 할 것처럼 보이지만 액세스 정책 언어 로직에서는 서로 다릅니다. 자세한 설명은 평가 로직 섹션을 참조하세요.

효과에 대해 지정할 수 있는 값이 2개뿐이지만(허용 또는 거부) 정책 평가 시점에는 서로 다른 3가지 결과가 나올 수 있습니다. 기본 거부, 허용 또는 명시적 거부입니다. 자세한 정보는 다음 개념과 평가 로직에서 확인하세요.

기본 거부

기본 거부(default deny)는 허용 또는 명시적 거부가 없는 정책의 기본 결과입니다.

허용

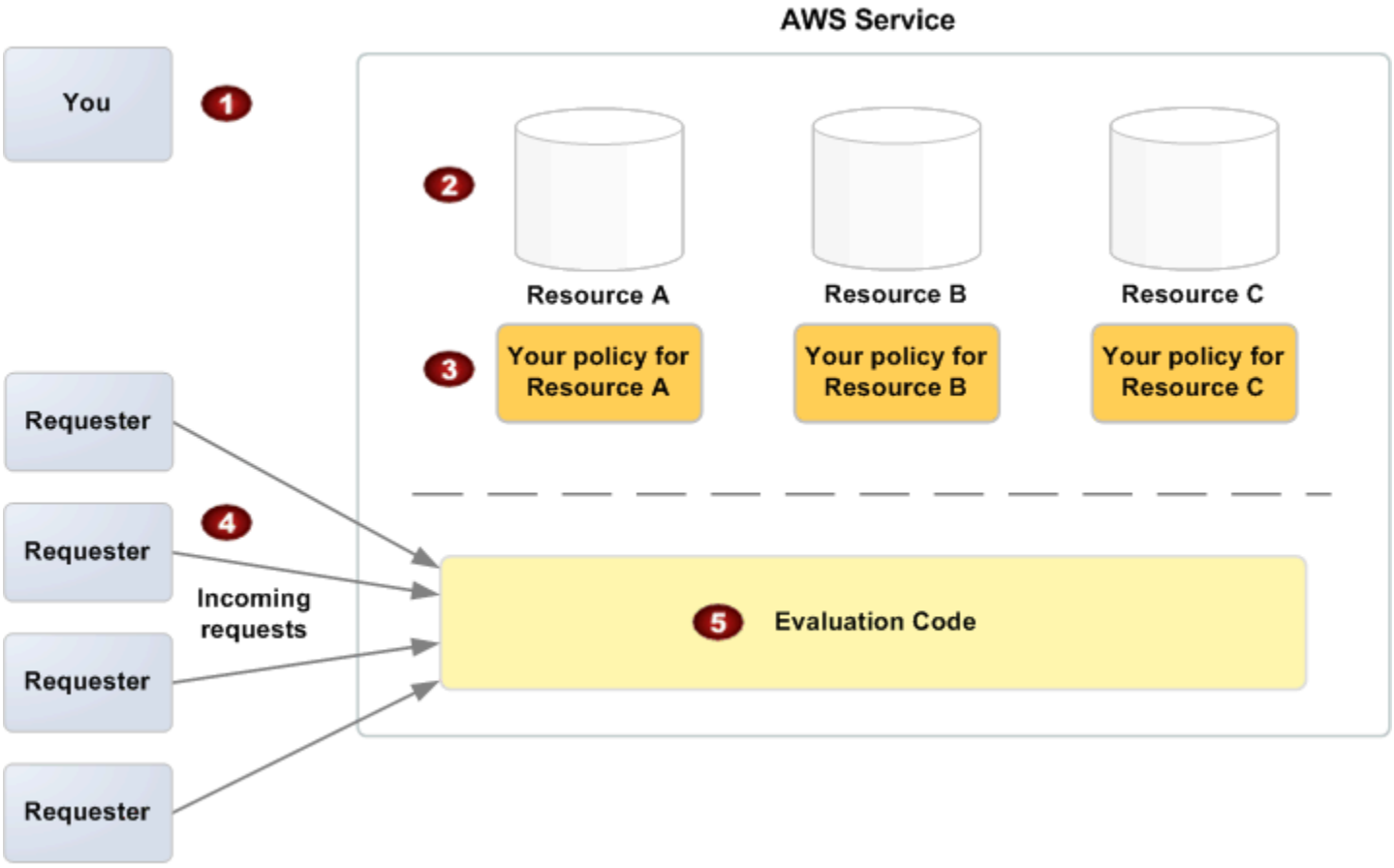
허용(allow)은 효과=기술된 모든 조건을 충족할 경우 허용인 설명의 결과입니다. 예: 2010년 4월 30일 오후 1:00 이전에 수신된 요청이면 허용. 허용은 모든 기본 거부를 무시하지만 명시적 거부는 무시하지 않습니다.

명시적 거부

명시적 거부(explicit deny)는 효과=기술된 모든 조건을 충족할 경우 거부인 설명의 결과입니다. 예: 남극 대륙에서 보낸 모든 요청 거부. 남극 대륙에서 보낸 모든 요청은 다른 어떤 정책에서 허용하더라도 항상 거부됩니다.

아키텍처 개요

다음 그림과 표는 상호 작용을 통해 리소스에 대한 액세스 제어를 수행하는 주요 구성 요소에 대해 설명합니다.

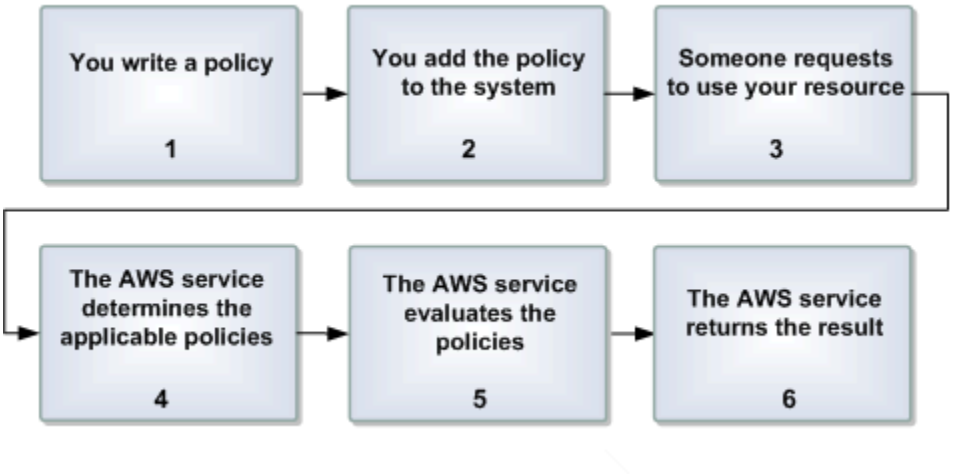


1	리소스 소유자
2	AWS 서비스(예: Amazon SQS 대기열) 내에 포함된 리소스입니다.
3	정책.

	일반적으로 리소스당 하나의 정책이 있지만, 여러 개의 정책을 가질 수도 있습니다. AWS 서비스 자체에서 제공하는 API를 사용하여 정책을 업로드하고 관리할 수 있습니다.
4	요청자 및 요청자가 AWS 서비스에서 수신하는 요청입니다.
5	액세스 정책 언어 평가 코드입니다. 이는 AWS 서비스에 포함된 코드 모음으로서 적용 가능한 정책에 따라 수신 요청을 평가하고 요청자에게 리소스에 대한 액세스를 허용할지 여부를 결정합니다. 서비스에서 결정을 내리는 방식에 대한 자세한 내용은 평가 로직에서 확인하세요.

액세스 정책 언어 사용

다음 그림과 표는 액세스 정책 언어를 사용하여 액세스를 제어하는 일반적인 프로세스를 설명합니다.



액세스 정책 언어를 통한 액세스 제어를 사용하는 프로세스

1	리소스에 대한 정책을 작성합니다. 예를 들어, Amazon SNS 주제에 대한 권한을 지정하는 정책을 작성합니다.
2	AWS에 정책을 업로드합니다. AWS 서비스 자체에서 제공하는 API를 사용하여 정책을 업로드할 수 있습니다. 예를 들어 Amazon SNS `SetTopicAttributes` 작업을 사용하여 특정 Amazon SNS 주제에 대한 정책을 업로드합니다.
3	어떤 사람이 리소스를 사용하기 위해 요청을 전송합니다.

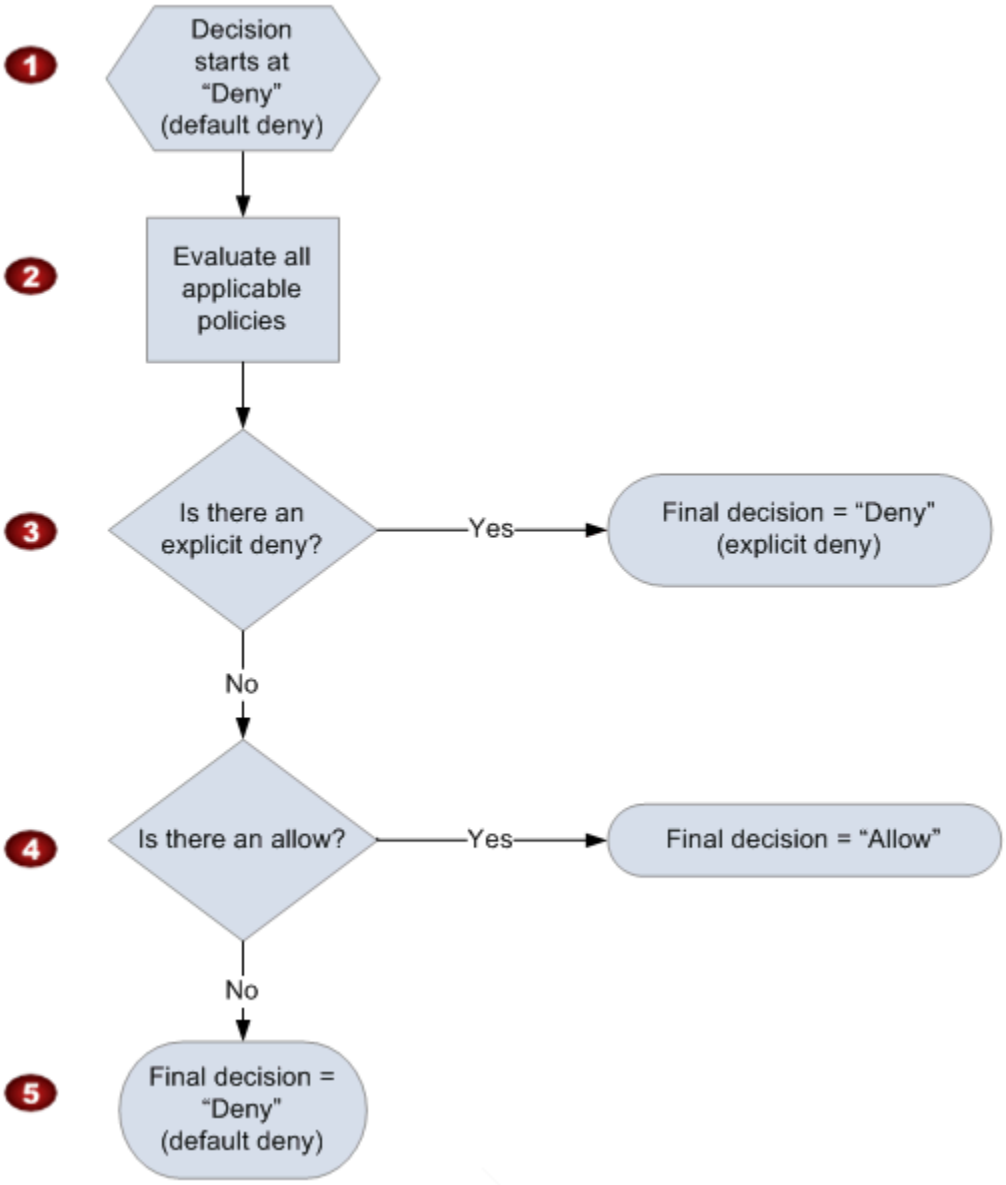
	예를 들어, 사용자가 주제 중 하나를 사용하기 위해 Amazon SNS에 요청을 전송합니다.
4	AWS 서비스는 이 요청에 어떤 정책을 적용할 수 있는지를 결정합니다. 예를 들어, Amazon SNS는 사용 가능한 모든 Amazon SNS 정책을 살펴보고 어떤 정책을 적용할 수 있는지를 결정합니다(리소스가 무엇이고 요청자가 누구인지 등을 기준으로).
5	AWS 서비스가 정책을 평가합니다. 예를 들어, Amazon SNS는 정책을 평가하고 요청자가 주제를 사용하도록 허용할지 여부를 결정합니다. 결정 로직에 대한 자세한 내용은 평가 로직에서 확인하세요.
6	AWS 서비스는 요청을 거부하거나 계속 처리합니다. 이를테면 정책 평가 결과에 따라 서비스가 요청자에게 "액세스 거부" 오류로 반환되거나 계속 요청을 처리합니다.

평가 로직

평가 시점의 목표는 권한 요청(grant request)을 허용 또는 거부할지를 결정하는 것입니다. 평가 로직에서는 몇 가지 기본 규칙을 따릅니다.

- 기본적으로, 본인을 제외한 사람이 리소스를 사용하겠다고 요청하면 모두 거부됩니다.
- 허용은 모든 기본 거부를 무시합니다.
- 명시적 거부는 모든 허용을 무시합니다.
- 정책이 평가되는 순서는 중요하지 않습니다.

그러한 결정이 내려지는 방법은 다음 순서도와 설명에서 자세히 다룹니다.



1	결정은 기본 거부에서 시작합니다.
2	그러면 적용 코드에서 리소스, 보안 주체, 작업, 조건을 고려하여 해당 요청에 적용 가능한 모든 정책을 평가합니다. 적용 코드에서 정책을 평가하는 순서는 중요하지 않습니다.
3	적용 코드는 이 모든 정책에서 해당 요청에 적용될 명시적 거부 명령을 찾습니다.

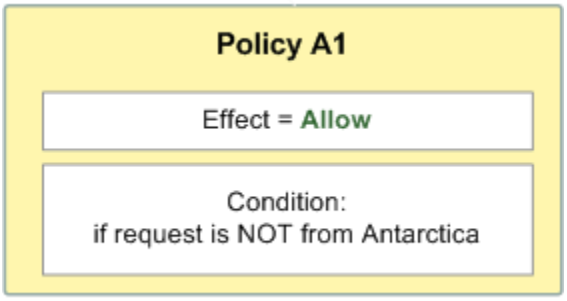
	하나라도 찾으면 적용 코드는 "거부" 결정을 반환하고 프로세스가 종료됩니다. 이것이 명시적 거부입니다. 자세한 정보는 명시적 거부에서 확인하세요.
4	명시적 거부가 없을 경우 적용 코드는 해당 요청에 적용될 "허용" 명령을 찾습니다. 하나라도 찾으면 적용 코드는 "허용" 결정을 반환하고 프로세스가 완료됩니다. 서비스에서는 계속 요청을 처리합니다.
5	허용이 없을 경우 최종 결정은 "거부"입니다. 명시적 거부 또는 허용이 없으므로 기본 거부로 간주됩니다. 자세한 정보는 기본 거부에서 확인하세요.

명시적 거부와 기본 거부의 상호 작용

어떤 정책이 해당 요청에 직접적으로 적용되지 않을 경우 그 결과는 기본 거부입니다. 예를 들어, 사용자가 Amazon SNS 사용을 요청하지만 그 주제에 대한 정책에서 사용자의 AWS 계정이 전혀 언급되지 않은 경우 이 정책의 결과는 기본 거부입니다.

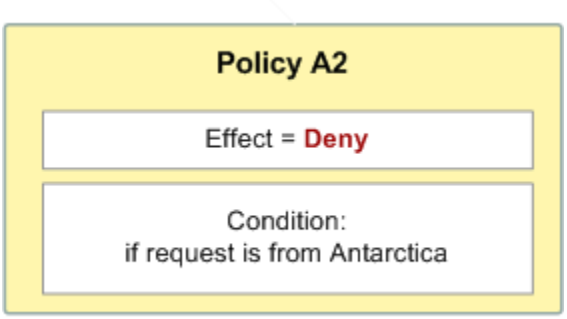
설명의 조건이 충족되지 않을 경우에도 정책의 결과는 기본 거부입니다. 설명의 모든 조건이 충족될 경우, 정책의 결과는 정책에 포함된 효과 요소의 값에 따라 허용 또는 명시적 거부가 됩니다. 정책에서는 어떤 조건이 충족되지 않을 때 해야 할 일을 지정하지 않으므로, 그러한 경우 결과는 기본 거부가 됩니다.

이를테면 남극 대륙에서 보내는 요청을 거부하고 싶습니다. 남극 대륙에서 오지 않은 요청만 허용하도록 정책을 작성합니다(정책 A1). 다음 다이어그램은 이 정책을 보여줍니다.



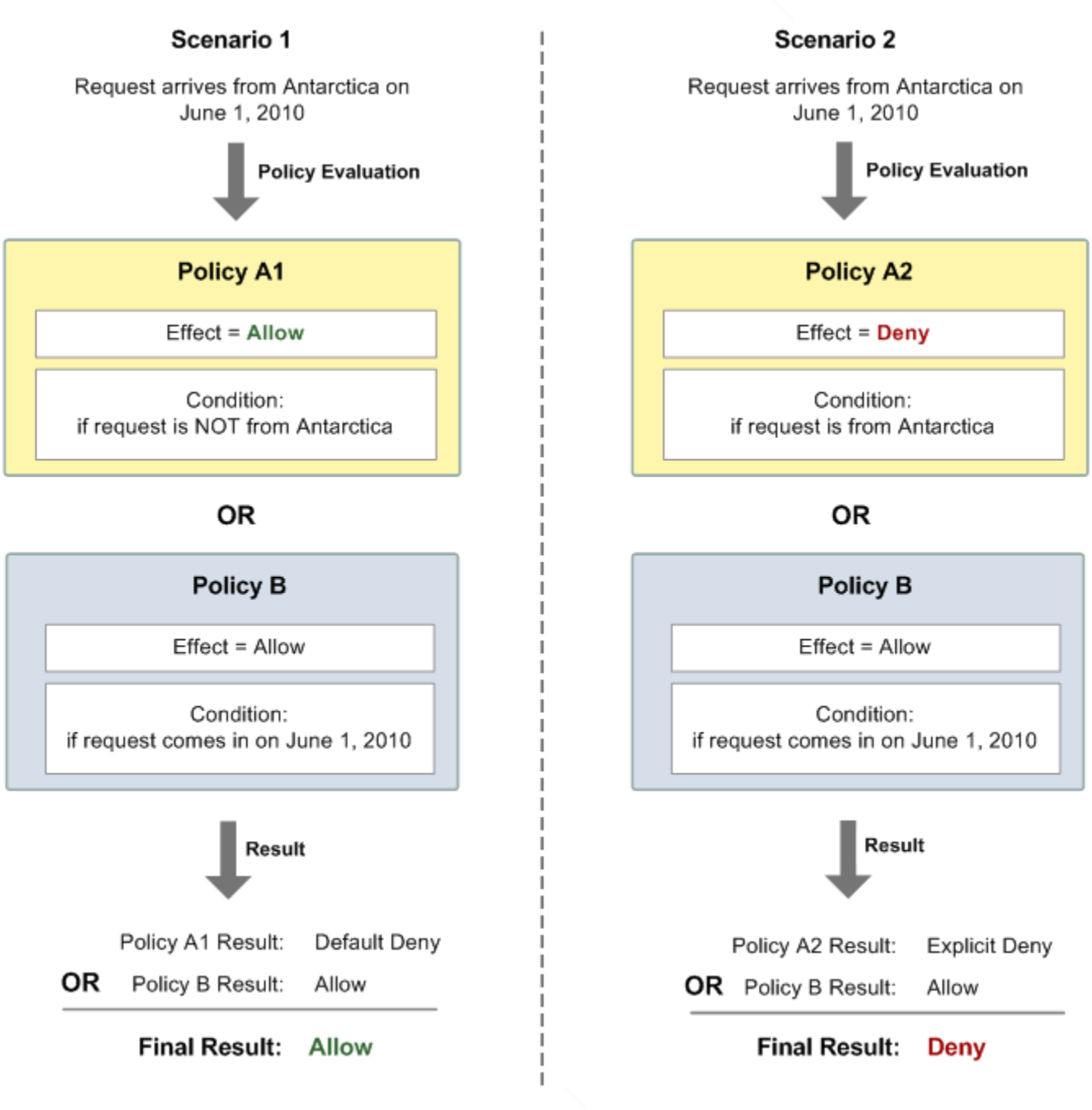
누군가가 미국에서 요청을 보낼 경우 남극 대륙에서 온 요청이 아니므로 조건을 충족합니다. 따라서 요청이 허용됩니다. 누군가가 남극 대륙에서 요청을 보낼 경우 조건을 충족하며 정책의 결과는 기본 거부가 됩니다.

다음 다이어그램처럼 정책을 다시 작성하여(정책 A2) 결과를 명시적 거부로 바꿀 수 있습니다. 이 정책은 남극 대륙에서 온 요청을 명시적으로 거부합니다.



누군가가 남극 대륙에서 요청을 보낼 경우 조건을 충족하므로 정책의 결과는 명시적 거부입니다.

기본 거부는 허용으로 재정의할 수 있지만 명시적 거부는 재정의할 수 없기 때문에 기본 거부와 명시적 거부를 반드시 구분해야 합니다. 예를 들어, 2010년 6월 1일에 도착하는 요청을 허용하는 또 다른 정책이 있다고 가정하겠습니다. 이 정책이 남극 대륙의 액세스를 제한하는 정책과 결합될 경우 종합적인 결과는 어떻게 됩니까? 날짜 기반 정책(정책 B)을 앞의 정책 A1 및 A2와 결합했을 때 종합적인 결과를 비교해보겠습니다. 시나리오 1은 정책 A1과 정책 B를, 시나리오 2에서는 정책 A2를 정책 B와 결합합니다. 다음 그림과 설명은 남극 대륙에서 보낸 요청이 2010년 6월 1일에 도착했을 때의 결과를 보여 줍니다.



시나리오 1에서는 이 섹션에서 설명한 것처럼 정책 A1이 기본 거부를 반환합니다. 정책 B는 허용을 반환합니다. 이 정책의 정의에 따라 2010년 6월 1일에 도착한 요청을 허용하기 때문입니다. 정책 B의 허용은 정책 A1의 기본 거부를 무시하므로 이 요청은 허용됩니다.

시나리오 2에서는 앞서 이 섹션에서 설명한 것처럼 정책 A2가 명시적 거부를 반환합니다. 역시 정책 B는 허용을 반환합니다. 정책 A2의 명시적 거부가 정책 B의 허용을 무시하므로 요청은 거부됩니다.

Amazon SNS 액세스 제어의 예제 사례

이 섹션에서는 몇 가지 예를 통해 액세스 제어의 일반적인 사용 사례를 보여줍니다.

주제

주제에 대한 AWS 계정 액세스 권한 부여

Amazon SNS 시스템에 어떤 주제가 있다고 가정합니다. 가장 단순한 경우로, 하나 이상의 AWS 계정에서 특정 주제 작업(예: 게시)에 액세스하는 것을 허용하려 합니다.

Amazon SNS API 작업 `AddPermission`을 사용하여 이렇게 할 수 있습니다. 이는 주제, AWS 계정 ID의 목록, 작업의 목록, 레이블을 가져와 주제의 액세스 제어 정책에서 새로운 설명을 자동으로 만듭니다. 여기서는 사용자가 직접 정책을 작성하지 않습니다. Amazon SNS에서 사용자를 대신하여 새로운 정책 설명을 자동으로 생성하기 때문입니다. 나중에 `RemovePermission`을 레이블과 함께 호출하여 이 정책 설명을 제거할 수 있습니다.

예를 들어 arn:aws:sns:us-east- 2:444455556666: 주제에 대해 MyTopic ID AWS 계정 1111-2222-3333과 작업 및 레이블을 사용하여 `AddPermission` 호출하면 Amazon SNS는 다음과 같은 액세스 제어 정책 설명을 생성하고 삽입합니다. `Publish` `grant-1234-publish`

```
{
  "Statement": [{
    "Sid": "grant-1234-publish",
    "Effect": "Allow",
    "Principal": {
      "AWS": "111122223333"
```

```
    },
    "Action": ["sns:Publish"],
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic"
  }]
}
```

이 설명이 추가되면 AWS 계정 1111-2222-3333의 사용자는 이 주제에 메시지를 게시할 수 있습니다.

HTTPS로 구독 제한

다음 예제에서는 알림 전송 프로토콜을 HTTPS로 제한합니다.

주제에 대한 정책을 직접 작성하는 방법을 알아야 합니다. Amazon SNS `AddPermission` 작업에서는 어떤 사람에게 주제에 대한 액세스 권한을 부여할 때 프로토콜 제한을 지정할 수 없기 때문입니다. 여기서는 직접 정책을 작성한 다음 `SetTopicAttributes` 작업을 사용하여 주제의 `Policy` 속성을 새로운 정책으로 설정합니다.

전체 정책을 보여주는 다음 예에서는 AWS 계정 ID 1111-2222-3333에 주제의 알림을 구독할 수 있는 권한을 부여합니다.

```
{
  "Statement": [{
    "Sid": "Statement1",
    "Effect": "Allow",
    "Principal": {
      "AWS": "111122223333"
    },
    "Action": ["sns:Subscribe"],
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
    "Condition": {
      "StringEquals": {
        "sns:Protocol": "https"
      }
    }
  }]
}
```

Amazon SQS 대기열에 메시지를 게시합니다.

이 사용 사례에서는 주제의 메시지를 Amazon SQS 대기열에 게시하려 합니다. Amazon SNS와 마찬가지로 Amazon SQS는 Amazon의 액세스 제어 정책 언어를 사용합니다. Amazon SNS에서 메시지를

전송할 수 있도록 허용하려면 Amazon SQS 작업 `SetQueueAttributes`을 사용하여 대기열에 대한 정책을 설정해야 합니다.

이 경우에도 정책을 직접 작성하는 방법을 알아야 합니다. Amazon SQS `AddPermission` 작업은 조건이 있는 정책 문을 생성하지 않기 때문입니다.

> **Note**
>
> 아래에 표시된 예는 대기열에 대한 액세스를 제어하는 Amazon SQS 정책이며, 주제에 대한 액세스를 제어하는 Amazon SNS 정책이 아닙니다. 작업은 Amazon SQS 작업이며, 리소스는 대기열의 ARN(Amazon 리소스 이름)입니다. `QueueArn` 작업으로 대기열의 `GetQueueAttributes` 속성을 검색하여 대기열의 ARN을 확인할 수 있습니다.

```
{
  "Statement": [{
    "Sid": "Allow-SNS-SendMessage",
    "Effect": "Allow",
    "Principal": {
      "Service": "sns.amazonaws.com"
    },
    "Action": ["sqs:SendMessage"],
    "Resource": "arn:aws:sqs:us-east-2:444455556666:MyQueue",
    "Condition": {
      "ArnEquals": {
        "aws:SourceArn": "arn:aws:sns:us-east-2:444455556666:MyTopic"
      }
    }
  }]
}
```

이 정책은 `aws:SourceArn` 조건을 사용하여 대기열에 보내지는 메시지의 출처에 따라 대기열에 대한 액세스를 제한합니다. 이 정책 유형을 사용하면 사용자의 주제 중 하나에서 나온 메시지에 한해 Amazon SNS에서 대기열에 메시지를 보내는 것을 허용할 수 있습니다. 이 경우 ARN이 arn:aws:sns:us-east- 2:444455556666:인 특정 주제를 지정해야 합니다. MyTopic

앞의 정책은 사용자가 작성하여 특정 대기열에 추가할 수 있는 Amazon SQS 정책의 예입니다. 이 정책은 Amazon SNS 및 기타 AWS 서비스에 대한 액세스 권한을 부여합니다. Amazon SNS에서 새로 생성된 모든 주제에 기본 정책을 부여합니다. 기본 정책은 주제에 대한 액세스 권한을 다른 모든 AWS 서비

스에 부여합니다. 이 기본 정책에서는 aws:SourceArn 조건을 사용하여 AWS 서비스가 사용자 소유의 AWS 리소스를 대신하는 경우에만 사용자의 주제에 액세스할 수 있게 합니다.

Amazon S3 이벤트 알림을 주제에 게시하도록 허용

여기서는 다른 AWS 계정의 Amazon S3 버킷에서 사용자의 주제에 게시할 수 있도록 주제의 정책을 구성하려 합니다. Amazon S3의 알림을 게시하는 방법에 대한 자세한 정보는 버킷 이벤트의 알림 설정을 참조하세요.

이 사례에서는 사용자가 직접 정책을 작성한 다음 SetTopicAttributes 작업을 사용하여 주제의 Policy 속성을 새로운 정책으로 설정한다고 가정합니다.

다음 예제 문은 SourceAccount 조건을 사용하여 Amazon S3 소유자 계정만 주제에 액세스할 수 있도록 합니다. 이 예제에서는 주제 소유자가 111122223333이고 Amazon S3 소유자는 444455556666입니다. 이 예제에서는 444455556666이 소유한 모든 Amazon S3 버킷을 게시할 수 있다고 명시하고 있습니다. MyTopic

```
{
  "Statement": [{
    "Effect": "Allow",
     "Principal": {
      "Service": "s3.amazonaws.com"
    },
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:us-east-2:111122223333:MyTopic",
    "Condition": {
      "StringEquals": {
        "AWS:SourceAccount": "444455556666"
      }
    }
  }]
}
```

Amazon SNS에 이벤트를 게시할 때 다음 서비스가 aws:SourceAccount를 지원합니다.

- Amazon API Gateway
- 아마존 CloudWatch
- 아마존 DevOps 전문가
- 아마존 ElastiCache
- 아마존 GameLift

- Amazon Pinpoint SMS 및 음성 API
- Amazon RDS
- Amazon Redshift
- Amazon S3 Glacier
- Amazon SES
- Amazon Simple Storage Service(S3)
- AWS CodeCommit
- AWS Directory Service
- AWS Lambda
- AWS Systems Manager Incident Manager

Amazon SES가 다른 계정이 소유한 주제에 게시하도록 허용

다른 AWS 서비스가 다른 AWS 계정이 소유한 주제에 게시하도록 허용할 수 있습니다. 111122223333 계정에 로그인하고 Amazon SES 를 연 다음 이메일을 생성했다고 가정합니다. 444455556666 계정이 소유한 Amazon SNS 주제에 이 이메일에 대한 알림을 게시하려면 다음과 같은 정책을 생성해야 합니다. 이 작업을 수행하려면 보안 주체(다른 서비스)와 각 리소스의 소유권에 대한 정보를 제공해야 합니다.. 이 `Resource` 문은 주제 소유자의 계정 ID인 444455556666을 포함하는 주제 ARN을 제공합니다. 이 `"aws:SourceOwner": "111122223333"` 문은 계정이 해당 이메일을 소유하도록 지정합니다.

```
{
  "Version": "2008-10-17",
  "Id": "__default_policy_ID",
  "Statement": [
    {
      "Sid": "__default_statement_ID",
      "Effect": "Allow",
      "Principal": {
        "Service": "ses.amazonaws.com"
      },
      "Action": "SNS:Publish",
      "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
      "Condition": {
        "StringEquals": {
          "aws:SourceOwner": "111122223333"
        }
      }
    }
```

```
  ]
}
```

Amazon SNS에 이벤트를 게시할 때 다음 서비스가 `aws:SourceOwner`를 지원합니다.

- Amazon API Gateway
- 아마존 CloudWatch
- 아마존 DevOps 전문가
- 아마존 ElastiCache
- 아마존 GameLift
- Amazon Pinpoint SMS 및 음성 API
- Amazon RDS
- Amazon Redshift
- Amazon SES
- AWS CodeCommit
- AWS Directory Service
- AWS Lambda
- AWS Systems Manager Incident Manager

`aws:SourceAccount`과 `aws:SourceOwner` 비교

> **Important**
>
> `aws:SourceOwner`는 더 이상 사용되지 않으며 새 서비스는 `aws:SourceArn` 및 `aws:SourceAccount`를 통해서만 Amazon SNS와 통합할 수 있습니다. Amazon SNS는 현재 `aws:SourceOwner`를 지원하는 기존 서비스에 대해 이전 버전과의 호환성을 여전히 유지합니다.

`aws:SourceAccount` 및 `aws:SourceOwner` 조건 키는 Amazon SNS 주제에 게시할 때 일부 AWS 서비스에서 각각 설정됩니다. 지원되는 경우 값은 서비스가 대신 데이터를 게시하는 12자리 AWS 계정 ID입니다. 일부 서비스는 하나를 지원하고 일부는 다르게 지원합니다.

- Amazon S3 알림이 `aws:SourceAccount`를 사용하는 방식과 해당 조건을 지원하는 AWS 서비스 목록은 Amazon S3 이벤트 알림을 주제에 게시하도록 허용에서 확인하세요.

- Amazon SES가 `aws:SourceOwner`를 사용하는 방법과 해당 조건을 지원하는 AWS 서비스 목록은 Amazon SES가 다른 계정이 소유한 주제에 게시하도록 허용에서 확인하세요..

AWS Organizations에서 조직의 계정이 다른 계정의 주제에 게시하도록 허용

이 AWS Organizations 서비스를 사용하면 중앙에서 결제를 관리하고, 액세스 및 보안을 제어하고, AWS 계정에서 리소스를 공유할 수 있습니다.

Organizations 콘솔에서 조직 ID를 찾을 수 있습니다. 자세한 정보는 관리 계정에서 조직 세부 정보 보기를 참조하세요.

이 예제에서는 조직 `myOrgId`의 모든 AWS 계정이 계정 `444455556666`의 Amazon SNS 주제 `MyTopic`에 게시할 수 있습니다. 이 정책은 `aws:PrincipalOrgID` 전역 조건 키를 사용하여 조직 ID 값을 확인합니다.

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "SNS:Publish",
            "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalOrgID": "myOrgId"
                }
            }
        }
    ]
}
```

모든 CloudWatch 알람이 다른 계정의 주제에 게시되도록 허용

이 경우 계정의 모든 CloudWatch 경보를 계정 `111122223333` 내 Amazon SNS 주제에 게시할 수 있습니다. `444455556666`

```
{
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "SNS:Publish",
            "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
            "Condition": {
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:cloudwatch:us-
east-2:111122223333:alarm:*"
                }
            }
        }
    ]
}
```

특정 VPC 엔드포인트에서만 Amazon SNS 주제에 게시할 수 있도록 제한

이 경우 계정 444455556666의 주제는 ID vpce-1ab2c34d의 VPC 엔드포인트에서만 게시할 수 있습니다.

```
{
  "Statement": [{
    "Effect": "Deny",
    "Principal": "*",
    "Action": "SNS:Publish",
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
    "Condition": {
      "StringNotEquals": {
        "aws:sourceVpce": "vpce-1ab2c34d"
      }
    }
  }]
}
```

Amazon SNS가 IAM으로 작동하는 방식

IAM을 사용하여 Amazon SNS에 대한 액세스를 관리하기 전에 Amazon SNS에서 사용할 수 있는 IAM 기능에 대해 알아봅니다.

Amazon SNS와 함께 사용할 수 있는 IAM 기능

IAM 특성	Amazon SNS 지원
ID 기반 정책	예
리소스 기반 정책	예
정책 작업	예
정책 리소스	예
정책 조건 키(서비스별)	예
ACL	아니요
ABAC(정책 내 태그)	부분
Temporary credentials(임시 보안 인증)	예
보안 주체 권한	예
Service roles(서비스 역할)	예
Service-linked roles(서비스 연결 역할)	아니요

Amazon SNS 및 기타 AWS 서비스가 대부분의 IAM 기능과 작동하는 방법을 개괄적으로 알아보려면 IAM 사용 설명서의 IAM으로 작업하는 AWS 서비스를 참조하세요.

Amazon SNS의 정책 작업

정책 작업 지원	예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지 지정할 수 있습니다.

JSON 정책의 `Action` 요소는 정책에서 액세스를 허용하거나 거부하는 데 사용할 수 있는 태스크를 설명합니다. 일반적으로 정책 작업의 이름은 연결된 AWS API 작업의 이름과 동일합니다. 일치하는 API

작업이 없는 권한 전용 작업 같은 몇 가지 예외도 있습니다. 정책에서 여러 작업이 필요한 몇 가지 작업도 있습니다. 이러한 추가 작업을 종속 작업이라고 합니다.

연결된 작업을 수행할 수 있는 권한을 부여하기 위한 정책에 작업을 포함합니다.

Amazon SNS 작업의 목록을 보려면 서비스 권한 부여 참조의 Amazon Simple Notification Service에서 정의한 작업을 참조하세요.

Amazon SNS의 정책 작업은 작업 앞에 다음 접두사를 사용합니다.

```
sns
```

단일 문에서 여러 작업을 지정하려면 다음과 같이 쉼표로 구분합니다.

```
"Action": [
      "sns:action1",
      "sns:action2"
         ]
```

Amazon SNS 자격 증명 기반 정책 예제를 보려면 Amazon Simple Notification Service의 자격 증명 기반 정책 예 단원을 참조하세요.

Amazon SNS의 정책 리소스

정책 리소스 지원	예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지 지정할 수 있습니다.

`Resource` JSON 정책 요소는 작업이 적용되는 하나 이상의 객체를 지정합니다. 문장에는 `Resource` 또는 `NotResource` 요소가 반드시 추가되어야 합니다. 모범 사례에 따라 Amazon 리소스 이름(ARN)을 사용하여 리소스를 지정합니다. 리소스 수준 권한이라고 하는 특정 리소스 유형을 지원하는 작업에 대해 이 작업을 수행할 수 있습니다.

작업 나열과 같이 리소스 수준 권한을 지원하지 않는 작업의 경우, 와일드카드(*)를 사용하여 해당 문이 모든 리소스에 적용됨을 나타냅니다.

```
"Resource": "*"
```

Amazon SNS 리소스 유형 및 해당 ARN의 목록을 보려면 서비스 권한 부여 참조의 Amazon Simple Notification Service에서 정의한 작업을 참조하세요. 각 리소스의 ARN을 지정할 수 있는 작업을 알아보려면 Amazon Simple Notification Service에서 정의한 리소스를 참조하세요.

Amazon SNS 자격 증명 기반 정책 예제를 보려면 Amazon Simple Notification Service의 자격 증명 기반 정책 예 단원을 참조하세요.

Amazon SNS의 정책 조건 키

서비스별 정책 조건 키 지원	예

관리자는 AWS JSON 정책을 사용하여 누가 무엇에 액세스할 수 있는지를 지정할 수 있습니다. 즉, 어떤 보안 주체가 어떤 리소스와 어떤 조건에서 작업을 수행할 수 있는지 지정할 수 있습니다.

`Condition` 요소(또는 `Condition` 블록)를 사용하면 정책이 발효되는 조건을 지정할 수 있습니다. `Condition` 요소는 선택 사항입니다. 같음이나 미만 같은 조건 연산자를 사용하여 정책의 조건을 요청의 값과 일치시키는 조건식을 생성할 수 있습니다.

한 문에서 여러 `Condition` 요소를 지정하거나 단일 `Condition` 요소에서 여러 키를 지정하는 경우 AWS는 논리적 `AND` 태스크를 사용하여 평가합니다. 단일 조건 키의 여러 값을 지정하는 경우 AWS는 논리적 `OR` 태스크를 사용하여 조건을 평가합니다. 문의 권한을 부여하기 전에 모든 조건을 충족해야 합니다.

조건을 지정할 때 자리표시자 변수를 사용할 수도 있습니다. 예를 들어, IAM 사용자에게 IAM 사용자 이름으로 태그가 지정된 경우에만 리소스에 액세스할 수 있는 권한을 부여할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM 정책 요소: 변수 및 태그를 참조하세요.

AWS은(는) 전역 조건 키와 서비스별 조건 키를 지원합니다. 모든 AWS 전역 조건 키를 보려면 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

Amazon SNS 조건 키 목록을 보려면 서비스 권한 부여 참조의 Amazon Simple Notification Service에 사용되는 조건 키를 참조하세요. 조건 키를 사용할 수 있는 작업과 리소스를 알아보려면 Amazon Simple Notification Service에서 정의한 리소스를 참조하세요.

Amazon SNS 자격 증명 기반 정책 예제를 보려면 Amazon Simple Notification Service의 자격 증명 기반 정책 예 단원을 참조하세요.

Amazon SNS의 ACL

ACL 지원	아니요

ACL(액세스 제어 목록)은 어떤 보안 주체(계정 멤버, 사용자 또는 역할)가 리소스에 액세스할 수 있는 권한을 가지고 있는지를 제어합니다. ACL은 JSON 정책 문서 형식을 사용하지 않지만 리소스 기반 정책과 유사합니다.

Amazon SNS의 ABAC

ABAC(정책 내 태그) 지원	부분

속성 기반 액세스 제어(ABAC)는 속성을 기반으로 권한을 정의하는 권한 부여 전략입니다. AWS에서는 이러한 속성을 태그라고 합니다. IAM 엔터티(사용자 또는 역할) 및 많은 AWS 리소스에 태그를 연결할 수 있습니다. ABAC의 첫 번째 단계로 엔터티 및 리소스에 태그를 지정합니다. 그런 다음 보안 주체의 태그가 액세스하려는 리소스의 태그와 일치할 때 작업을 허용하도록 ABAC 정책을 설계합니다.

ABAC는 빠르게 성장하는 환경에서 유용하며 정책 관리가 번거로운 상황에 도움이 됩니다.

태그를 기반으로 액세스를 제어하려면 `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`또는 `aws:TagKeys` 조건 키를 사용하여 정책의 조건 요소에 태그 정보를 제공합니다.

서비스가 모든 리소스 유형에 대해 세 가지 조건 키를 모두 지원하는 경우 값은 서비스에 대해 예입니다. 서비스가 일부 리소스 유형에 대해서만 세 가지 조건 키를 모두 지원하는 경우 값은 부분적입니다.

ABAC에 대한 자세한 내용은 IAM 사용 설명서의 ABAC란 무엇입니까?를 참조하세요. ABAC 설정 단계가 포함된 자습서를 보려면 IAM 사용 설명서의 속성 기반 액세스 제어(ABAC) 사용을 참조하세요.

Amazon SNS에서 임시 자격 증명 사용

임시 보안 인증 지원	예

일부 AWS 서비스은(는) 임시 보안 인증을 사용하여 로그인할 때 작동하지 않습니다. 임시 보안 인증으로 작동하는 AWS 서비스를 비롯한 추가 정보는 IAM 사용 설명서의 IAM을 사용하는 AWS 서비스을(를) 참조하세요.

사용자 이름과 암호를 제외한 다른 방법을 사용하여 AWS Management Console에 로그인하면 임시 보안 인증을 사용하는 것입니다. 예를 들어 회사의 Single Sign-On(SSO) 링크를 사용하여 AWS에 액세스하면 해당 프로세스에서 자동으로 임시 보안 인증을 생성합니다. 또한 콘솔에 사용자로 로그인한 다음 역할을 전환할 때 임시 보안 인증을 자동으로 생성합니다. 역할 전환에 대한 자세한 내용은 IAM 사용 설명서의 역할로 전환(콘솔)을 참조하세요.

AWS CLI 또는 AWS API를 사용하여 임시 보안 인증을 수동으로 만들 수 있습니다 그런 다음 이러한 임시 보안 인증을 사용하여 AWS에 액세스할 수 있습니다. AWS에서는 장기 액세스 키를 사용하는 대신 임시 보안 인증을 동적으로 생성할 것을 권장합니다. 자세한 내용은 IAM의 임시 보안 자격 증명을(를) 참조하세요.

Amazon SNS에 대한 교차 서비스 보안 주체 권한

전달 액세스 세션(FAS) 지원	예

IAM 사용자 또는 역할을 사용하여 AWS에서 작업을 수행하는 사람은 보안 주체로 간주됩니다. 일부 서비스를 사용하는 경우 다른 서비스에서 다른 작업을 시작하는 작업을 수행할 수 있습니다. FAS는 AWS 서비스를 직접 호출하는 보안 주체의 권한과 요청하는 AWS 서비스를 함께 사용하여 다운스트림 서비스에 대한 요청을 수행합니다. FAS 요청은 서비스에서 완료를 위해 다른 AWS 서비스 또는 리소스와의 상호 작용이 필요한 요청을 받은 경우에만 이루어집니다. 이 경우 두 작업을 모두 수행할 수 있는 권한이 있어야 합니다. FAS 요청 시 정책 세부 정보는 전달 액세스 세션을 참조하십시오.

Amazon SNS의 서비스 역할

서비스 역할 지원	예

서비스 역할은 서비스가 사용자를 대신하여 작업을 수행하는 것으로 가정하는 IAM role(IAM 역할)입니다. IAM 관리자는 IAM 내에서 서비스 역할을 생성, 수정 및 삭제할 수 있습니다. 자세한 내용은 IAM 사용 설명서의 AWS 서비스에 대한 권한을 위임할 역할 생성을 참조하세요.

> **Warning**
>
> 서비스 역할에 대한 권한을 변경하면 Amazon SNS 기능이 중단될 수 있습니다. Amazon SNS에서 관련 지침을 제공하는 경우에만 서비스 역할을 편집합니다.

Amazon SNS의 서비스 연결 역할

서비스 연결 역할 지원	아니요

서비스 연결 역할은 AWS 서비스에 연결된 서비스 역할의 한 유형입니다. 서비스는 사용자를 대신하여 작업을 수행하기 위해 역할을 수임할 수 있습니다. 서비스 링크 역할은 AWS 계정에 나타나고, 서비스가 소유합니다. IAM 관리자는 서비스 링크 역할의 권한을 볼 수 있지만 편집은 할 수 없습니다.

서비스 연결 역할 생성 또는 관리에 대한 자세한 내용은 IAM으로 작업하는 AWS서비스을(를) 참조하세요. Service-linked role(서비스 연결 역할) 열에서 Yes이(가) 포함된 서비스를 테이블에서 찾습니다. 해당 서비스에 대한 서비스 연결 역할 설명서를 보려면 Yes(네) 링크를 선택합니다.

Amazon Simple Notification Service의 자격 증명 기반 정책 예

기본적으로 사용자 및 역할은 Amazon SNS 리소스를 생성하거나 수정할 수 있는 권한이 없습니다. 또한 AWS Management Console, AWS Command Line Interface(AWS CLI) 또는 AWS API를 사용해 태스크를 수행할 수 없습니다. 사용자에게 사용자가 필요한 리소스에서 작업을 수행할 권한을 부여하려면 IAM 관리자가 IAM 정책을 생성하면 됩니다. 그런 다음 관리자가 IAM 정책을 역할에 추가하고, 사용자가 역할을 맡을 수 있습니다.

이러한 예제 JSON 정책 문서를 사용하여 IAM ID 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

각 리소스 유형에 대한 ARN 형식을 포함하여 Amazon SNS에서 정의한 작업 및 리소스 유형에 대한 자세한 내용은 서비스 권한 부여 참조에서 Amazon Simple Notification Service에 대한 작업, 리소스 및 조건 키를 참조하세요.

주제

- 정책 모범 사례
- Amazon SNS 콘솔 사용
- 기타 정책 유형
- 여러 정책 유형
- 사용자가 자신의 고유한 권한을 볼 수 있도록 허용

정책 모범 사례

ID 기반 정책에 따라 계정에서 사용자가 Amazon SNS 리소스를 생성, 액세스 또는 삭제할 수 있는지 여부가 결정됩니다. 이 작업으로 인해 AWS 계정에 비용이 발생할 수 있습니다. 자격 증명 기반 정책을 생성하거나 편집할 때는 다음 지침과 권장 사항을 따르세요.

- AWS 관리형 정책으로 시작하고 최소 권한을 향해 나아가기 - 사용자 및 워크로드에 권한 부여를 시작하려면 많은 일반 사용 사례에 대한 권한을 부여하는 AWS 관리형 정책을 사용합니다. AWS 계정에서 사용할 수 있습니다. 사용 사례에 고유한 AWS 고객 관리형 정책을 정의하여 권한을 줄이는 것이 좋습니다. 자세한 정보는 IAM 사용 설명서의 AWS 관리형 정책 또는 AWS 직무에 대한 관리형 정책을 참조하세요.
- Apply least-privilege permissions(최소 권한 적용) – IAM 정책을 사용하여 권한을 설정하는 경우 태스크를 수행하는 데 필요한 권한만 부여합니다. 이렇게 하려면 least-privilege permissions(최소 권한)으로 알려진 특정 조건에서 특정 리소스에 대해 수행할 수 있는 작업을 정의합니다. IAM을 사용하여 권한을 적용하는 방법에 대한 자세한 정보는 IAM 사용 설명서에 있는 IAM의 정책 및 권한을 참조하세요.
- Use conditions in IAM policies to further restrict access(IAM 정책의 조건을 사용하여 액세스 추가 제한) – 정책에 조건을 추가하여 작업 및 리소스에 대한 액세스를 제한할 수 있습니다. 예를 들어 SSL을 사용하여 모든 요청을 전송해야 한다고 지정하는 정책 조건을 생성할 수 있습니다. 특정 AWS 서비스(예: AWS CloudFormation)를 통해 사용되는 경우에만 서비스 작업에 대한 액세스 권한을 부여할 수도 있습니다. 자세한 내용은 IAM 사용 설명서의 IAM JSON 정책 요소: 조건을 참조하세요.
- IAM Access Analyzer를 통해 IAM 정책을 검증하여 안전하고 기능적인 권한 보장 – IAM Access Analyzer에서는 IAM 정책 언어(JSON)와 모범 사례가 정책에서 준수되도록 신규 및 기존 정책을 검증합니다. IAM Access Analyzer는 100개 이상의 정책 확인 항목과 실행 가능한 권장 사항을 제공하여 안전하고 기능적인 정책을 생성하도록 돕습니다. 자세한 정보는 IAM 사용 설명서의 IAM Access Analyzer 정책 검증을 참조하세요.
- Require multi-factor authentication(MFA) (다중 인증 필요) – AWS 계정 계정에 IAM 사용자 또는 루트 사용자가 필요한 시나리오가 있는 경우 추가 보안을 위해 MFA를 설정합니다. API 작업을 호출할 때 MFA를 요구하려면 정책에 MFA 조건을 추가합니다. 자세한 정보는 IAM 사용 설명서의 MFA 보호 API 액세스 구성을 참조하세요.

IAM의 모범 사례에 대한 자세한 내용은 IAM 사용 설명서의 IAM의 보안 모범 사례를 참조하세요.

Amazon SNS 콘솔 사용

Amazon Simple Notification Service 콘솔에 액세스하려면 최소한의 권한 집합이 있어야 합니다. 이러한 권한은 AWS 계정에서 Amazon SNS 리소스에 대한 세부 정보를 나열하고 볼 수 있도록 허용해야

합니다. 최소 필수 권한보다 더 제한적인 ID 기반 정책을 만들면 콘솔이 해당 정책에 연결된 개체(사용자 또는 역할)에 대해 의도대로 작동하지 않습니다.

AWS CLI 또는 AWS API만 호출하는 사용자에게 최소 콘솔 권한을 허용할 필요가 없습니다. 그 대신, 수행하려는 API 작업과 일치하는 작업에만 액세스할 수 있도록 합니다.

사용자와 역할이 Amazon SNS 콘솔을 계속해서 사용할 수 있도록 하려면 Amazon SNS *ConsoleAccess* 또는 *ReadOnly* AWS 관리형 정책을 엔터티에 추가합니다. 자세한 내용은 IAM 사용 설명서의 사용자에게 권한 추가를 참조하십시오.

기타 정책 유형

AWS은(는) 비교적 일반적이지 않은 추가 정책 유형을 지원합니다. 이러한 정책 유형은 더 일반적인 정책 유형에 따라 사용자에게 부여되는 최대 권한을 설정할 수 있습니다.

- 권한 경계 – 권한 경계는 ID 기반 정책에 따라 IAM 엔터티(IAM 사용자 또는 역할)에 부여할 수 있는 최대 권한을 설정하는 고급 기능입니다. 개체에 대한 권한 경계를 설정할 수 있습니다. 그 결과로 얻는 권한은 개체의 ID 기반 정책과 그 권한 경계의 교집합입니다. `Principal` 필드에서 사용자나 역할을 보안 주체로 지정하는 리소스 기반 정책은 권한 경계를 통해 제한되지 않습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 권한 경계에 대한 자세한 정보는 IAM 사용 설명서의 IAM 엔터티에 대한 권한 경계 섹션을 참조하세요.
- 서비스 제어 정책(SCP) – SCP는 AWS Organizations에서 조직 또는 조직 단위(OU)에 최대 권한을 지정하는 JSON 정책입니다. AWS Organizations은(는) 기업이 소유하는 여러 개의 AWS 계정을(를) 그룹화하고 중앙에서 관리하기 위한 서비스입니다. 조직에서 모든 기능을 활성화할 경우 서비스 제어 정책(SCP)을 임의의 또는 모든 계정에 적용할 수 있습니다. SCP는 각 AWS 계정 루트 사용자를 비롯하여 멤버 계정의 엔터티에 대한 권한을 제한합니다. 조직 및 SCP에 대한 자세한 정보는 AWS Organizations 사용 설명서의 SCP 작동 방식을 참조하세요.
- 세션 정책 – 세션 정책은 역할 또는 페더레이션 사용자에 대해 임시 세션을 프로그래밍 방식으로 생성할 때 파라미터로 전달하는 고급 정책입니다. 결과적으로 얻는 세션의 권한은 사용자 또는 역할의 ID 기반 정책의 교차와 세션 정책입니다. 또한 권한을 리소스 기반 정책에서 가져올 수도 있습니다. 이러한 정책 중 하나에 포함된 명시적 거부는 허용을 재정의합니다. 자세한 정보는 IAM 사용 설명서의 세션 정책을 참조하세요.

여러 정책 유형

여러 정책 유형이 요청에 적용되는 경우, 결과 권한은 이해하기가 더 복잡합니다. 여러 정책 유형이 관련될 때 AWS가 요청을 허용할지 여부를 결정하는 방법을 알아보려면 IAM 사용 설명서의 정책 평가 로직을 참조하십시오.

사용자가 자신의 고유한 권한을 볼 수 있도록 허용

이 예제는 IAM 사용자가 자신의 사용자 보안 인증에 연결된 인라인 및 관리형 정책을 볼 수 있도록 허용하는 정책을 생성하는 방법을 보여줍니다. 이 정책에는 콘솔에서 또는 AWS CLI나 AWS API를 사용하여 프로그래밍 방식으로 이 작업을 완료할 수 있는 권한이 포함됩니다.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon SNS의 자격 증명 기반 정책

자격 증명 기반 정책 지원	예

ID 기반 정책은 IAM 사용자, 사용자 그룹 또는 역할과 같은 ID에 연결할 수 있는 JSON 권한 정책 문서입니다. 이러한 정책은 사용자와 역할이 어떤 리소스와 어떤 조건에서 어떤 태스크를 수행할 수 있는지를 제어합니다. 자격 증명 기반 정책을 생성하는 방법을 알아보려면 IAM 사용 설명서의 IAM 정책 생성을 참조하세요.

IAM 자격 증명 기반 정책을 사용하면 허용되거나 거부되는 작업과 리소스뿐 아니라 작업이 허용되거나 거부되는 조건을 지정할 수 있습니다. 자격 증명 기반 정책에서는 보안 주체가 연결된 사용자 또는 역할에 적용되므로 보안 주체를 지정할 수 없습니다. JSON 정책에서 사용하는 모든 요소에 대해 알아보려면 IAM 사용 설명서의 IAM JSON 정책 요소 참조를 참조하세요.

Amazon SNS의 자격 증명 기반 정책 예

Amazon SNS 자격 증명 기반 정책 예제를 보려면 Amazon Simple Notification Service의 자격 증명 기반 정책 예 단원을 참조하세요.

Amazon SNS 내의 리소스 기반 정책

리소스 기반 정책 지원	예

리소스 기반 정책은 리소스에 연결하는 JSON 정책 설명서입니다. 리소스 기반 정책의 예는 IAM 역할 신뢰 정책과 Amazon S3 버킷 정책입니다. 리소스 기반 정책을 지원하는 서비스에서 서비스 관리자는 이러한 정책을 사용하여 특정 리소스에 대한 액세스를 제어할 수 있습니다. 정책이 연결된 리소스의 경우 정책은 지정된 보안 주체가 해당 리소스와 어떤 조건에서 어떤 태스크를 수행할 수 있는지를 정의합니다. 리소스 기반 정책에서 보안 주체를 지정해야 합니다. 보안 주체에는 계정, 사용자, 역할, 페더레이션 사용자 또는 AWS 서비스이(가) 포함될 수 있습니다.

크로스 계정 액세스를 활성화하려는 경우 전체 계정이나 다른 계정의 IAM 엔터티를 리소스 기반 정책의 보안 주체로 지정할 수 있습니다. 리소스 기반 정책에 크로스 계정 보안 주체를 추가하는 것은 신뢰 관계 설정의 절반밖에 되지 않는다는 것을 유념하세요. 보안 주체와 리소스가 서로 다른 AWS 계정에 있는 경우 신뢰할 수 있는 계정의 IAM 관리자는 보안 주체 엔터티(사용자 또는 역할)에도 리소스 액세스 권한을 부여해야 합니다. 엔터티에 자격 증명 기반 정책을 연결하여 권한을 부여합니다. 하지만 리

소스 기반 정책이 동일 계정의 보안 주체에 액세스를 부여하는 경우 추가 자격 증명 기반 정책이 필요하지 않습니다. 자세한 내용은 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하세요.

Amazon SNS로 자격 증명 기반 정책 사용

주제

Amazon Simple Notification Service는 AWS Identity and Access Management(IAM)와 통합되므로 AWS 계정의 사용자가 Amazon SNS 리소스로 수행할 수 있는 Amazon SNS 작업을 지정할 수 있습니다. 정책에서 특정 주제를 지정할 수 있습니다. 예를 들어, AWS 계정의 특정 주제와 함께 `Publish` 작업을 사용할 권한을 조직의 특정 사용자에게 부여하는 IAM 정책을 생성할 때 변수를 사용할 수 있습니다. 자세한 정보는 IAM 사용 가이드의 정책 변수를 참조하세요.

> **Important**
>
> Amazon SNS를 IAM과 함께 사용하더라도 Amazon SNS 사용 방법에는 변화가 없습니다. Amazon SNS 작업은 변경되지 않으며 사용자 및 액세스 제어와 관련된 새로운 Amazon SNS 작업도 없습니다.

Amazon SNS 작업 및 리소스에 대한 정책의 예는 Amazon SNS에 대한 정책 예제에서 확인하세요.

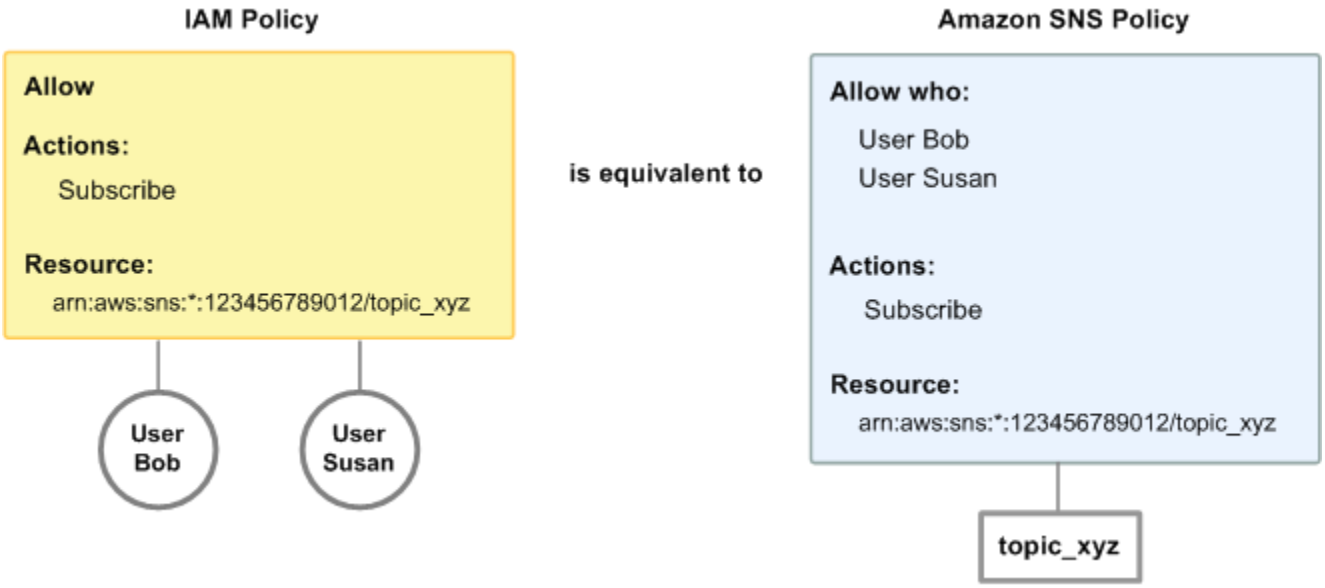
IAM 및 Amazon SNS 정책을 함께 사용

IAM 정책을 사용하여 Amazon SNS 작업 및 주제에 대한 사용자의 액세스를 제한합니다. IAM 정책을 사용하여 다른 AWS 계정이 아닌 AWS 계정 내 사용자에게만 액세스를 제한할 수 있습니다.

Amazon SNS 정책을 특정 주제와 함께 사용하여 해당 주제를 작업할 수 있는 사람(예: 해당 주제에 메시지를 게시할 수 있는 사람, 해당 주제를 구독할 수 있는 사람 등)을 제한합니다. Amazon SNS 정책은 다른 AWS 계정이나 자체 AWS 계정에게 액세스 권한을 부여할 수 있습니다.

사용자에게 Amazon SNS 주제에 대한 권한을 부여하기 위해 IAM 정책, Amazon SNS 정책 또는 둘 다를 사용할 수 있습니다. 대개는 어떤 방법으로도 같은 결과를 얻을 수 있습니다. 예를 들어, 다음 다이어

그램은 동일한 IAM 정책과 Amazon SNS 정책을 보여줍니다. IAM 정책에서는 AWS 계정의 topic_xyz라는 주제에 대한 Amazon SNS `Subscribe` 작업을 허용합니다. IAM 정책은 사용자 Bob과 Susan에게 연결됩니다. 즉 Bob과 Susan은 정책에 명시된 권한을 갖습니다. Amazon SNS 정책도 같은 방식으로 Bob과 Susan에게 topic_xyz에 대한 `Subscribe`에 액세스할 수 있는 권한을 부여합니다.



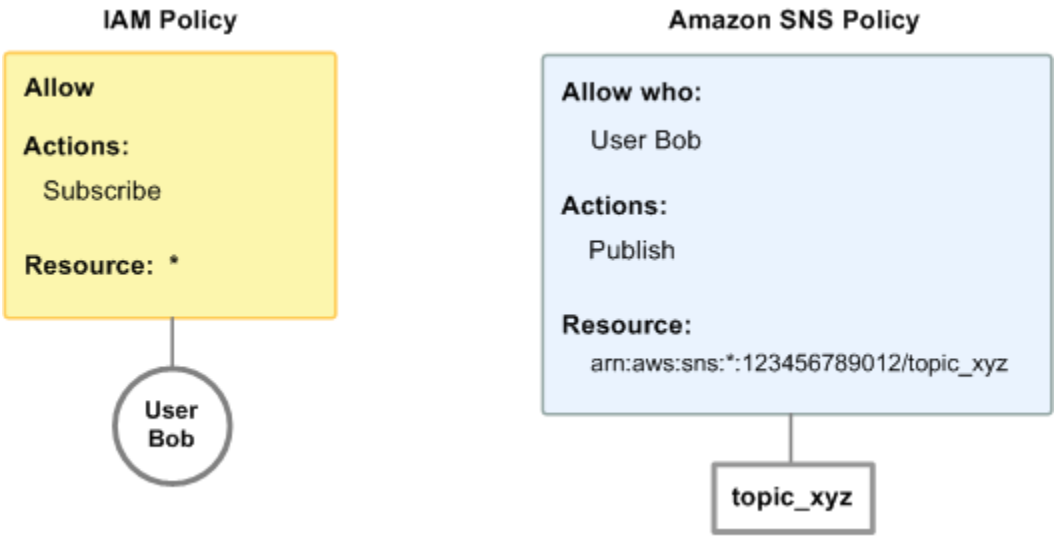
> **Note**
>
> 이전 예제에서는 조건없는 간단한 정책을 보여줍니다. 두 정책 중 어느 쪽에도 특정 조건을 지정할 수 있으며 동일한 결과를 얻을 수 있습니다.

AWS IAM 정책과 Amazon SNS 정책 간에는 한 가지 다른 점이 있습니다. Amazon SNS 정책 시스템에서는 다른 AWS 계정에 권한을 부여할 수 있는 반면, IAM 정책은 그렇지 않습니다.

사용자의 필요에 따라 두 시스템을 함께 효과적으로 사용하면서 권한을 관리해야 합니다. 다음 예는 두 정책 시스템이 어떻게 연계하는지 보여줍니다.

Example 1

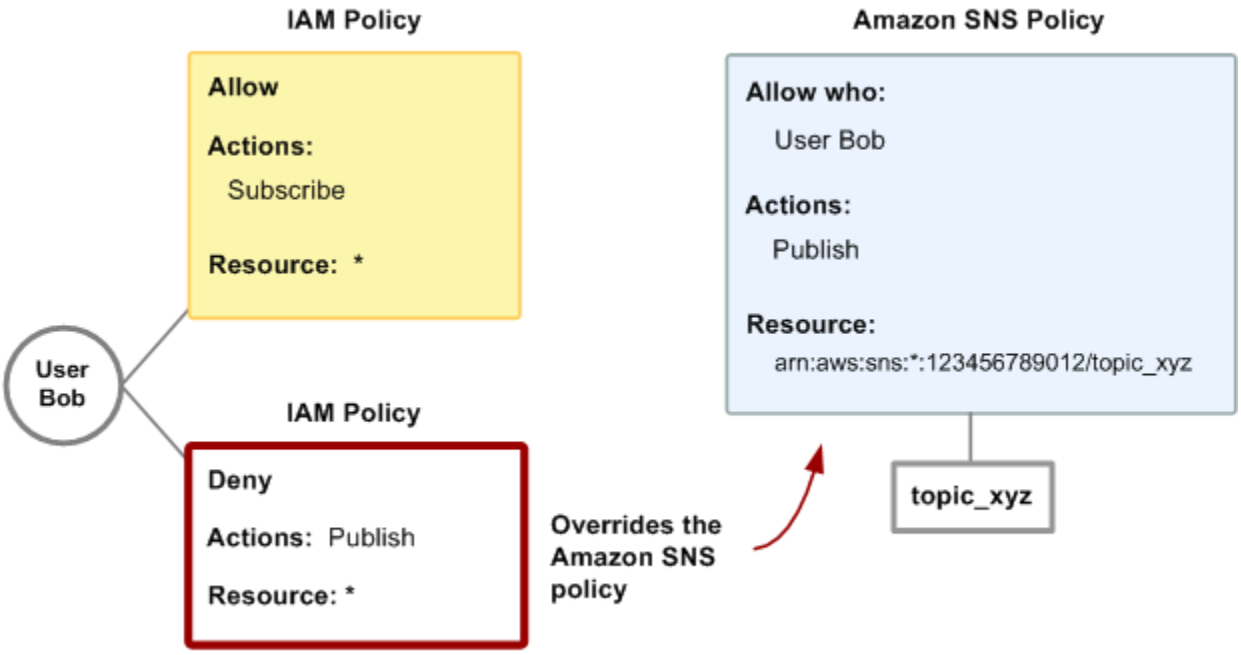
이 예제에서는 IAM 정책과 Amazon SNS 정책이 Bob에게 모두 적용됩니다. IAM 정책은 Bob에게 AWS 계정의 모든 주제에 대해 `Subscribe` 권한을 부여하는 반면 Amazon SNS 정책은 Bob에게 특정 주제(topic_xyz)에 대해서만 `Publish`를 사용할 권한을 부여합니다. 다음 다이어그램에서 관련 개념을 설명합니다.



Bob이 AWS 계정의 어떤 주제를 구독하기 위해 요청을 보내면 IAM 정책에서 이 작업을 허용합니다. Bob이 topic_xyz에 메시지를 게시하기 위해 요청을 보내면 Amazon SNS 정책에서 이 작업을 허용합니다.

Example 2

여기서는 예 1(Bob에게 2가지 정책이 적용됨)을 토대로 합니다. Bob이 opic_xyz에 메시지를 게시하는데, 사실 그에게는 권한이 없습니다. 따라서 그가 주제에 게시하는 것을 아예 금지하려 합니다. 가장 쉬운 방법은 모든 주제에서 `Publish` 작업에 대한 액세스를 거부하는 IAM 정책을 추가하는 것입니다. 이 세 번째 정책은 원래 topic_xyz에 대한 게시할 권한을 부여한 Amazon SNS 정책을 재정의합니다. 명시적 거부는 항상 허용을 재정의하기 때문입니다(정책 평가 로직에 대한 자세한 정보는 평가 로직 섹션 참조). 다음 다이어그램에서 관련 개념을 설명합니다.



Amazon SNS 작업 및 리소스에 대한 정책의 예는 Amazon SNS에 대한 정책 예제에서 확인하세요. Amazon SNS 정책 작성에 대한 자세한 정보는 Amazon SNS에 대한 기술 설명서를 참조하세요.

Amazon SNS 리소스 ARN 형식

Amazon SNS에서는 주제가 정책에서 지정할 수 있는 유일한 리소스 유형입니다. 다음은 주제의 Amazon 리소스 이름(ARN) 형식입니다.

```
arn:aws:sns:region:account_ID:topic_name
```

ARN에 대한 자세한 내용을 확인하려면 IAM 사용 설명서의 ARN로 이동하세요.

Example

다음은 123456789012에 속하는 us-east-2 MyTopic 지역에서 이름이 지정된 주제에 대한 ARN입니다. AWS 계정

```
arn:aws:sns:us-east-2:123456789012:MyTopic
```

Example

Amazon SNS가 지원하는 각 MyTopic 지역에 이름이 지정된 주제가 있는 경우 다음 ARN을 사용하여 주제를 지정할 수 있습니다.

```
arn:aws:sns:*:123456789012:MyTopic
```

주제 이름에 *와 ?를 사용할 수 있습니다. 예를 들어, 다음은 Bob이 생성했고 bob_라는 접두사를 붙인 모든 주제를 가리킬 수 있습니다.

```
arn:aws:sns:*:123456789012:bob_*
```

사용자의 편의를 위해 사용자가 주제를 만들면 Amazon SNS는 그 주제의 ARN을 반환합니다.

Amazon SNS API 작업

IAM 정책에서는 Amazon SNS에서 제공하는 모든 작업을 지정할 수 있습니다. 그러나 `ConfirmSubscription` 및 `Unsubscribe` 작업에는 인증이 필요하지 않습니다. 즉 정책에서 해당 작업을 지정하더라도 IAM은 해당 작업에 대한 사용자의 액세스를 제한하지 않습니다.

정책에서 지정하는 각 작업은 소문자 문자열의 접두사가 붙어야 합니다`sns:`. 예를 들어, 모든 Amazon SNS 작업을 지정하려면 `sns:*`를 사용합니다. 작업 목록을 보려면 Amazon Simple Notification Service API 참조로 이동하세요.

Amazon SNS 정책 키

Amazon SNS는 다음 AWS 전체 정책 키와 일부 서비스별 키를 구현합니다.

각 AWS 서비스에서 지원하는 조건 키 목록은 IAM 사용 설명서에서 AWS 서비스에 사용되는 작업, 리소스 및 조건 키를 참조하세요. 여러 AWS 서비스에서 사용할 수 있는 조건 키 목록은 IAM 사용 설명서의 AWS 전역 조건 컨텍스트 키를 참조하세요.

Amazon SNS는 다음과 같은 서비스별 키를 사용합니다. `Subscribe` 요청에 대한 액세스를 제한하는 정책에서 이 키를 사용합니다.

- sns:endpoint— `Subscribe` 요청 또는 이전에 확인된 구독의 URL, 이메일 주소 또는 ARN입니다. 특정 엔드포인트(예: *@yourcompany.com)에 대한 액세스를 제한하려면 문자열 조건과 함께 사용합니다(Amazon SNS에 대한 정책 예제 참조).
- sns:protocol— `Subscribe` 요청 또는 이전에 확인한 구독의 `protocol` 값입니다. 특정 전송 프로토콜(예: https)에 대한 게시를 제한하려면 문자열 조건과 함께 사용합니다(Amazon SNS에 대한 정책 예제 참조).

Amazon SNS에 대한 정책 예제

여기서는 Amazon SNS에 대한 사용자 액세스를 제어하는 몇몇 간단한 정책을 보여줍니다.

> **Note**
>
> 나중에 Amazon SNS는 정책에 명시된 목표를 기반으로 다음 정책 중 하나에 논리적으로 포함되어야 하는 새로운 작업을 추가할 수 있습니다.

Example 1: 어떤 그룹에서 주제를 만들고 관리하도록 허용

여기서는 `CreateTopic`, `ListTopics`, `SetTopicAttributes`, `DeleteTopic`에 대한 액세스 권한을 부여하는 정책을 만듭니다.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": ["sns:CreateTopic", "sns:ListTopics", "sns:SetTopicAttributes",
 "sns:DeleteTopic"],
    "Resource": "*"
  }]
}
```

Example 2: IT 그룹에서 특정 주제에 메시지를 게시하는 것을 허용

여기서는 IT를 위한 그룹을 만들고 관심 있는 특정 주제에 대해 `Publish`에 액세스할 수 있는 권한을 부여하는 정책을 할당합니다.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:*:123456789012:MyTopic"
  }]
}
```

Example 3: AWS 계정의 사용자에게 주제를 구독할 수 있는 권한 부여

여기서는 `Subscribe` 작업에 대한 액세스 권한을 부여하는 정책을 만듭니다. 이를 위해 `sns:Protocol` 및 `sns:Endpoint` 정책 키에 대한 문자열 일치 조건을 사용합니다.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": ["sns:Subscribe"],
    "Resource": "*",
    "Condition": {
      "StringLike": {
        "SNS:Endpoint": "*@example.com"
      },
      "StringEquals": {
        "sns:Protocol": "email"
      }
    }
  }]
}
```

Example 4: 파트너가 특정 주제에 메시지를 게시하는 것을 허용

Amazon SNS 정책 또는 IAM 정책을 사용하여 파트너가 특정 주제에 게시하도록 허용할 수 있습니다. 파트너가 AWS 계정이 있다면 더 수월하게 Amazon SNS 정책을 사용할 수 있습니다. 그러나 파트너 회사에서 AWS 보안 자격 증명을 가진 누구라도 주제에 메시지를 게시할 수 있습니다. 여기서는 특정 인물(또는 애플리케이션)에게만 액세스 권한을 부여한다고 가정합니다. 이렇게 하려면 파트너를 자체 회사 내의 사용자와 같이 처리해야 하며, IAM 정책 대신 Amazon SNS 정책을 사용해야 합니다.

이 예시에서는 파트너 회사를 대표하는 WidgetCo 라는 그룹을 만들고, 액세스 권한이 필요한 파트너 회사의 특정 사용자 (또는 애플리케이션) 를 위한 사용자를 만든 다음, 사용자를 그룹에 추가합니다.

그런 다음 이름이 지정된 특정 주제에 대한 `Publish` 액세스 권한을 그룹에 부여하는 정책을 WidgetPartnerTopic첨부합니다.

또한 WidgetCo 그룹이 주제와 관련된 다른 활동을 하지 못하도록 하기 위해 다른 주제를 제외한 모든 Amazon SNS 활동에 `Publish` 대해 권한을 거부하는 설명을 추가합니다. WidgetPartnerTopic 이는 시스템의 어딘가에서 사용자에게 Amazon SNS에 대한 폭넓은 액세스 권한을 부여하는 광범위한 정책이 있는 경우에만 필요합니다.

```
{
  "Statement": [{
      "Effect": "Allow",
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:*:123456789012:WidgetPartnerTopic"
    },
```

```
    {
      "Effect": "Deny",
      "NotAction": "sns:Publish",
      "NotResource": "arn:aws:sns:*:123456789012:WidgetPartnerTopic"
    }
  ]
}
```

Amazon SNS에 대한 임시 보안 자격 증명 사용

IAM에서는 보안 자격 증명이 있는 IAM 사용자를 만들 뿐 아니라 임의의 사용자에게 임시 보안 자격 증명을 부여하여 이 사용자가 AWS 서비스와 리소스에 액세스하는 것을 허용할 수 있습니다. AWS 계정이 있는 사용자를 관리할 수 있습니다. 이 사용자는 IAM 사용자입니다. 또한 시스템에서 AWS 계정이 없는 사용자도 관리할 수 있습니다. 이 사용자는 페더레이션 사용자라고 합니다. AWS 리소스에 액세스하도록 생성한 애플리케이션도 "사용자"가 될 수 있습니다.

Amazon SNS에 요청할 때 이 임시 보안 자격 증명을 사용할 수 있습니다. API 라이브러리는 요청 인증 시 이 자격 증명을 사용하여 필요한 서명 값을 계산합니다. 만료된 자격 증명으로 요청을 보내면 Amazon SNS는 요청을 거부합니다.

임시 보안 자격 증명에 대한 IAM 지원에 대한 자세한 내용을 확인하려면 IAM 사용의 AWS 리소스에 대한 임시 액세스 권한 부여로 이동하세요.

Example Amazon SNS 요청 인증에 임시 보안 자격 증명 사용

다음 예는 Amazon SNS 요청을 인증하기 위해 임시 보안 자격 증명을 얻는 방법을 보여줍니다.

```
http://sns.us-east-2.amazonaws.com/
?Name=My-Topic
&Action=CreateTopic
&Signature=gfzIF53exFVdpSNb8AiwN3Lv%2FNYXh6S%2Br3yySK70oX4%3D
&SignatureVersion=2
&SignatureMethod=HmacSHA256
&Timestamp=2010-03-31T12%3A00%3A00.000Z
&SecurityToken=SecurityTokenValue
&AWSAccessKeyId=Access Key ID provided by AWS Security Token Service
```

Amazon SNS API 권한: 작업 및 리소스 참조

다음 목록은 액세스 제어의 Amazon SNS 구현에 대한 정보를 제공합니다.

- 각각의 정책은 하나의 주제에만 적용(규제 작성 시 다른 주제에 적용하는 문구를 포함하지 마세요)해야 합니다.
- 각각의 정책에는 고유한 규제가 있어야 합니다. `Id`
- 정책의 각 문구에는 고유한 문구가 있어야 합니다. `sid`

정책 할당량

다음 표에는 정책 정보에 대한 최대 할당량이 나열되어 있습니다.

명칭	최대 할당량
바이트	30kb
Statement	100
Principal	1~200(0은 유효하지 않음)
Resource	1(0은 유효하지 않습니다. 값은 정책 주제의 ARN과 일치해야 합니다)

유효한 Amazon SNS 정책 작업

Amazon SNS는 다음의 표에 표시된 작업을 지원합니다.

작업	설명
sns: AddPermission	주제 정책에 권한을 추가할 권한을 부여합니다.
sns: DeleteTopic	주제를 삭제할 권한을 부여합니다.
sns: GetDataProtectionPolicy	주제의 데이터 보호 정책을 검색할 수 있는 권한을 부여합니다.
sns: GetTopicAttributes	모든 주제 속성을 수신할 권한을 부여합니다.
sns: ListSubscriptionsByTopic	특정 주제에 대한 모든 구독을 검색할 권한을 부여합니다.
sns: ListTagsForResource	지정된 주제에 추가된 모든 태그를 나열할 수 있는 권한을 부여합니다.

작업	설명
sns:Publish	주제 또는 엔드포인트에 게시 및 배치 게시 모두에 대한 권한을 부여합니다. 자세한 내용은 게시 및 PublishBatchAmazon 단순 알림 서비스 API 참조를 참조하십시오.
sns: PutDataProtectionPolicy	주제의 데이터 보호 정책을 설정할 수 있는 권한을 부여합니다.
sns: RemovePermission	주제 정책에서 모든 권한을 제거할 권한을 부여합니다.
sns: SetTopicAttributes	주제의 속성을 설정할 권한을 부여합니다.
sns:Subscribe	주제를 구독할 권한을 부여합니다.

서비스별 키

Amazon SNS는 다음과 같은 서비스별 키를 사용합니다. `Subscribe` 요청에 대한 액세스를 제한하는 정책에서 이러한 키를 사용할 수 있습니다.

- sns:endpoint— `Subscribe` 요청 또는 이전에 확인된 구독의 URL, 이메일 주소 또는 ARN입니다. 특정 엔드포인트(예: *@example.com)에 대한 액세스를 제한하려면 문자열 조건과 함께 사용합니다(Amazon SNS에 대한 정책 예제 참조).
- sns:protocol— `Subscribe` 요청 또는 이전에 확인한 구독의 `protocol` 값입니다. 특정 전송 프로토콜(예: https)에 대한 게시를 제한하려면 문자열 조건과 함께 사용합니다(Amazon SNS에 대한 정책 예제 참조).

> **Important**
>
> sns:Endpoint로 액세스를 규제하는 정책 사용 시 DNS 문제가 향후 엔드포인트 이름 확인에 영향을 미칠 수 있습니다.

Amazon Simple Notification Service ID 및 액세스 문제 해결

다음 정보를 사용하여 Amazon SNS 및 IAM으로 작업할 때 발생할 수 있는 일반적인 문제를 진단하고 수정할 수 있습니다.

주제

- Amazon SNS에서 작업을 수행할 권한이 없음
- 저는 iam을 수행할 권한이 없습니다. PassRole
- 내 AWS 계정 외부의 사람이 내 Amazon SNS 리소스에 액세스할 수 있게 허용하고 싶음

Amazon SNS에서 작업을 수행할 권한이 없음

작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 작업을 수행할 수 있도록 정책을 업데이트해야 합니다.

다음 예제 오류는 `mateojackson` 사용자가 콘솔을 사용하여 가상 *`my-example-widget`* 리소스에 대한 세부 정보를 보려고 하지만 가상 `sns:`*`GetWidget`* 권한이 없을 때 발생합니다.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 sns:GetWidget on resource: my-example-widget
```

이 경우 Mateo의 정책은 `sns:`*`GetWidget`* 작업을 사용하여 *`my-example-widget`* 리소스에 액세스하도록 허용하도록 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하십시오. 관리자는 로그인 보안 인증을 제공한 사용자입니다.

저는 iam을 수행할 권한이 없습니다. PassRole

`iam:PassRole` 작업을 수행할 수 있는 권한이 없다는 오류가 수신되면 Amazon SNS에 역할을 전달할 수 있도록 정책을 업데이트해야 합니다.

일부 AWS 서비스에서는 새 서비스 역할 또는 서비스 연결 역할을 생성하는 대신 해당 서비스에 기존 역할을 전달할 수 있습니다. 이렇게 하려면 사용자가 서비스에 역할을 전달할 수 있는 권한을 가지고 있어야 합니다.

다음 예제 오류는 `marymajor`라는 IAM 사용자가 콘솔을 사용하여 Amazon SNS에서 태스크를 수행하려고 하는 경우에 발생합니다. 하지만 작업을 수행하려면 서비스 역할이 부여한 권한이 서비스에 있어야 합니다. Mary는 서비스에 역할을 전달할 수 있는 권한을 가지고 있지 않습니다.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

이 경우 Mary가 `iam:PassRole`작업을 수행할 수 있도록 Mary의 정책을 업데이트해야 합니다.

도움이 필요한 경우 AWS 관리자에게 문의하세요. 관리자는 로그인 보안 인증을 제공한 사용자입니다.

내 AWS 계정 외부의 사람이 내 Amazon SNS 리소스에 액세스할 수 있게 허용하고 싶음

다른 계정의 사용자 또는 조직 외부의 사람이 리소스에 액세스할 때 사용할 수 있는 역할을 생성할 수 있습니다. 역할을 수임할 신뢰할 수 있는 사람을 지정할 수 있습니다. 리소스 기반 정책 또는 액세스 제어 목록(ACL)을 지원하는 서비스의 경우 이러한 정책을 사용하여 다른 사람에게 리소스에 대한 액세스 권한을 부여할 수 있습니다.

자세한 내용은 다음을 참조하십시오.

- Amazon SNS에서 이러한 기능을 지원하는지 여부를 알아보려면 Amazon SNS가 IAM으로 작동하는 방식 단원을 참조하세요.
- 소유하고 있는 AWS 계정의 리소스에 대한 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 자신이 소유한 다른 AWS 계정의 IAM 사용자에 대한 액세스 권한 제공을 참조하십시오.
- 리소스에 대한 액세스 권한을 서드 파티 AWS 계정에게 제공하는 방법을 알아보려면 IAM 사용 설명서의 서드 파티가 소유한 AWS 계정에 대한 액세스 제공을 참조하세요.
- 자격 증명 페더레이션을 통해 액세스 권한을 제공하는 방법을 알아보려면 IAM 사용 설명서의 외부에서 인증된 사용자에게 액세스 권한 제공(자격 증명 페더레이션)을 참조하세요.
- 크로스 계정 액세스를 위한 역할과 리소스 기반 정책 사용의 차이점을 알아보려면 IAM 사용 설명서의 IAM 역할과 리소스 기반 정책의 차이를 참조하십시오.

Amazon SNS의 로깅 및 모니터링

이 단원에서는 Amazon SNS 주제 로깅 및 모니터링에 대해 설명합니다.

주제

- CloudTrail을 사용하여 Amazon SNS API 호출 로깅
- CloudWatch를 사용하여 Amazon SNS 주제 모니터링

CloudTrail을 사용하여 Amazon SNS API 호출 로깅

Amazon SNS는 Amazon SNS에서 사용자, 역할 또는 AWS 서비스가 수행한 작업에 대한 레코드를 제공하는 서비스인 AWS CloudTrail과 통합됩니다. CloudTrail은 Amazon SNS에 대한 API 호출을 이벤트로 캡처합니다. 캡처되는 호출에는 Amazon SNS 콘솔로부터의 호출과 Amazon SNS API 작업에 대한 코드 호출이 포함됩니다. 추적을 생성하면 Amazon SNS 이벤트를 포함한 CloudTrail 이벤트를 지속

적으로 Amazon S3 버킷에 배포할 수 있습니다. 추적을 구성하지 않은 경우에도 CloudTrail 콘솔의 이벤트 기록에서 최신 이벤트를 볼 수 있습니다. CloudTrail에서 수집한 정보를 사용하여 Amazon SNS에 수행된 요청, 요청이 수행된 IP 주소, 요청을 수행한 사람, 요청이 수행된 시간 및 추가 세부 정보를 확인할 수 있습니다.

구성 및 사용 방법을 포함하여 CloudTrail에 대한 자세한 내용은 AWS CloudTrail 사용 설명서를 참조하세요.

CloudTrail의 Amazon SNS 정보

CloudTrail은 계정 생성 시 AWS 계정에서 사용되도록 설정됩니다. 지원되는 이벤트 활동이 Amazon SNS에서 발생하면, 해당 활동이 이벤트 기록의 다른 AWS 서비스 이벤트와 함께 CloudTrail 이벤트에 기록됩니다. AWS 계정에서 최신 이벤트를 확인, 검색 및 다운로드할 수 있습니다. 자세한 정보는 CloudTrail 이벤트 기록을 사용하여 이벤트 보기를 참조하세요.

Amazon SNS의 이벤트를 포함하여 AWS 계정에 이벤트를 지속적으로 기록하려면 추적을 생성합니다. CloudTrail은 추적을 사용하여 Amazon S3 버킷으로 로그 파일을 전송할 수 있습니다. 콘솔에서 추적을 생성하면 기본적으로 모든 AWS 리전에 추적이 적용됩니다. 추적은 AWS 파티션에 있는 모든 리전의 이벤트를 로깅하고 지정된 Amazon S3 버킷으로 로그 파일을 전송합니다. 또는 CloudTrail 로그에서 수집된 이벤트 데이터를 추가 분석 및 처리하도록 다른 AWS 서비스를 구성할 수 있습니다. 자세한 내용은 다음 자료를 참조하세요.

- 추적 생성 개요
- CloudTrail 지원 서비스 및 통합
- CloudTrail에 대한 Amazon SNS 알림 구성
- 여러 리전에서 CloudTrail 로그 파일 받기 및 여러 계정에서 CloudTrail 로그 파일 받기

CloudTrail의 컨트롤 플레인 이벤트

Amazon SNS는 CloudTrail 로그 파일에 다음 작업을 이벤트로 로그합니다.

- `AddPermission`
- `CheckIfPhoneNumberIsOptedOut`
- `ConfirmSubscription`
- `CreatePlatformApplication`
- `CreatePlatformEndpoint`
- `CreateSMSSandboxPhoneNumber`

- CreateTopic
- DeleteEndpoint
- DeletePlatformApplication
- DeleteSMSSandboxPhoneNumber
- DeleteTopic
- GetDataProtectionPolicy
- GetEndpointAttributes
- GetPlatformApplicationAttributes
- GetSMSAttributes
- GetSMSSandboxAccountStatus
- GetSubscriptionAttributes
- GetTopicAttributes
- ListEndpointsByPlatformApplication
- ListOriginationNumbers
- ListPhoneNumbersOptedOut
- ListPlatformApplications
- ListSMSSandboxPhoneNumbers
- ListSubscriptions
- ListSubscriptionsByTopic
- ListTagsForResource
- ListTopics
- OptInPhoneNumber
- PutDataProtectionPolicy
- RemovePermission
- SetEndpointAttributes
- SetPlatformApplicationAttributes
- SetSMSAttributes
- SetSubscriptionAttributes
- SetTopicAttributes
- Subscribe

- TagResource
- Unsubscribe
- UntagResource
- VerifySMSSandboxPhoneNumber

> **Note**
>
> Amazon Web Services(인증되지 않은 모드)에 로그인하지 않은 상태에서 ConfirmSubscription 또는 Unsubscribe 작업이 호출되면 해당 작업은 CloudTrail에 로그되지 않습니다. 따라서 주제에 대해 대기 중인 구독을 확인하기 위해 이메일 알림에 포함된 링크를 선택하면 `ConfirmSubscription` 작업이 비인증 모드에서 호출됩니다. 이 예에서는 `ConfirmSubscription` 작업이 CloudTrail에 로그되지 않습니다.

모든 이벤트 및 로그 항목에는 요청을 생성한 사용자에 대한 정보가 들어 있습니다. 자격 증명 정보를 이용하면 다음을 쉽게 판단할 수 있습니다.

- 요청을 루트로 했는지 아니면 AWS Identity and Access Management(IAM) 사용자 자격 증명으로 했는지.
- 역할 또는 페더레이션 사용자에 대한 임시 보안 자격 증명을 사용하여 요청이 생성되었는지 여부.
- 다른 AWS 서비스에서 요청했는지 여부.

자세한 내용은 CloudTrail userIdentity 요소를 참조하세요.

CloudTrail의 데이터 영역 이벤트

CloudTrail 파일에서 다음 API 작업의 로깅을 활성화하려면 CloudTrail에서 데이터 영역 API 활동의 로깅을 활성화해야 합니다. 자세한 내용을 알아보려면 AWS CloudTrail 사용 설명서의 데이터 이벤트 로깅을 참조하세요.

또한 데이터 영역 이벤트를 리소스 유형별로 필터링하여 CloudTrail에서 선택적으로 로깅하고 비용을 지불하려는 Amazon SNS API 호출을 세부적으로 제어할 수 있습니다. 예를 들어 리소스 유형으로 `AWS::SNS::Topic`을 지정하면 주제에 대한 `Publish` 및 `PublishBatch` API 작업의 호출을 로깅할 수 있습니다. 마찬가지로, 리소스 유형으로 `AWS::SNS::PlatformEndpoint`를 지정하면 플랫폼 엔드포인트에 대한 Publish API 작업의 호출을 로깅할 수 있습니다. 자세한 내용은 AWS CloudTrail API 참조의 `AdvancedEventSelector` 섹션을 참조하세요.

> **Note**
>
> Amazon SNS 리소스 유형 `AWS::SNS::PhoneNumber`는 CloudTrail에 의해 로깅되지 않습니다.

Amazon SNS 데이터 영역 API

- `Publish`
- `PublishBatch`

예: Amazon SNS 로그 파일 항목

추적이란 지정한 Amazon S3 버킷에 이벤트를 로그 파일로 입력할 수 있게 하는 구성입니다. CloudTrail 로그 파일에는 하나 이상의 로그 항목이 포함될 수 있습니다. 이벤트는 모든 소스의 단일 요청을 나타내며 요청된 작업, 작업 날짜와 시간, 요청 파라미터 등에 대한 정보를 포함합니다. CloudTrail 로그 파일은 퍼블릭 API 호출의 주문 스택 트레이스가 아니므로 특정 순서로 표시되지 않습니다.

다음은 `ListTopics`, `CreateTopic` 및 `DeleteTopic` 작업을 보여 주는 CloudTrail 로그 항목이 나타낸 예입니다.

```
{
  "Records": [
    {
      "eventVersion": "1.02",
      "userIdentity": {
       "type":"IAMUser",
       "userName":"Bob"
       "principalId": "EX_PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/Bob",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
      },
      "eventTime": "2014-09-30T00:00:00Z",
      "eventSource": "sns.amazonaws.com",
      "eventName": "ListTopics",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "127.0.0.1",
      "userAgent": "aws-sdk-java/unknown-version",
      "requestParameters": {
        "nextToken": "ABCDEF1234567890EXAMPLE=="
```

```
    },
    "responseElements": null,
    "requestID": "example1-b9bb-50fa-abdb-80f274981d60",
    "eventID": "example0-09a3-47d6-a810-c5f9fd2534fe",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
  },
  {
    "eventVersion": "1.02",
    "userIdentity": {
     "type":"IAMUser",
     "userName":"Bob"
     "principalId": "EX_PRINCIPAL_ID",
      "arn": "arn:aws:iam::123456789012:user/Bob",
      "accountId": "123456789012",
      "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
    },
    "eventTime": "2014-09-30T00:00:00Z",
    "eventSource": "sns.amazonaws.com",
    "eventName": "CreateTopic",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "aws-sdk-java/unknown-version",
    "requestParameters": {
      "name": "hello"
    },
    "responseElements": {
      "topicArn": "arn:aws:sns:us-west-2:123456789012:hello-topic"
    },
    "requestID": "example7-5cd3-5323-8a00-f1889011fee9",
    "eventID": "examplec-4f2f-4625-8378-130ac89660b1",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
  },
  {
    "eventVersion": "1.02",
    "userIdentity": {
     "type":"IAMUser",
     "userName":"Bob"
     "principalId": "EX_PRINCIPAL_ID",
      "arn": "arn:aws:iam::123456789012:user/Bob",
      "accountId": "123456789012",
      "accessKeyId": "AKIAIOSFODNN7EXAMPLE"
    },
```

```
      "eventTime": "2014-09-30T00:00:00Z",
      "eventSource": "sns.amazonaws.com",
      "eventName": "DeleteTopic",
      "awsRegion": "us-west-2",
      "sourceIPAddress": "127.0.0.1",
      "userAgent": "aws-sdk-java/unknown-version",
      "requestParameters": {
        "topicArn": "arn:aws:sns:us-west-2:123456789012:hello-topic"
      },
      "responseElements": null,
      "requestID": "example5-4faa-51d5-aab2-803a8294388d",
      "eventID": "example8-6443-4b4d-abfd-1b867280d964",
      "eventType": "AwsApiCall",
      "recipientAccountId": "123456789012"
    },
  ]
}
```

다음 예제에는 Publish 및 PublishBatch 작업을 보여주는 CloudTrail 로그 항목이 나와 있습니다.

Publish

```
{
 "eventVersion": "1.09",
 "userIdentity": {
  "type": "AssumedRole",
  "principalId": "EX_PRINCIPAL_ID",
  "arn": "arn:aws:iam::123456789012:user/Bob",
  "accountId": "123456789012",
  "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
  "sessionContext": {
   "sessionIssuer": {
    "type": "Role",
    "principalId": "AKIAIOSFODNN7EXAMPLE",
    "arn": "arn:aws:iam::123456789012:role/Admin",
    "accountId": "123456789012",
    "userName": "ExampleUser"
   },
   "attributes": {
    "creationDate": "2023-08-21T16:44:05Z",
    "mfaAuthenticated": "false"
   }
  }
 },
```

```
 "eventTime": "2023-08-21T16:48:37Z",
 "eventSource": "sns.amazonaws.com",
 "eventName": "Publish",
 "awsRegion": "us-east-1",
 "sourceIPAddress": "192.0.2.0",
 "userAgent": "aws-cli/1.29.16 md/Botocore#1.31.16 ua/2.0 os/
linux#5.4.250-173.369.amzn2int.x86_64 md/arch#x86_64 lang/python#3.8.17 md/
pyimpl#CPython cfg/retry-mode#legacy botocore/1.31.16",
 "requestParameters": {
  "topicArn": "arn:aws:sns:us-east-1:123456789012:ExampleSNSTopic",
  "message": "HIDDEN_DUE_TO_SECURITY_REASONS",
  "subject": "HIDDEN_DUE_TO_SECURITY_REASONS",
  "messageStructure": "json",
  "messageAttributes": "HIDDEN_DUE_TO_SECURITY_REASONS"
 },
 "responseElements": {
  "messageId": "0787cd1e-d92b-521c-a8b4-90434e8ef840"
 },
 "requestID": "0a8ab208-11bf-5e01-bd2d-ef55861b545d",
 "eventID": "bb3496d4-5252-4660-9c28-3c6aebdb21c0",
 "readOnly": false,
 "resources": [{
  "accountId": "123456789012",
  "type": "AWS::SNS::Topic",
  "ARN": "arn:aws:sns:us-east-1:123456789012:ExampleSNSTopic"
 }],
 "eventType": "AwsApiCall",
 "managementEvent": false,
 "recipientAccountId": "123456789012",
 "eventCategory": "Data",
 "tlsDetails": {
  "tlsVersion": "TLSv1.2",
  "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
  "clientProvidedHostHeader": "sns.us-east-1.amazonaws.com"
 }
}
```

PublishBatch

```
{
 "eventVersion": "1.09",
 "userIdentity": {
  "type": "AssumedRole",
```

```
   "principalId": "EX_PRINCIPAL_ID",
   "arn": "arn:aws:iam::123456789012:user/Bob",
   "accountId": "123456789012",
   "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
   "sessionContext": {
    "sessionIssuer": {
     "type": "Role",
     "principalId": "AKIAIOSFODNN7EXAMPLE",
     "arn": "arn:aws:iam::123456789012:role/Admin",
     "accountId": "123456789012",
     "userName": "ExampleUser"
    },
    "attributes": {
     "creationDate": "2023-08-21T19:20:49Z",
     "mfaAuthenticated": "false"
    }
   }
  },
  "eventTime": "2023-08-21T19:22:01Z",
  "eventSource": "sns.amazonaws.com",
  "eventName": "PublishBatch",
  "awsRegion": "us-east-1",
  "sourceIPAddress": "192.0.2.0",
  "userAgent": "aws-cli/1.29.16 md/Botocore#1.31.16 ua/2.0 os/
linux#5.4.250-173.369.amzn2int.x86_64 md/arch#x86_64 lang/python#3.8.17 md/
pyimpl#CPython cfg/retry-mode#legacy botocore/1.31.16",
  "requestParameters": {
   "topicArn": "arn:aws:sns:us-east-1:123456789012:ExampleSNSTopic",
   "publishBatchRequestEntries": [{
     "id": "1",
     "message": "HIDDEN_DUE_TO_SECURITY_REASONS"
    },
    {
     "id": "2",
     "message": "HIDDEN_DUE_TO_SECURITY_REASONS"
    }
   ]
  },
  "responseElements": {
   "successful": [{
     "id": "1",
     "messageId": "30d68101-a64a-5573-9e10-dc5c1dd3af2f"
    },
    {
```

```
    "id": "2",
    "messageId": "c0aa0c5c-561d-5455-b6c4-5101ed84de09"
   }
  ],
  "failed": []
 },
 "requestID": "e2cdf7f3-1b35-58ad-ac9e-aaaea0ace2f1",
 "eventID": "10da9a14-0154-4ab6-b3a5-1825b229a7ed",
 "readOnly": false,
 "resources": [{
  "accountId": "123456789012",
  "type": "AWS::SNS::Topic",
  "ARN": "arn:aws:sns:us-east-1:123456789012:ExampleSNSTopic"
 }],
 "eventType": "AwsApiCall",
 "managementEvent": false,
 "recipientAccountId": "123456789012",
 "eventCategory": "Data",
 "tlsDetails": {
  "tlsVersion": "TLSv1.2",
  "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
  "clientProvidedHostHeader": "sns.us-east-1.amazonaws.com"
 }
}
```


CloudWatch를 사용하여 Amazon SNS 주제 모니터링

Amazon SNS와 Amazon CloudWatch가 통합되어 모든 활성 Amazon SNS 알림에 대한 지표를 수집, 확인 및 분석할 수 있습니다. Amazon SNS용 CloudWatch를 구성하고 나면 Amazon SNS 주제, 푸시 알림 및 SMS 전송의 성능에 대한 더 나은 인사이트를 얻을 수 있습니다. 예를 들어, `NumberOfNotificationsFailed`와 같은 Amazon SNS 지표에 대해 지정된 임계값에 도달하면 이메일 알림을 전송하도록 경보를 설정할 수 있습니다. Amazon SNS가 CloudWatch에 전송하는 모든 지표 목록은 Amazon SNS 지표에서 확인하세요. Amazon SNS 푸시 알림에 대한 자세한 정보는 모바일 푸시 알림에서 확인하세요.

> **Note**
>
> Amazon SNS 주제에 대해 CloudWatch로 구성한 지표는 자동으로 수집되어 1분 간격으로 CloudWatch에 푸시됩니다. 이러한 지표는 활성화에 대한 CloudWatch 지침을 충족하는 모든 주제에서 수집됩니다. 주제는 해당 주제에 대한 마지막 활동(즉, API 호출)부터 최대 여섯 시간 동안 CloudWatch에서 활성으로 간주됩니다.

CloudWatch에 보고된 Amazon SNS 지표에는 요금이 부과되지 않으며 Amazon SNS 서비스의 일부로 제공됩니다.

Amazon SNS에 대한 CloudWatch 지표 보기

CloudWatch 콘솔, CloudWatch의 자체 명령줄 인터페이스(CLI)를 사용하거나 프로그래밍 방식으로 CloudWatch API를 사용하여 Amazon SNS에 대한 지표를 모니터링할 수 있습니다. 다음 절차에서는 AWS Management Console을 사용하여 지표를 액세스하는 방법을 보여줍니다.

CloudWatch 콘솔을 사용하여 지표 보기

1. CloudWatch 콘솔에 로그인합니다.
2. 탐색 창에서 [Metrics]를 선택합니다.
3. 모든 지표 탭에서 SNS를 선택하고 다음 차원 중 하나를 선택합니다.
   - 국가, SMS 유형
   - 전화번호
   - 주제 지표
   - 차원을 포함하지 않은 지표
4. 세부 정보를 보려면 특정 항목을 선택합니다. 예를 들어 주제 지표를 선택한 다음, NumberOfMessagesPublished를 선택하면 6시간이라는 시간 범위 동안 1분 간 게시된 Amazon SNS 메시지의 평균 수가 표시됩니다.
5. Amazon SNS 사용 지표를 보려면 All metrics(모든 지표) 탭에서 Usage(사용량)를 선택하고 target Amazon SNS usage metric(대상 Amazon SNS 사용 지표)(예: `NumberOfMessagesPublishedPerAccount`)을 선택합니다.

Amazon SNS 지표에 대한 CloudWatch 경보 설정

CloudWatch에서는 지표에 대한 임계값에 도달한 경우에도 경보를 설정할 수 있습니다. 예를 들어, 메트릭 NumberOfNotificationsFailed에 대해 경보를 설정할 수 있어 샘플링 기간 내에 지정한 임계값에 도달할 경우 이메일 알림이 전송되어 이를 알립니다.

CloudWatch 콘솔을 사용한 경보 설정

1. AWS Management Console에 로그인하고 https://console.aws.amazon.com/cloudwatch/에서 CloudWatch 콘솔을 엽니다.

2. 경보를 선택한 다음 경보 생성 버튼을 선택합니다. 이로써 Create Alarm 마법사를 시작하게 됩니다.
3. Amazon SNS 지표를 스크롤하여 경보를 설정할 지표를 찾습니다. 경보를 설정할 지표를 선택하고 Continue(계속)를 선택합니다.
4. 지표의 Name(이름), Description(설명), Threshold(임계값), Time(시간)을 입력한 다음 Continue(계속)를 선택합니다.
5. 경보 상태대로 경보를 선택합니다. 경보 상태에 도달했을 때 CloudWatch에서 이메일을 보내도록 하려면 기존 Amazon SNS 주제를 선택하거나 새 이메일 주제 생성을 선택합니다. 새 이메일 주제 생성을 선택하면 새로운 주제의 이름과 이메일 주소를 설정할 수 있습니다. 이 목록은 저장되어 향후 경보의 드롭다운 상자에 표시됩니다. 계속을 선택합니다.

   > **Note**
   >
   > 새 이메일 주제 생성을 사용하여 새 Amazon SNS 주제를 생성하는 경우 알림을 수신하기 전에 이메일 주소를 확인해야 합니다. 이메일은 경보가 경보 상태에 입력될 때만 전송됩니다. 이러한 경보 상태 변경이 이메일이 검증되기 전에 발생할 경우에는 알림을 받지 못합니다.

6. 이 시점에서 Create Alarm 마법사는 사용자가 생성할 경보를 검토할 기회를 줍니다. 수정이 필요한 경우 오른쪽의 Edit 링크를 사용하면 됩니다. 만족스러우면 Create Alarm(경보 생성)을 선택합니다.

CloudWatch 및 경보 사용에 대한 자세한 정보는 CloudWatch 설명서를 참조하세요.

Amazon SNS 지표

Amazon SNS는 CloudWatch에 다음 지표를 전송합니다.

네임스페이스	지표	설명
AWS/SNS	`NumberOfMessagesPublished`	Amazon SNS 주제에 게시된 메시지의 수 Units: Count 유효한 차원: Application, PhoneNumber, Platform 및 TopicName

네임스페이스	지표	설명
		유효한 통계: Sum
AWS/SNS	NumberOfNotificationsDelivered	Amazon SNS 주제에서 구독 엔드포인트로 성공적으로 전송된 메시지의 수. 전송 시도가 성공하려면 엔드포인트의 구독이 메시지를 수락해야 합니다. 다음 경우에 구독은 메시지를 수락합니다. a.) 필터 정책이 없음, 또는 b.) 필터 정책에 메시지에 할당된 속성과 일치하는 속성이 포함됨 구독이 메시지를 거부할 경우 해당 전송 시도는 이 지표에 포함되지 않습니다. Units: Count 유효한 차원: Application, PhoneNumber, Platform 및 TopicName 유효한 통계: Sum

네임스페이스	지표	설명
AWS/SNS	`NumberOfNotificationsFailed`	Amazon SNS에서 전송에 실패한 메시지 수 Amazon SQS, 이메일, SMS 또는 모바일 푸시 엔드포인트의 경우 이 지표는 Amazon SNS가 메시지 전송 시도를 중지하면 1씩 증가합니다. HTTP 또는 HTTPS 엔드포인트의 경우, 이 지표는 최초 시도 후 이어지는 재시도를 포함하여 실패한 전송 시도를 모두 포함합니다. 기타 모든 엔드포인트의 경우, 메시지 전송이 실패하면 시도 횟수에 상관없이 수가 1씩 올라갑니다. 이 지표는 구독 필터 정책에 의해 거부된 메시지를 포함하지 않습니다. HTTP 엔드포인트의 재시도 수를 제어할 수 있습니다. 자세한 정보는 Amazon SNS 메시지 전송 재시도에서 확인하세요. Units: Count 유효한 차원: Application, PhoneNumber, Platform 및 TopicName 유효한 통계: Sum, Average

네임스페이스	지표	설명
AWS/SNS	NumberOfNotificationsFilteredOut	구독 필터 정책에 의해 거부된 메시지의 수. 메시지 속성이 정책 속성과 일치하지 않으면 필터 정책이 메시지를 거부합니다. Units: Count 유효한 차원: Application, PhoneNumber, Platform 및 TopicName 유효한 통계: Sum, Average
AWS/SNS	NumberOfNotificationsFilteredOut-MessageAttributes	속성 기반 필터링에 대한 구독 필터 정책에 의해 거부된 메시지의 수. 단위: CountValid 차원: Application, PhoneNumber, Platform 및 TopicName 유효한 통계: Sum, Average
AWS/SNS	NumberOfNotificationsFilteredOut-MessageBody	페이로드 기반 필터링에 대한 구독 필터 정책에 의해 거부된 메시지의 수. 단위: 개수 유효한 차원: Application, PhoneNumber, Platform 및 TopicName 유효한 통계: Sum, Average

네임스페이스	지표	설명
AWS/SNS	NumberOfNotificationsFilteredOut-InvalidAttributes	메시지의 속성이 잘못되어(예: 잘못된 형식의 속성 JSON) 구독 필터 정책에 의해 거부된 메시지의 수 Units: Count 유효한 차원: Application, PhoneNumber, Platform 및 TopicName 유효한 통계: Sum, Average
AWS/SNS	NumberOfNotificationsFilteredOut-NoMessageAttributes	메시지에 속성이 없어 구독 필터 정책에 의해 거부된 메시지의 수. Units: Count 유효한 차원: Application, PhoneNumber, Platform 및 TopicName 유효한 통계: Sum, Average
AWS/SNS	NumberOfNotificationsFilteredOut-InvalidMessageBody	메시지의 본문이 필터링에 대해 유효하지 않아(예: 잘못된 JSON 메시지 본문) 구독 필터 정책에 의해 거부된 메시지의 수. 단위: 개수 유효한 차원: Application, PhoneNumber, Platform 및 TopicName 유효한 통계: Sum, Average

네임스페이스	지표	설명
AWS/SNS	NumberOfNotificationsRedrivenToDlq	배달 못한 편지 대기열로 이동한 메시지 수입니다. Units: Count 유효한 차원: Application, PhoneNumber, Platform 및 TopicName 유효한 통계: Sum, Average
AWS/SNS	NumberOfNotificationsFailedToRedriveToDlq	배달 못한 편지 대기열로 이동할 수 없는 메시지 수입니다. Units: Count 유효한 차원: Application, PhoneNumber, Platform 및 TopicName 유효한 통계: Sum, Average
AWS/SNS	PublishSize	게시된 메시지의 크기. Units: Bytes 유효한 차원: Application, PhoneNumber, Platform 및 TopicName Valid Statistics: Minimum, Maximum, Average, Count

네임스페이스	지표	설명
AWS/SNS	SMSMonthToDateSpentUSD	당월 초부터 사용한 SMS 메시지 전송에 대한 요금입니다. 이 지표에 대한 경보를 설정하여 당월 누적 요금이 사용자 계정의 SMS 월 지출 할당량에 도달했는지 확인할 수 있습니다. SMS 메시지를 전송하면 이러한 할당량을 초과하는 비용이 발생할 것을 Amazon SNS에서 확인하면 몇 분 이내에 SMS 메시지 게시를 중지합니다. 사용자의 월별 SMS 지출 할당량 설정에 대한 자세한 내용 또는 AWS에서 지출 할당량 증가 요청에 대한 자세한 내용은 SMS 메시징 기본 설정 지정에서 확인하세요. 단위: USD 유효한 차원: PhoneNumber 유효한 통계: Maximum
AWS/SNS	SMSSuccessRate	SMS 메시지 전송 성공 비율입니다. Units: Count 유효한 차원: PhoneNumber 유효한 통계: Sum, Average, Data Samples

Amazon SNS 지표 차원

Amazon Simple Notification Service는 CloudWatch에 다음 차원을 전송합니다.

차원	설명
Application	APN 및 FCM과 같이 지원되는 푸시 알림 서비스 중 하나에 등록된 앱 및 디바이스를 나타내는 애플리케이션 객체를 필터링합니다.
Application,Platform	애플리케이션 및 플랫폼 객체를 필터링합니다. 여기서 플랫폼 객체는 APN 및 FCM과 같은 지원되는 푸시 알림 서비스용입니다.
Country	SMS 메시지의 대상 국가 또는 리전을 필터링합니다. 국가 또는 리전은 ISO 3166-1 alpha-2 코드로 표시됩니다.
PhoneNumber	주제 없이 전화번호에 SMS를 직접 게시할 때 전화번호를 필터링합니다.
Platform	APN 및 FCM과 같은 푸시 알림 서비스용 플랫폼 객체를 필터링합니다.
TopicName	Amazon SNS 주제 이름을 필터링합니다.
SMSType	SMS 메시지의 메시지 유형을 필터링합니다. 프로모션 또는 트랜잭션일 수 있습니다.

Amazon SNS 사용 지표

Amazon Simple Notification Service는 CloudWatch에 다음 사용 지표를 전송합니다.

네임스페이스	서비스	지표	리소스	유형	설명
AWS/사용량	SNS	ResourceCount	NumberOfMessagesPublishedPerAccount	리소스	• AWS 계정 전체에서 Amazon SNS 주제에 게시된 메시지의 수 • 단위: 없음

네임스페이스	서비스	지표	리소스	유형	설명
					• 유효한 통계: Sum
AWS/사용량	SNS	ResourceCount	ApproximateNumberOfTopics	리소스	• AWS 계정 전체의 대략적 주제 수 • 단위: 없음 • 유효한 통계: Average, Minimum, Maximum, Sum
AWS/사용량	SNS	ResourceCount	ApproximateNumberOfFilterPolicies	리소스	• AWS 계정 전체의 대략적 필터 정책 수 • 단위: 없음 • 유효한 통계: Average, Minimum, Maximum, Sum

네임스페이스	서비스	지표	리소스	유형	설명
AWS/사용량	SNS	ResourceCount	ApproximateNumberOfPendingSubscriptions	리소스	• AWS 계정 전체에서 대기 중인 구독의 대략적 수 • 단위: 없음 • 유효한 통계: Average, Minimum, Maximum, Sum

네임스페이스	서비스	지표	리소스	유형	설명
AWS/사용량	SNS	CallCount	• AddPermission • CheckIfPhoneNumberIsOptedOut • CreatePlatformApplication • CreatePlatformEndpoint • ConfirmSubscription • CreateSMSSandboxPhoneNumber • CreateTopic • DeleteEndpoint • DeletePlatformApplication • DeleteSMSSandboxPhoneNumber • DeleteTopic	API	• AWS 계정 전체에서 선택한 Amazon SNS API에 대한 API 호출 수입니다. • 단위: 없음 • 유효한 통계: Sum

네임스페이스	서비스	지표	리소스	유형	설명
			• GetEndpointAttributes • GetPlatformApplicationAttributes • GetSMSAttributes • GetSMSSandboxAccountStatus • GetSubscriptionAttributes • GetTopicAttributes • ListEndpointsByPlatformApplication • ListOriginationNumbers • ListPhoneNumbersOptedOut • ListPlatformApplications		

네임스페이스	서비스	지표	리소스	유형	설명
			• ListSMSSandboxPhoneNumbers • ListSubscriptions • ListSubscriptionsByTopic • ListTagsForResource • ListTopics • OptInPhoneNumber • RemovePermission • SetEndpointAttributes • SetPlatformApplicationAttributes • SetSMSAttributes • SetSubscriptionAttributes • SetTopicAttributes		

네임스페이스	서비스	지표	리소스	유형	설명
			• Subscribe • Unsubscribe • UntagResource • VerifySMSSandboxPhoneNumber		

Amazon SNS에 대한 규정 준수 확인

타사 감사자는 미국 건강 보험 양도 및 책임에 관한 법(HIPAA)을 비롯한 여러 AWS 규정 준수 프로그램의 일환으로 Amazon SNS의 보안 및 규정 준수를 평가합니다.

특정 규정 준수 프로그램의 범위 내에 있는 AWS 서비스 목록은 규정 준수 프로그램 제공 범위 내 AWS 서비스를 참조하세요. 일반적인 정보는 AWS 규정 준수 프로그램을 참조하세요.

AWS Artifact를 사용하여 타사 감사 보고서를 다운로드할 수 있습니다. 자세한 정보는 AWS Artifact에서 보고서 다운로드를 참조하세요.

Amazon SNS 사용 시 규정 준수 책임은 데이터의 민감도, 회사의 규정 준수 목표 및 관련 법률과 규정에 따라 결정됩니다. AWS에서는 규정 준수를 지원할 다음과 같은 리소스를 제공합니다.

- 보안 및 규정 준수 빠른 시작 안내서 – 이 배포 안내서에서는 아키텍처 고려 사항에 관해 설명하고 AWS에서 보안 및 규정 준수에 중점을 둔 기본 환경을 배포하기 위한 단계를 제공합니다.
- HIPAA 보안 및 규정 준수 기술 백서 아키텍팅 - 이 백서는 기업에서 AWS를 사용하여 HIPAA를 준수하는 애플리케이션을 생성하는 방법을 설명합니다.
- AWS 규정 준수 리소스 - 고객 조직이 속한 산업 및 위치에 적용될 수 있는 워크북 및 가이드 콜렉션입니다.
- AWS Config 개발자 안내서의 규칙을 사용하여 리소스 평가 – AWS Config 서비스는 내부 사례, 산업 지침 및 규제에 대한 리소스 구성의 준수 상태를 평가합니다.

- AWS Security Hub - 이 AWS 서비스는 보안 산업 표준 및 모범 사례 규정 준수 여부를 확인하는 데 도움이 되도록 AWS 내 보안 상태를 종합적으로 보여줍니다.

Amazon SNS의 복원성

Amazon SNS의 복원력은 가용 영역을 중심으로 하는 AWS 글로벌 인프라를 활용함으로써 보장됩니다. AWS 리전 AWS 리전 지연 시간이 짧고 처리량이 높으며 중복성이 높은 네트워킹으로 연결된 물리적으로 분리되고 격리된 가용 영역을 제공합니다. 이 아키텍처는 가용 영역 간에 중단 없이 원활한 장애 조치가 가능하므로 애플리케이션과 데이터베이스의 내결함성과 확장성이 기본적으로 기존 데이터 센터 인프라에 비해 내결함성과 확장성이 뛰어납니다. Amazon SNS 구독자는 가용 영역을 사용함으로써 가용성과 안정성이 향상되어 잠재적인 장애에도 불구하고 메시지 전송을 보장받을 수 있습니다. 가용 영역 AWS 리전 및 가용 영역에 대한 자세한 내용은 글로벌 인프라를 참조하십시오AWS .

또한 전송 재시도 및 데드레터 대기열로 Amazon SNS 주제 구독을 구성할 수 있으므로 일시적 장애를 자동으로 처리하고 메시지가 의도한 목적지에 안정적으로 도달하도록 할 수 있습니다.

또한 Amazon SNS는 메시지 필터링 및 메시지 속성을 지원하므로 특정 사용 사례에 맞게 복원력 전략을 조정하여 애플리케이션의 전반적인 안정성을 강화할 수 있습니다.

Amazon SNS의 인프라 보안

관리형 서비스인 Amazon SNS는 보안, ID 및 규정 준수 모범 사례 문서에 있는 AWS 글로벌 네트워크 보안 절차에 의해 보호됩니다.

AWS API 작업을 사용하여 네트워크를 통해 Amazon SNS에 액세스할 수 있습니다. 클라이언트가 전송 계층 보안(TLS) 1.2 이상을 지원해야 합니다. 클라이언트는 Ephemeral Diffie-Hellman(DHE) 또는 Elliptic Curve Ephemeral Diffie-Hellman(ECDHE)과 같은 PFS(전달 완전 보안, Perfect Forward Secrecy)가 포함된 암호 제품군도 지원해야 합니다.

IAM 보안 주체와 연결되어 있는 액세스 키 ID 및 보안 액세스 키를 모두 사용해 요청에 서명해야 합니다. 또는 AWS Security Token Service(AWS STS)를 사용해 임시 보안 자격 증명을 생성하여 요청에 서명할 수 있습니다.

이러한 API 작업은 모든 네트워크 위치에서 호출할 수 있지만, Amazon SNS는 원본 IP 주소를 기반으로 하는 제한을 포함할 수 있는 리소스 기반 액세스 정책을 지원합니다. Amazon SNS 정책을 사용하여 특정 Amazon VPC 엔드포인트 또는 특정 VPC에서 액세스를 제어할 수도 있습니다. 이렇게 하면 특정 Amazon SNS 주제에 대한 네트워크 액세스를 네트워크 내의 AWS 특정 VPC에서만 효과적으로 분리

할 수 있습니다. 자세한 내용은 특정 VPC 엔드포인트에서만 Amazon SNS 주제에 게시할 수 있도록 제한을(를) 참조하세요.

Amazon SNS 보안 모범 사례

AWS는 Amazon SNS에 대해 많은 보안 기능을 제공합니다. 사용자 고유의 보안 정책 맥락에서 이러한 보안 기능을 검토합니다.

> **Note**
>
> 이러한 보안 기능에 대한 지침은 일반적인 사용 사례 및 구현에 적용됩니다. 특정 사용 사례, 아키텍처 및 위협 모델의 맥락에서 이러한 모범 사례를 검토하는 것이 좋습니다.

예방적 모범 사례

다음은 Amazon SNS에 대한 예방 보안 모범 사례입니다.

주제

- 주제에 공개적으로 액세스할 수 없도록 보장
- 최소 권한 액세스 구현
- Amazon SNS 액세스가 필요한 애플리케이션 및 AWS 서비스에 IAM 역할 사용
- 서버 측 암호화 구현
- 전송 중인 데이터의 암호화 강제 시행
- VPC 엔드포인트를 사용한 Amazon SNS 액세스 고려
- 구독이 원시 http 엔드포인트에 전달되도록 구성되지 않았는지 확인합니다.

주제에 공개적으로 액세스할 수 없도록 보장

인터넷에 있는 모든 사람에게 Amazon SNS 주제를 읽거나 쓸 수 있도록 명시적으로 요구하지 않는 한, 주제에 공개적으로 액세스(전 세계의 모든 사람 또는 인증된 AWS 사용자가 액세스 가능)할 수 없도록 해야 합니다.

- `Principal`이 `""`로 설정된 정책을 생성하지 마세요.
- 와일드카드(*)를 사용하지 마세요. 대신에 특정 사용자 또는 사용자의 이름을 지정합니다.

최소 권한 액세스 구현

권한을 부여할 때 권한을 받는 사용자, 권한이 있는 주제 및 이러한 주제를 허용하려는 특정 API 작업을 결정합니다. 최소 권한 원칙을 구현하는 것은 보안 위험을 줄이는 데 중요합니다. 또한 실수나 고의에 의한 부정적인 영향을 줄이는 데 도움이 됩니다.

최소 권한 부여에 대한 표준 보안 권고 사항을 따릅니다. 즉, 특정 작업을 수행하는 데 필요한 권한만 부여합니다. 사용자 액세스와 관련된 보안 정책 조합을 사용하여 최소 권한을 구현할 수 있습니다.

Amazon SNS는 세 가지 유형의 사용자 계정 액세스가 필요한 게시자-구독자 모델을 사용합니다.

- 관리자 – 주제를 생성, 수정 및 삭제할 수 있습니다. 또한 관리자는 주제 정책도 제어합니다.
- 게시자 – 주제에 메시지를 보낼 수 있습니다.
- 구독자 – 주제를 구독할 수 있습니다.

자세한 정보는 다음 섹션을 참조하세요.

- Amazon SNS의 Identity and Access Management
- Amazon SNS API 권한: 작업 및 리소스 참조

Amazon SNS 액세스가 필요한 애플리케이션 및 AWS 서비스에 IAM 역할 사용

Amazon EC2와 같은 애플리케이션 또는 AWS 서비스가 Amazon SNS 주제에 액세스하려면 AWS API 요청에 유효한 AWS 자격 증명을 사용해야 합니다. 이러한 자격 증명은 자동으로 교체되지 않으므로 애플리케이션 또는 EC2 인스턴스에 AWS 자격 증명을 직접 저장해서는 안 됩니다.

IAM 역할을 사용하여 Amazon SNS에 액세스해야 하는 애플리케이션이나 서비스의 임시 자격 증명을 관리해야 합니다. 역할을 사용할 때 AWS Lambda와 같은 EC2 인스턴스 또는 AWS 서비스에 장기 자격 증명(예: 사용자 이름, 암호 및 액세스 키)을 배포할 필요가 없습니다. 그 대신 역할은 애플리케이션에서 다른 AWS 리소스에 호출할 때 사용할 수 있는 임시 권한을 제공합니다.

자세한 정보는 IAM 사용 설명서의 IAM 역할 및 역할에 대한 일반 시나리오: 사용자, 애플리케이션 및 서비스를 참조하세요.

서버 측 암호화 구현

데이터 유출 문제를 완화하려면 메시지를 저장하는 위치와 다른 위치에 저장된 키를 통해 메시지를 암호화하는 유휴 시 데이터 암호화를 사용합니다. 서버 측 암호화(SSE)는 저장된 데이터 암호화 기능을

제공합니다. Amazon SNS는 데이터를 저장할 때 메시지 수준에서 데이터를 암호화하고 사용자가 액세스할 때 메시지를 복호화합니다. SSE는 AWS Key Management Service의 관리형 키를 사용합니다. 요청을 인증하고 액세스 권한이 있는 경우 주제의 암호화 여부와 관계없이 액세스 방식에는 차이가 없습니다.

자세한 정보는 저장 시 암호화 및 키 관리에서 확인하세요.

전송 중인 데이터의 암호화 강제 시행

HTTP를 사용하여 전송 중에 암호화되지 않은 메시지를 게시하는 것은 가능하지만 권장되지는 않습니다. 그러나 암호화된 SNS 주제에 게시할 때는 HTTP를 사용할 수 없습니다.

AWS에서는 HTTP 대신 HTTPS를 사용할 것을 권장합니다. HTTPS를 사용하면 SNS 주제 자체가 암호화되지 않은 경우에도 전송 중에 메시지가 자동으로 암호화됩니다. HTTPS가 없으면 네트워크 기반 공격자가 중간자 공격(MITM)과 같은 공격을 사용하여 네트워크 트래픽을 도청하거나 조작할 수 있습니다.

HTTPS를 통해 암호화된 연결만 적용하려면 암호화되지 않은 SNS 주제에 연결된 IAM 정책에 `aws:SecureTransport` 조건을 추가합니다. 이렇게 하면 메시지 게시자가 HTTP 대신 HTTPS를 사용하게 됩니다. 다음 예제 정책을 가이드로 사용합니다.

```
{
  "Id": "ExamplePolicy",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowPublishThroughSSLOnly",
      "Action": "SNS:Publish",
      "Effect": "Deny",
      "Resource": [
        "arn:aws:sns:us-east-1:1234567890:test-topic"
      ],
      "Condition": {
        "Bool": {
          "aws:SecureTransport": "false"
        }
      },
      "Principal": "*"
    }
  ]
}
```

VPC 엔드포인트를 사용한 Amazon SNS 액세스 고려

상호 작용할 수 있어야 하지만 인터넷에 절대 노출되지 않아야 하는 주제가 있는 경우, VPC 엔드포인트를 사용하여 특정 VPC 내의 호스트에 대한 주제 액세스만 제한합니다. 주제 정책을 사용하여 특정 Amazon VPC 엔드포인트 또는 특정 VPC에서 주제에 대한 액세스를 제어할 수 있습니다.

Amazon SNS VPC 엔드포인트는 메시지에 대한 액세스를 제어하는 두 가지 방법을 제공합니다.

- 사용자는 특정 VPC 종단점을 통해 허용되는 요청, 사용자 또는 그룹을 제어할 수 있습니다.
- 주제 정책을 사용하여 주제에 액세스할 수 있는 VPC 또는 VPC 엔드포인트를 제어할 수 있습니다.

자세한 정보는 엔드포인트 생성 및 Amazon SNS용 Amazon VPC 엔드포인트 정책 생성에서 확인하세요.

구독이 원시 http 엔드포인트에 전달되도록 구성되지 않았는지 확인합니다.

구독이 원시 http 엔드포인트에 전달되도록 구성하지 마세요. 항상 구독을 엔드포인트 도메인 이름으로 전달해야 합니다. 예를 들어, 엔드포인트에 전달하도록 구성된 구독 `http://1.2.3.4/my-path`는 `http://my.domain.name/my-path`로 변경되어야 합니다.

Amazon SNS 주제 문제 해결

이 섹션에서는 Amazon SNS 주제 문제 해결에 대해 설명합니다.

AWS X-Ray를 사용하여 Amazon SNS 주제 문제 해결

AWS X-Ray은 애플리케이션이 제공하는 요청에 대한 데이터를 수집하고 데이터를 보고 필터링하여 잠재적 문제 및 최적화 기회를 식별할 수 있도록 합니다. 애플리케이션에 대한 모든 트레이스된 요청에서, 요청 및 응답뿐 아니라 애플리케이션이 다운스트림 AWS 리소스, 마이크로서비스, 데이터베이스 및 HTTP 웹 API에 대해 하는 호출에 대해서도 상세한 정보를 확인할 수 있습니다.

X-Ray와 Amazon SNS를 함께 사용하여 애플리케이션을 통한 메시지를 추적하고 분석할 수 있습니다. AWS Management Console을 사용하여 Amazon SNS와 애플리케이션이 사용하는 다른 서비스의 연결 맵을 볼 수 있습니다. 또한 콘솔을 사용하여 평균 대기 시간 및 실패율과 같은 메트릭을 볼 수 있습니다. 자세한 정보는 AWS X-Ray 개발자 안내서의 Amazon SNS 및 AWS X-Ray를 참조하세요.

Amazon SNS에서의 활성 추적

Amazon SNS 주제를 통해 Amazon Data Firehose AWS Lambda, Amazon SQS 및 HTTP/S 엔드포인트 구독으로 이동하는 사용자 요청을 추적하고 분석하는 데 사용할 AWS X-Ray 수 있습니다. X-Ray에서는 전체 요청을 end-to-end 볼 수 있으므로 Amazon SNS 주제라고 하는 주제와 주제 구독의 다운스트림에 해당하는 항목을 볼 수 있습니다. 메시지 및 해당 백엔드 서비스의 지연 시간을 분석할 수 있습니다(예: 특정 주제에서 요청이 소요되는 시간, 각 주제의 구독에 메시지를 전달하는 데 걸린 시간).

> **Important**
>
> 구독이 많은 Amazon SNS 주제는 크기 한도에 도달하여 추적이 완전하지 않을 수 있습니다. 추적 문서 크기 한도에 대한 자세한 내용은 AWS 일반 참조의 X-ray service quotas(X-ray 서비스 할당량)를 참조하세요.

이미 추적 중인 서비스에서 Amazon SNS API를 호출하는 경우, API에서 X-Ray 추적이 활성화되어 있지 않더라도 Amazon SNS가 트레이스를 패스스루합니다.

Amazon SNS는 표준 및 FIFO 주제에 대해 X-Ray 추적을 지원합니다. Amazon SNS 콘솔, Amazon SNS `SetTopicAttributes` API, Amazon Simple Notification Service CLI 참조 또는 AWS CloudFormation을 사용하여 Amazon SNS 주제에 대해 X-Ray를 활성화할 수 있습니다.

Amazon SNS를 X-Ray와 함께 사용하는 방법에 대한 자세한 내용은 AWS X-Ray 개발자 가이드의 Amazon SNS 및 AWS X-Ray를 참조하세요.

주제

활성 추적 권한

Amazon SNS 콘솔을 사용할 때 Amazon SNS는 Amazon SNS 주제가 X-Ray를 호출하는 데 필요한 권한을 생성하려고 시도합니다. Amazon SNS 콘솔을 사용할 충분한 권한이 없는 경우 시도가 거부될 수 있습니다. 자세한 내용은 Amazon SNS의 Identity and Access Management 및 Amazon SNS 액세스 제어의 예제 사례 섹션을 참조하세요.

CLI를 사용할 때는 권한을 수동으로 구성해야 합니다. 이러한 권한은 리소스 정책을 사용하여 구성됩니다. X-Ray에서 필요한 권한을 사용하는 방법에 대한 자세한 내용은 Amazon SNS 및 AWS X-Ray를 참조하세요.

Amazon SNS 주제에 대한 활성 추적 활성화(콘솔)

Amazon SNS 주제에서 활성 추적을 활성화하면 트레이스 ID를 읽고, 트레이스 ID를 기반으로 고객에게 데이터를 전송하고, 다운스트림 서비스에 트레이스 ID를 전파합니다.

1. Amazon SNS 콘솔에 로그인합니다.
2. 주제를 선택하거나 새로운 주제를 생성합니다. 주제 생성에 대한 자세한 내용은 Amazon SNS 주제 생성을 참조하세요.
3. 주제 생성 페이지의 세부 정보 섹션에서 FIFO 또는 표준을 선택합니다.
   a. 주제의 이름을 입력합니다.
   b. (선택 사항) 주제의 표시 이름을 입력합니다.

4. Active tracing(활성 추적)을 확장하고 Use active tracing(액티브 트레이싱 사용)을 선택합니다.

Amazon SNS 주제에 대해 X-Ray를 활성화한 후에는 X-Ray 서비스 맵을 사용하여 해당 주제에 대한 end-to-end 트레이스 및 서비스 맵을 볼 수 있습니다.

Amazon SNS 주제에 대한 활성 추적 활성화(AWS SDK)

다음 코드 예시는 AWS SDK for Java를 사용하여 Amazon SNS 주제에 활성 추적을 활성화하는 방법을 보여줍니다.

```
public static void enableActiveTracing(SnsClient snsClient, String topicArn) {

        try {

            SetTopicAttributesRequest request = SetTopicAttributesRequest.builder()
                .attributeName("TracingConfig")
                .attributeValue("Active")
                .topicArn(topicArn)
                .build();

            SetTopicAttributesResponse result = snsClient.setTopicAttributes(request);
            System.out.println("\n\nStatus was " +
 result.sdkHttpResponse().statusCode() + "\n\nTopic " + request.topicArn()
                + " updated " + request.attributeName() + " to " +
 request.attributeValue());

        } catch (SnsException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
    }
```

Amazon SNS 주제에 활성 추적 활성화(AWS CLI)

다음 코드 예시는 AWS CLI를 사용하여 Amazon SNS 주제에 활성 추적을 활성화하는 방법을 보여줍니다.

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name TracingConfig \
    --attribute-value Active
```

Amazon SNS 주제에 활성 추적 활성화(AWS CloudFormation)

다음 AWS CloudFormation 스택은 Amazon SNS 주제에 활성 추적을 활성화하는 방법을 보여줍니다.

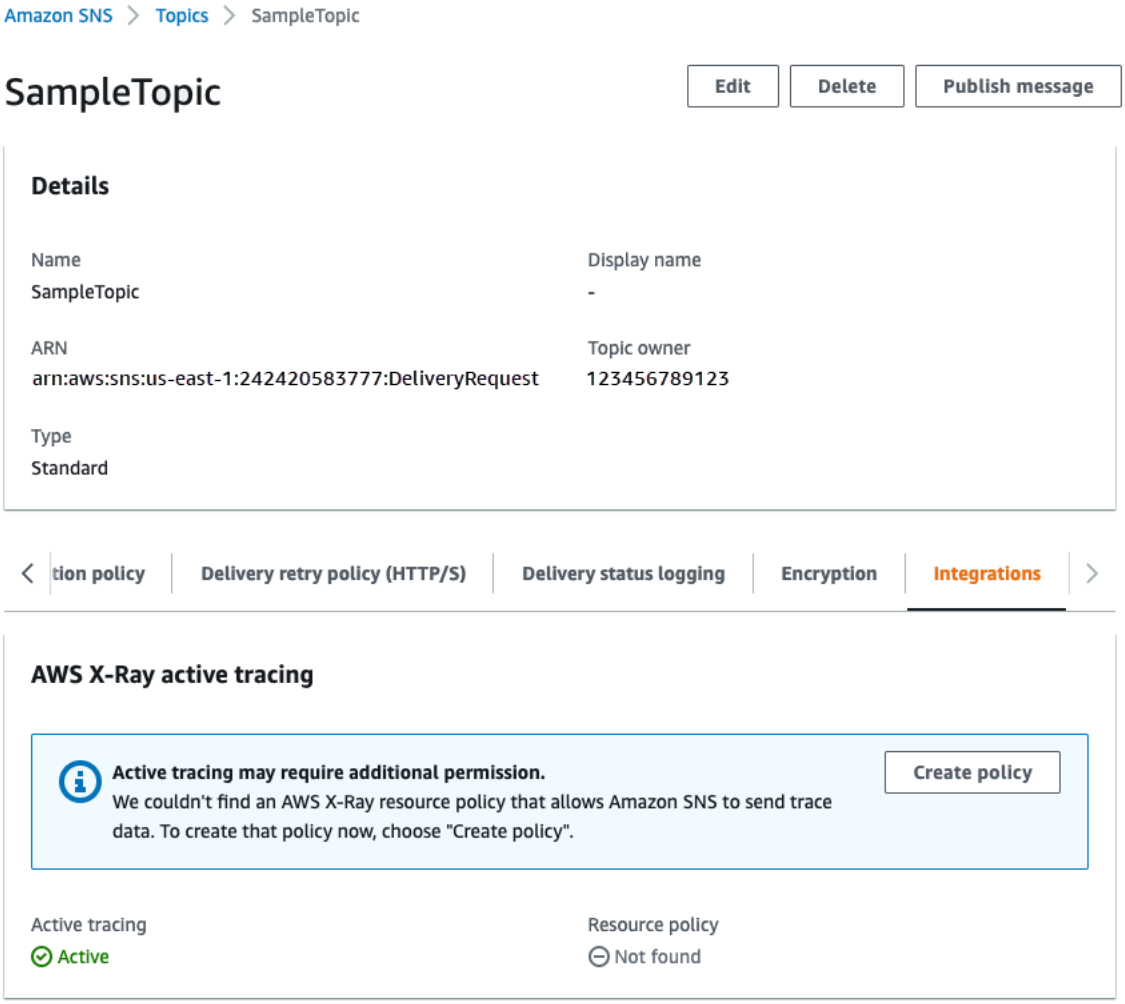
```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MyTopicResource:
    Type: 'AWS::SNS::Topic'
    Properties:
      TopicName: 'MyTopic'
      TracingConfig: 'Active'
```

주제에 활성 추적이 활성화되었는지 확인

Amazon SNS 콘솔을 사용하여 주제에 활성 추적이 활성화되어 있는지 확인하거나 리소스 정책을 추가하지 못했을 때 확인할 수 있습니다.

1. Amazon SNS 콘솔에 로그인합니다.
2. 왼쪽 탐색 창에서 주제를 선택합니다.
3. Topics(주제) 페이지에서 주제를 선택합니다.
4. 통합(Integrations) 탭을 선택합니다.

   활성 추적이 활성화되면 녹색 Active(활성) 아이콘이 표시됩니다.
5. 활성 추적을 활성화했는데 리소스 정책이 추가된 것이 보이지 않는 경우 Create policy(정책 생성)를 선택하여 필요한 추가 권한을 추가하세요.



활성 추적 테스트

1. Amazon SNS 콘솔에 로그인합니다.
2. Amazon SNS 주제를 생성합니다. 이 작업을 수행하는 자세한 방법은 를 사용하여 주제를 만들려면 AWS Management Console 단원을 참조하십시오.
3. Active tracing(활성 추적)을 확장하고 Use active tracing(액티브 트레이싱 사용)을 선택합니다.
4. Amazon SNS 주제에 메시지를 게시합니다. 이 작업을 수행하는 자세한 방법은 AWS Management Console을 사용하여 Amazon SNS 주제에 메시지를 게시하려면 단원을 참조하십시오.
5. X-Ray 서비스 맵을 사용하여 주제에 대한 end-to-end 추적 및 서비스 맵을 볼 수 있습니다.


51ms
0.2 t/min
https://sqs.us-east-1...
AWS::SQS::Queue
11ms
0.2 t/min
xray-topic
AWS::SNS::Topic
77ms
0.2 t/min
xray-lambda
AWS::Lambda
Client
387ms
0.2 t/min
remote
13ms
0.2 t/min
KinesisFirehose
AWS::KinesisFirehose

문서 기록

다음 표에서는 Amazon Simple Notification Service 개발자 안내서의 최근 변경 사항을 설명합니다.

서비스 기능은 서비스를 사용할 수 있는 AWS 지역에 점진적으로 배포되는 경우가 있습니다. 이 문서는 첫 번째 릴리스에 대해서만 업데이트됩니다. 리전 가용성에 대한 정보를 제공하거나 후속 리전 롤아웃을 발표하지 않습니다. 서비스 기능의 지역 가용성에 대한 자세한 내용과 업데이트에 대한 알림을 구독하려면 무엇이 새로워졌나요? 를 AWS 참조하십시오. .

변경 사항	설명	날짜
FIFO 주제에 대한 캐나다 서부(캘거리) 지원	Amazon SNS는 캐나다 서부(캘거리) 에서 FIFO 주제를 지원합니다.	2024년 3월 28일
5개의 새로운 지역에서 Amazon SNS SMS 지원	Amazon SNS는 아시아 태평양 (하이데라바드), 아시아 태평양 (멜버른), 중동 (UAE), 유럽 (취리히), 유럽 (스페인) 지역에 SMS 지원을 추가했습니다.	2024년 2월 8일
파이어베이스 클라우드 메시징 (FCM) HTTP v1 지원	Amazon SNS는 FCM v1 자격 증명을 지원합니다.	2024년 1월 18일
아시아 태평양(자카르타)에서 Amazon SNS SMS 지원	Amazon SNS가 아시아 태평양 (자카르타)에서 SMS를 지원합니다.	2023년 12월 14일
AWS CloudFormation Amazon SNS 주제에 DeliveryStatusLogging 대한 구성 지원	AWS CloudFormation Amazon SNS 주제를 만들거나 DeliveryStatusLogging 업데이트하는 동안 구성할 수 있도록 지원이 제공됩니다.	2023년 12월 7일
새 메시지 필터링 연산자가 추가됨	이제 Amazon SNS 메시지를 필터링할 때 접미사 일치,	2023년 11월 16일

	equals-ignore case 및 OR 연산자를 사용할 수 있습니다.	
메시지 아카이브 및 재생 지원 추가	주제 소유자는 메시지를 주제에 최대 365일 동안 아카이브할 수 있습니다. 주제 구독자는 아카이브된 메시지를 구독된 엔드포인트에 다시 재생하여 다운스트림 애플리케이션의 오류로 인해 메시지를 복구하거나 기존 애플리케이션의 상태를 복제할 수 있습니다.	2023년 10월 26일
표준 대기열에서 FIFO 주제 구독에 대한 지원 추가	Amazon SQS FIFO 대기열 또는 표준 대기열에서 Amazon SNS FIFO 주제를 구독할 수 있습니다. Amazon SQS FIFO 대기열만이 메시지가 중복되지 않고 순서대로 수신되도록 보장합니다.	2023년 9월 14일
이스라엘(텔아비브)에 대한 SMS 지원 추가	Amazon SNS SMS가 이제 이스라엘(텔아비브) 리전에서 지원됩니다.	2023년 8월 28일
FIFO 주제에 추가된 X-Ray 활성 추적에 대한 지원	이전에는 Amazon SNS 표준 주제에서만 지원되었지만 AWS X-Ray 이제는 FIFO 주제를 통해 Amazon Data Firehose AWS Lambda, Amazon SQS 및 HTTP/S 엔드포인트 구독으로 이동하는 사용자 요청을 추적하고 분석합니다.	2023년 5월 31일
확장 콘텐츠 유형 헤더 지원	요청 정책의 콘텐츠 유형 헤더를 설정해 알림의 미디어 유형을 지정합니다.	2023년 3월 23일

활성 추적 지원 추가	AWS X-Ray Amazon SNS 표준 주제를 통해 Amazon Data Firehose AWS Lambda, Amazon SQS 및 HTTP/S 엔드포인트 구독으로 이동하는 사용자 요청을 추적하고 분석합니다.	2023년 2월 8일
싱가포르 발신자 ID 등록	싱가포르에서 발신자 ID를 등록하기 위한 지침이 추가되었습니다.	2023년 1월 10일
페이로드 기반 메시지 필터링	페이로드 기반 필터링을 사용하면 메시지 페이로드를 기반으로 메시지를 필터링하여 원치 않는 데이터 처리와 관련된 비용을 피할 수 있습니다.	2022년 11월 22일
Amazon SNS 메시지 서명을 위해 SHA256 해시 알고리즘 추가	Amazon SNS 메시지 서명을 사용할 때 SHA256 해시 알고리즘에 대한 지원이 추가되었습니다.	2022년 9월 15일
SMS 메시징에 추가 리전 추가	Amazon SNS는 아프리카 (케이프타운), 아시아 태평양 (오사카), 유럽 (밀라노), AWS GovCloud (미국 동부) 지역에서 SMS 메시징을 지원합니다.	2022년 9월 9일
메시지 데이터 보호 지원 추가됨	메시지 데이터 보호는 데이터 보호 정책을 사용하여 애플리케이션 또는 AWS 서비스 간에 이동하는 민감한 정보를 감사 및 차단함으로써 Amazon SNS 주제에 게시된 데이터를 보호합니다.	2022년 9월 8일

새로운 수신자 부담 전화번호 등록 절차	수신자 부담 전화 번호(TFN)를 사용하여 미국 수신자에게 Amazon SNS 메시지를 보내는 지원이 추가되었습니다.	2022년 8월 1일
ABAC(속성 기반 액세스 제어) 지원	`Publish`와 `PublishBatch` 를 포함하는 API 작업에 대한 속성 기반 액세스 제어(ABAC) 지원을 추가했습니다. ABAC는 IAM 사용자 및 역할과 같은 IAM 리소스와 Amazon SNS 주제와 같은 리소스에 첨부하여 권한 관리를 단순화할 수 있는 태그를 기반으로 액세스 권한을 정의하는 권한 부여 전략입니다. AWS	2022년 1월 10일
푸시 알림에 대한 Apple 토큰 기반 인증 지원	앱 개발자임을 식별하는 정보를 제공하여 Amazon SNS가 iOS나 macOS 앱에 푸시 알림을 보낼 수 있는 권한을 부여할 수 있습니다.	2021년 10월 28일
SMS 메시지의 새 발신자는 SMS 샌드박스에 배치	SMS 샌드박스는 사기 및 부정 사용을 방지하고 발신자의 평판을 보호하기 위해 존재합니다. AWS 계정이 SMS 샌드박스에 있는 동안에는 확인된 대상 전화번호로만 SMS 메시지를 보낼 수 있습니다.	2021년 6월 1일

SMS 메시지의 새 발신자는 SMS 샌드박스에 배치	SMS 샌드박스는 사기 및 부정 사용을 방지하고 발신자의 평판을 보호하기 위해 존재합니다. AWS 계정이 SMS 샌드박스에 있는 동안에는 확인된 대상 전화번호로만 SMS 메시지를 보낼 수 있습니다.	2021년 6월 1일
인도의 수신자에게 SMS 메시지를 보내기 위한 새로운 속성	이제 인도의 수신자에게 SMS 메시지를 보내기 위해서는 엔터티 ID 및 템플릿 ID라는 두 개의 새로운 속성이 필요합니다.	2021년 4월 22일
메시지 필터링 연산자 업데이트	새 연산자 `cidr`은 일치하는 메시지 소스 IP 주소 및 서브넷에 사용할 수 있습니다. 이제 속성 키가 없는지 확인하고 속성 문자열 일치를 위해 `anything-but` 연산자와 함께 접두사를 사용할 수도 있습니다.	2021년 4월 7일
미국 대상에 대한 P2P 긴 코드 지원 종료	2021년 6월 1일부터 미국 통신 사업자는 미국 목적지로의 (A2P) 통신을 위한 person-to-person application-to-person (P2P) 롱 코드 사용을 더 이상 지원하지 않습니다. 대신 단축 코드, 무료 전화번호 또는 10DLC라는 새로운 유형의 발신 번호를 사용할 수 있습니다.	2021년 2월 16일
1분 아마존 CloudWatch 메트릭스 지원	Amazon SNS의 1분 CloudWatch 지표는 이제 모든 AWS 지역에서 사용할 수 있습니다.	2021년 1월 28일

Amazon Data Firehose 엔드포인트 지원	Firehose 전송 스트림을 구독하여 SNS 주제를 구독할 수 있습니다. 이를 통해 Amazon Simple Storage Service (Amazon S3) 버킷, Amazon Redshift 테이블, OpenSearch Amazon OpenSearch 서비스 (서비스) 등과 같은 보관 및 분석 엔드포인트에 알림을 보낼 수 있습니다.	2021년 1월 12일
발신 번호 사용 가능	문자 메시지(SMS)를 보낼 때 발신 번호를 사용할 수 있습니다.	2020년 10월 23일
Amazon SNS FIFO 주제 지원	거의 실시간으로 데이터 일관성이 필요한 분산 애플리케이션을 통합하기 위해 Amazon SQS FIFO 대기열과 함께 Amazon SNS FIFO(선입 선출) 주제를 사용할 수 있습니다.	2020년 10월 22일
Java용 Amazon SNS 확장 클라이언트 라이브러리를 사용 가능	이 라이브러리를 사용하여 대용량 Amazon SNS 메시지를 게시할 수 있습니다.	2020년 8월 25일
중국 리전에서 SSE 사용 가능	Amazon SNS용 서버 측 암호화(SSE)는 중국 리전에서 사용할 수 있습니다.	2020년 1월 20일
전달할 수 없는 메시지를 캡처하기 위한 DLQ 사용 지원	전달할 수 없는 메시지를 캡처하기 위해 Amazon SNS 구독과 함께 Amazon SQS 배달 못한 편지 대기열(DLQ)을 사용할 수 있습니다.	2019년 11월 14일
사용자 지정 APN 헤더 값 지정 지원	사용자 지정 APN 헤더 값을 지정할 수 있습니다.	2019년 10월 18일

APN에 대한 'apns-push-type' 헤더 필드 지원	APN을 통해 전송되는 모바일 알림에 apns-push-type 헤더 필드를 사용할 수 있습니다.	2019년 9월 10일
를 사용한 주제 문제 해결 지원 AWS X-Ray	X-Ray를 사용하여 SNS 주제를 통해 전달되는 메시지 문제를 해결할 수 있습니다.	2019년 7월 24일
'exists' 연산자를 사용하는 속성 키 일치 지원	수신 메시지에 필터 정책에 나열된 키의 속성이 있는지 확인하려면 exists 연산자를 사용할 수 있습니다.	2019년 7월 5일
다중 숫자 값의 anything-but 일치 지원	다중 문자열 외에도 Amazon SNS는 다중 숫자값의 anything-but 일치를 허용합니다.	2019년 7월 5일
Amazon SNS 릴리스 정보를 RSS 피드로 이용 가능	이 페이지의 제목(설명서 기록)을 따라 RSS를 선택합니다.	2019년 6월 22일

AWS 용어집

최신 AWS 용어는 AWS 용어집 참조서의 AWS 용어집을 참조하세요.

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.