aws

Panduan Developerr

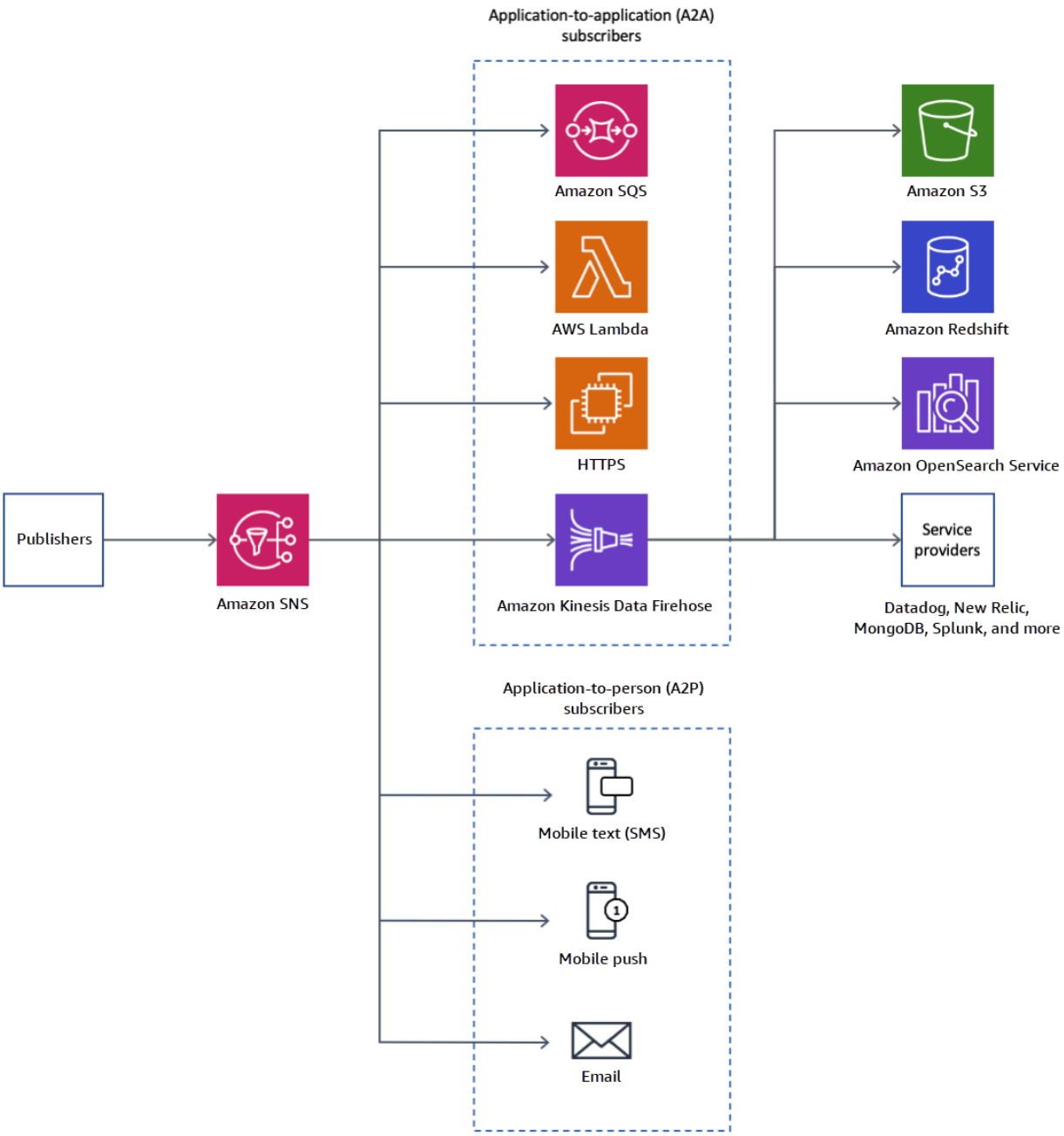
Amazon Simple Notification Service

Amazon Simple Notification Service: Panduan Developerr

Table of Contents

Apa yang dimaksud dengan Amazon SNS?

Amazon Simple Notification Service (AmazonSNS) adalah layanan terkelola yang menyediakan pengiriman pesan dari penerbit ke pelanggan (juga dikenal sebagai produsen dan konsumen). Penerbit berkomunikasi secara asinkron dengan pelanggan dengan mengirim pesan ke topik, yang merupakan titik akses logis dan saluran komunikasi. Klien dapat berlangganan SNS topik Amazon dan menerima pesan yang dipublikasikan menggunakan jenis titik akhir yang didukung, seperti Amazon Data Firehose, SQS Amazon,, AWS Lambda,HTTP, email, pemberitahuan push seluler, dan pesan SMS teks seluler ().



SNSFitur dan kemampuan Amazon

Amazon SNS menawarkan serangkaian fitur komprehensif yang dirancang untuk meningkatkan perpesanan antara aplikasi dan pengguna. Fitur-fitur ini memungkinkan komunikasi tanpa batas,

pengiriman pesan yang aman, dan manajemen pesan yang kuat, memastikan ketersediaan tinggi, daya tahan, dan fleksibilitas untuk berbagai kasus penggunaan pesan.

- Application-to-application pesan

  pplication-to-applicationPesan mendukung pelanggan seperti aliran pengiriman Amazon Data Firehose, fungsi Lambda, antrian SQS Amazon, titik akhir /SHTTP, dan Saluran Pipa Event Fork. AWS Ini memungkinkan pengiriman pesan yang efisien dalam arsitektur berbasis peristiwa.

- Application-to-person pemberitahuan

  pplication-to-personNotifikasi memberikan pemberitahuan pengguna kepada pelanggan seperti aplikasi seluler, nomor ponsel, dan alamat email.

- Standar dan FIFO topik

  FIFOtopik memastikan pemesanan pesan yang ketat, pengelompokan pesan, dan deduplikasi, memungkinkan FIFO dan antrian standar untuk berlangganan pemrosesan pesan. Topik standar digunakan saat pemesanan pesan dan kemungkinan duplikasi tidak penting, mendukung semua protokol pengiriman untuk kasus penggunaan yang lebih luas.

- Daya tahan pesan

  Amazon SNS menggunakan sejumlah strategi yang bekerja sama untuk memberikan daya tahan pesan:

  - Pesan yang diterbitkan disimpan di beberapa server dan pusat data yang terpisah secara geografis.
  - Jika titik akhir berlangganan tidak tersedia, Amazon SNS menjalankan kebijakan coba lagi pengiriman.
  - Untuk menyimpan pesan yang tidak terkirim sebelum kebijakan pengiriman ulang berakhir, Anda dapat membuat antrean surat mati.

- Pengarsipan pesan, pemutaran ulang, dan analitik

  Anda dapat mengarsipkan pesan dengan Amazon dengan berbagai SNS cara termasuk berlangganan aliran pengiriman Firehose SNS ke topik, yang memungkinkan Anda mengirim pemberitahuan ke titik akhir analitik seperti bucket Amazon Simple Storage Service (Amazon S3), tabel Amazon Redshift, dan banyak lagi. Selain itu, SNS FIFO topik Amazon mendukung pengarsipan pesan dan pemutaran ulang sebagai arsip pesan di tempat tanpa kode yang memungkinkan pemilik topik menyimpan (atau mengarsipkan) pesan dalam topik mereka. Pelanggan topik kemudian dapat mengambil (atau memutar ulang) pesan yang diarsipkan kembali

ke titik akhir berlangganan. Untuk lebih lanjut, lihatPengarsipan dan pemutaran ulang SNS pesan Amazon untuk topik FIFO.

- Atribut pesan

  Atribut SNS pesan Amazonmemungkinkan Anda memberikan metadata arbitrer tentang pesan tersebut.

- Pemfilteran pesan

  Secara default, setiap pelanggan menerima setiap pesan yang diterbitkan ke topik. Untuk menerima subset pesan, pelanggan harus menetapkan kebijakan filter untuk langganan topik. Pelanggan juga dapat menentukan cakupan kebijakan filter untuk mengaktifkan pemfilteran berbasis muatan atau atribusi. Nilai default untuk lingkup kebijakan filter adalah`MessageAttributes`. Ketika atribut pesan yang masuk sesuai dengan atribut kebijakan filter, pesan dikirim ke titik akhir langganan. Jika tidak, pesan akan difilter. Jika cakupan kebijakan filter berada`MessageBody`, atribut kebijakan filter dicocokkan dengan muatan. Untuk informasi selengkapnya, lihat Pemfilteran pesan.

- Keamanan pesan

  Enkripsi sisi server melindungi konten pesan yang disimpan dalam SNS topik Amazon, menggunakan kunci enkripsi yang disediakan oleh. AWS KMS Untuk informasi selengkapnya, lihat the section called “Mengamankan data dengan enkripsi sisi server”.

  Anda juga dapat membuat koneksi pribadi antara Amazon SNS dan cloud pribadi virtual Anda (VPC). untuk informasi selengkapnya, lihatthe section called “Mengamankan lalu lintas dengan titik akhir VPC”.

AWS layanan yang biasa digunakan dengan Amazon SNS

Anda dapat mengintegrasikan Amazon SNS dengan beberapa Layanan AWS untuk meningkatkan fungsionalitas dan pengelolaan. Layanan ini memungkinkan penanganan pesan yang dioptimalkan, kontrol akses yang aman, aplikasi berbasis peristiwa, dan penyediaan sumber daya otomatis.

- Amazon SQS menawarkan antrian host yang aman, tahan lama, dan tersedia yang memungkinkan Anda mengintegrasikan dan memisahkan sistem dan komponen perangkat lunak terdistribusi. Amazon SQS terkait dengan Amazon SNS dengan cara-cara berikut:
  - Amazon SNS menyediakan antrian surat mati yang didukung oleh Amazon SQS untuk pesan yang tidak terkirim.

  - Anda dapat berlangganan SQS antrian Amazon ke SNS topik Amazon.
  - Anda dapat berlangganan SQS FIFOantrian Amazon atau antrian standar ke topik Amazon SNS FIFO. Hanya SQS FIFO antrian Amazon yang menjamin pesan diterima secara berurutan dan tanpa duplikat.
- AWS Lambda memungkinkan Anda membangun aplikasi yang merespons informasi baru dengan cepat. Jalankan kode aplikasi Anda dalam fungsi Lambda pada infrastruktur komputasi dengan ketersediaan tinggi. Untuk informasi selengkapnya, lihat Panduan Developer AWS Lambda. Anda dapat berlangganan fungsi Lambda ke suatu SNS topik.
- AWS Identity and Access Management (IAM) membantu Anda mengontrol akses ke AWS sumber daya untuk pengguna Anda dengan aman. Gunakan IAM untuk mengontrol siapa yang dapat menggunakan SNS topik Amazon Anda (otentikasi), topik apa yang dapat mereka gunakan, dan bagaimana mereka dapat menggunakannya (otorisasi). Untuk informasi selengkapnya, lihat Menggunakan kebijakan berbasis identitas dengan Amazon SNS.
- AWS CloudFormationmemungkinkan Anda untuk memodelkan dan mengatur AWS sumber daya Anda. Buat templat yang menjelaskan AWS sumber daya yang Anda inginkan, termasuk SNS topik dan langganan Amazon. AWS CloudFormation mengurus penyediaan dan konfigurasi sumber daya tersebut untuk Anda. Untuk informasi selengkapnya, lihat Panduan Pengguna AWS CloudFormation.

Mengakses Amazon SNS

Anda dapat mengakses dan mengelola Amazon SNS melalui konsol, AWS CLI, atau AWS SDKs, tergantung pada metode interaksi pilihan Anda. Konsol menawarkan antarmuka grafis untuk tugas-tugas dasar, sementara AWS CLI dan SDKs menyediakan konfigurasi lanjutan dan kemampuan otomatisasi untuk kasus penggunaan yang lebih kompleks.

- SNSKonsol Amazon menyediakan antarmuka pengguna yang nyaman untuk membuat topik dan langganan, mengirim dan menerima pesan, serta memantau peristiwa dan log.
- The AWS Command Line Interface (AWS CLI) memberi Anda akses langsung ke Amazon SNS API untuk konfigurasi lanjutan dan kasus penggunaan otomatisasi. Untuk informasi selengkapnya, lihat Menggunakan Amazon SNS dengan AWS CLI.
- AWS menyediakan SDKs dalam berbagai bahasa. Untuk informasi selengkapnya, lihat SDKsdan Toolkit.

Harga untuk Amazon SNS

Amazon tidak SNS memiliki biaya di muka. Anda membayar berdasarkan jumlah pesan yang Anda terbitkan, jumlah notifikasi yang Anda kirimkan, dan API panggilan tambahan apa pun untuk mengelola topik dan langganan. Harga pengiriman bervariasi berdasarkan jenis titik akhir. Anda dapat memulai secara gratis dengan tingkat SNS gratis Amazon. Untuk selengkapnya, lihat SMSHarga Seluruh Dunia.

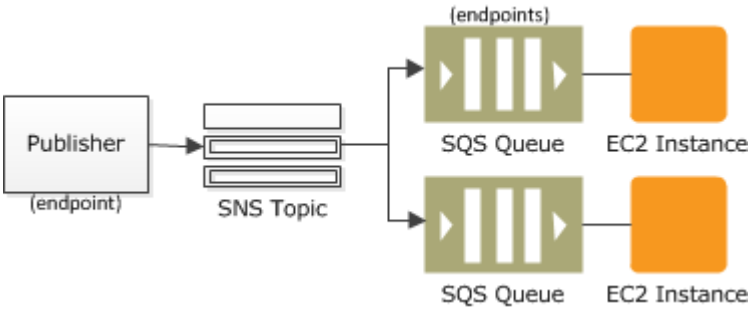
SNSSkenario Amazon yang umum

Gunakan SNS skenario Amazon umum ini untuk mengimplementasikan arsitektur yang dapat diskalakan dan didorong oleh peristiwa dan memastikan komunikasi real-time yang andal antara aplikasi dan pengguna.

Integrasi aplikasi

Skenario Fanout adalah ketika pesan yang dipublikasikan ke suatu SNS topik direplikasi dan didorong ke beberapa titik akhir, seperti aliran pengiriman Firehose, SQS antrian Amazon, titik akhir (HTTPS), dan fungsi Lambda. Skenario ini memungkinkan untuk pemrosesan asinkron paralel.

Misalnya, Anda dapat mengembangkan aplikasi yang menerbitkan pesan ke suatu SNS topik setiap kali pesanan dilakukan untuk suatu produk. Kemudian, SQS antrian yang berlangganan SNS topik menerima pemberitahuan identik untuk pesanan baru. Instans server Amazon Elastic Compute Cloud (AmazonEC2) yang dilampirkan ke salah satu SQS antrian dapat menangani pemrosesan atau pemenuhan pesanan. Dan Anda dapat melampirkan instance EC2 server Amazon lain ke gudang data untuk analisis semua pesanan yang diterima.



Anda juga dapat menggunakan fanout untuk mereplikasi data yang dikirim ke lingkungan produksi Anda dengan lingkungan pengujian Anda. Memperluas contoh sebelumnya, Anda dapat berlangganan SQS antrian lain ke SNS topik yang sama untuk pesanan masuk baru. Kemudian, dengan melampirkan SQS antrian baru ini ke lingkungan pengujian Anda, Anda dapat terus

meningkatkan dan menguji aplikasi Anda menggunakan data yang diterima dari lingkungan produksi Anda.

> **Important**
>
> Pastikan untuk mempertimbangkan privasi dan keamanan data sebelum Anda mengirim data produksi ke lingkungan pengujian Anda.

Untuk informasi selengkapnya, lihat sumber daya berikut:

- Aliran pengiriman Fanout ke Firehose
- SNSPemberitahuan Amazon Fanout ke fungsi Lambda untuk pemrosesan otomatis
- SNSPemberitahuan Fanout Amazon ke SQS antrian Amazon untuk pemrosesan asinkron
- SNSPemberitahuan Fanout Amazon ke titik akhir HTTPS
- Komputasi Berbasis Acara dengan Amazon SNS dan Layanan AWS Komputasi, Penyimpanan, Database, dan Jaringan

Pemberitahuan aplikasi

Pemberitahuan aplikasi dan sistem adalah notifikasi yang dipicu oleh ambang batas yang telah ditetapkan. Amazon SNS dapat mengirim pemberitahuan ini ke pengguna tertentu melalui SMS dan email. Misalnya, Anda dapat menerima pemberitahuan langsung saat peristiwa terjadi, seperti perubahan spesifik pada grup EC2 Auto Scaling Amazon, file baru yang diunggah ke bucket Amazon S3, atau ambang batas metrik yang dilanggar di Amazon. CloudWatch Untuk informasi selengkapnya, lihat Menyiapkan SNS notifikasi Amazon di Panduan CloudWatch Pengguna Amazon.

Notifikasi pengguna

Amazon SNS dapat mengirim pesan email push dan pesan teks (SMSpesan) ke individu atau grup. Misalnya, Anda dapat mengirim konfirmasi pesanan perdagangan elektronik sebagai notifikasi pengguna. Untuk informasi selengkapnya tentang menggunakan Amazon SNS untuk mengirim SMS pesan, lihatPesan teks seluler dengan Amazon SNS.

Notifikasi push seluler

Notifikasi push seluler memungkinkan Anda mengirim pesan secara langsung ke aplikasi seluler. Misalnya, Anda dapat menggunakan Amazon SNS untuk mengirim pemberitahuan pembaruan ke

aplikasi. Pesan notifikasi dapat menyertakan tautan untuk mengunduh dan menginstal pembaruan. Untuk informasi selengkapnya tentang menggunakan Amazon SNS untuk mengirim pesan pemberitahuan push, lihatMengirim notifikasi push seluler dengan Amazon SNS.

Menggunakan Amazon SNS dengan AWS SDK

AWS kit pengembangan perangkat lunak (SDKs) tersedia untuk banyak bahasa pemrograman populer. Masing-masing SDK menyediakanAPI, contoh kode, dan dokumentasi yang memudahkan pengembang untuk membangun aplikasi dalam bahasa pilihan mereka.

SDKdokumentasi	Contoh kode
AWS SDK for C++	AWS SDK for C++ contoh kode
AWS CLI	AWS CLI contoh kode
AWS SDK for Go	AWS SDK for Go contoh kode
AWS SDK for Java	AWS SDK for Java contoh kode
AWS SDK for JavaScript	AWS SDK for JavaScript contoh kode
AWS SDK for Kotlin	AWS SDK for Kotlin contoh kode
AWS SDK for .NET	AWS SDK for .NET contoh kode
AWS SDK for PHP	AWS SDK for PHP contoh kode
AWS Tools for PowerShell	Alat untuk contoh PowerShell kode
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) contoh kode
AWS SDK for Ruby	AWS SDK for Ruby contoh kode
AWS SDK for Rust	AWS SDK for Rust contoh kode
AWS SDK untuk SAP ABAP	AWS SDK untuk SAP ABAP contoh kode
AWS SDK for Swift	AWS SDK for Swift contoh kode

Untuk contoh khusus untuk AmazonSNS, lihatContoh kode untuk Amazon SNS menggunakan AWS SDKs.

> **Ketersediaan contoh**
>
> Tidak dapat menemukan apa yang Anda butuhkan? Minta contoh kode menggunakan tautan Berikan umpan balik di bagian bawah halaman ini.

Buat SNS topik Amazon dan publikasikan pesan

Topik ini memberikan langkah-langkah dasar untuk mengelola SNS sumber daya Amazon, yang secara khusus berfokus pada topik, langganan, dan penerbitan pesan. Pertama, Anda akan mengatur izin akses yang diperlukan untuk AmazonSNS, memastikan bahwa Anda memiliki izin yang benar untuk membuat dan mengelola sumber daya AmazonSNS. Selanjutnya, Anda akan membuat SNS topik Amazon baru, yang berfungsi sebagai hub pusat untuk mengelola dan mengirimkan pesan ke pelanggan. Setelah membuat topik, Anda akan melanjutkan untuk membuat langganan ke topik ini, memungkinkan titik akhir tertentu untuk menerima pesan yang dipublikasikan ke sana.

Setelah topik dan langganan tersedia, Anda akan mempublikasikan pesan ke topik tersebut, mengamati bagaimana Amazon SNS secara efisien mengirimkan pesan ke semua titik akhir berlangganan. Terakhir, Anda akan belajar cara menghapus langganan dan topik, menyelesaikan siklus hidup SNS sumber daya Amazon yang telah Anda kelola. Pendekatan ini memberi Anda pemahaman yang jelas tentang operasi mendasar di AmazonSNS, membekali Anda dengan keterampilan praktis yang diperlukan untuk mengelola alur kerja perpesanan menggunakan konsol AmazonSNS.

Menyiapkan akses untuk Amazon SNS

Sebelum Anda dapat menggunakan Amazon SNS untuk pertama kalinya, Anda harus menyelesaikan langkah-langkah berikut.

Topik

Buat Akun AWS dan IAM pengguna

Untuk mengakses AWS layanan apa pun, Anda harus terlebih dahulu membuat file Akun AWS. Anda dapat menggunakan laporan aktivitas dan penggunaan Anda untuk mengelola autentikasi dan akses. Akun AWS

Mendaftar untuk Akun AWS

Jika Anda tidak memiliki Akun AWS, selesaikan langkah-langkah berikut untuk membuatnya.

Untuk mendaftar untuk Akun AWS

1. Buka https://portal.aws.amazon.com/billing/pendaftaran.
2. Ikuti petunjuk online.

   Bagian dari prosedur pendaftaran melibatkan tindakan menerima panggilan telepon dan memasukkan kode verifikasi di keypad telepon.

   Ketika Anda mendaftar untuk Akun AWS, pengguna Akun AWS root dibuat. Pengguna root memiliki akses ke semua Layanan AWS dan sumber daya di akun. Sebagai praktik keamanan terbaik, tetapkan akses administratif ke pengguna, dan gunakan hanya pengguna root untuk melakukan tugas yang memerlukan akses pengguna root.

AWS mengirimi Anda email konfirmasi setelah proses pendaftaran selesai. Kapan saja, Anda dapat melihat aktivitas akun Anda saat ini dan mengelola akun Anda dengan masuk https://aws.amazon.comke/ dan memilih Akun Saya.

Buat pengguna dengan akses administratif

Setelah Anda mendaftar Akun AWS, amankan pengguna Akun AWS root Anda, aktifkan AWS IAM Identity Center, dan buat pengguna administratif sehingga Anda tidak menggunakan pengguna root untuk tugas sehari-hari.

Amankan pengguna Akun AWS root Anda

1. Masuk ke AWS Management Consolesebagai pemilik akun dengan memilih pengguna Root dan memasukkan alamat Akun AWS email Anda. Di laman berikutnya, masukkan kata sandi.

   Untuk bantuan masuk dengan menggunakan pengguna root, lihat Masuk sebagai pengguna root di AWS Sign-In Panduan Pengguna.
2. Aktifkan otentikasi multi-faktor (MFA) untuk pengguna root Anda.

   Untuk petunjuk, lihat Mengaktifkan MFA perangkat virtual untuk pengguna Akun AWS root (konsol) Anda di Panduan IAM Pengguna.

Buat pengguna dengan akses administratif

1. Aktifkan Pusat IAM Identitas.

Untuk mendapatkan petunjuk, silakan lihat Mengaktifkan AWS IAM Identity Center di Panduan Pengguna AWS IAM Identity Center .

2. Di Pusat IAM Identitas, berikan akses administratif ke pengguna.

   Untuk tutorial tentang menggunakan Direktori Pusat Identitas IAM sebagai sumber identitas Anda, lihat Mengkonfigurasi akses pengguna dengan default Direktori Pusat Identitas IAM di Panduan AWS IAM Identity Center Pengguna.

Masuk sebagai pengguna dengan akses administratif

- Untuk masuk dengan pengguna Pusat IAM Identitas, gunakan login URL yang dikirim ke alamat email saat Anda membuat pengguna Pusat IAM Identitas.

  Untuk bantuan masuk menggunakan pengguna Pusat IAM Identitas, lihat Masuk ke portal AWS akses di Panduan AWS Sign-In Pengguna.

Tetapkan akses ke pengguna tambahan

1. Di Pusat IAM Identitas, buat set izin yang mengikuti praktik terbaik menerapkan izin hak istimewa paling sedikit.

   Untuk petunjuknya, lihat Membuat set izin di Panduan AWS IAM Identity Center Pengguna.

2. Tetapkan pengguna ke grup, lalu tetapkan akses masuk tunggal ke grup.

   Untuk petunjuk, lihat Menambahkan grup di Panduan AWS IAM Identity Center Pengguna.

Langkah selanjutnya

Sekarang setelah Anda siap bekerja dengan AmazonSNS, mulailah dengan:

1. Membuat SNS topik Amazon
2. Membuat langganan ke SNS topik Amazon
3. Menerbitkan SNS pesan Amazon
4. Menghapus SNS topik dan langganan Amazon

Membuat SNS topik Amazon

SNSTopik Amazon adalah titik akses logis yang bertindak sebagai saluran komunikasi. Sebuah topik memungkinkan Anda mengelompokkan beberapa titik akhir (seperti AWS Lambda, AmazonSQS, HTTP /S, atau alamat email).

Untuk menyiarkan pesan dari sistem pembuat pesan (misalnya, sebuah situs web perdagangan elektronik) yang bekerja dengan beberapa layanan lain yang memerlukan pesannya (misalnya, sistem checkout/pembayaran dan pemenuhan), Anda dapat membuat topik untuk sistem pembuat pesan Anda.

SNSTugas Amazon pertama dan paling umum adalah membuat topik. Halaman ini menunjukkan bagaimana Anda dapat menggunakan AWS Management Console, yang AWS SDK for Java, dan AWS SDK for .NET untuk membuat topik.

Selama pembuatan, Anda memilih jenis topik (standar atauFIFO) dan memberi nama topik. Setelah membuat topik, Anda tidak dapat mengubah jenis atau nama topik. Semua pilihan konfigurasi lainnya bersifat opsional selama pembuatan topik, dan Anda dapat mengeditnya nanti.

> **Important**
>
> Jangan menambahkan informasi identitas pribadi (PII) atau informasi rahasia atau sensitif lainnya dalam nama topik. Nama topik dapat diakses oleh Amazon Web Services lainnya, termasuk CloudWatch Log. Nama topik tidak dimaksudkan untuk digunakan untuk data pribadi atau sensitif.

Topik

Untuk membuat topik menggunakan AWS Management Console

Membuat topik di Amazon SNS menetapkan dasar untuk distribusi pesan, memungkinkan Anda untuk mempublikasikan pesan yang dapat menyebar ke beberapa pelanggan. Langkah ini penting untuk mengonfigurasi jenis topik, pengaturan enkripsi, dan kebijakan akses, memastikan topik tersebut memenuhi persyaratan keamanan, kepatuhan, dan operasional organisasi.

1. Masuk ke SNSkonsol Amazon.
2. Lakukan salah satu hal berikut ini:

   - Jika tidak ada topik yang pernah dibuat di bawah Anda Akun AWS sebelumnya, baca deskripsi Amazon SNS di beranda.

   - Jika topik telah dibuat di bawah Akun AWS sebelumnya, pada panel navigasi, pilih Topik.
3. Di halaman Topics (Topik), pilih Create topic (Buat topik).
4. Di halaman Create topic (Buat topik), di bagian Details (Detail), lakukan hal-hal berikut:

   a. Untuk Jenis, pilih jenis topik (Standar atau FIFO).

   b. Masukkan Nama untuk topik. Untuk FIFOtopik, tambahkan .fifo ke akhir nama.

   c. (Opsional) Masukkan Nama tampilan untuk topik.

   > **Important**
   >
   > Saat berlangganan titik akhir email, jumlah karakter gabungan untuk nama tampilan SNS topik Amazon dan alamat email pengirim (misalnya, no-reply@sns.amazonaws.com) tidak boleh melebihi 320 UTF -8 karakter. Anda dapat menggunakan alat pengkodean pihak ketiga untuk memverifikasi panjang alamat pengiriman sebelum mengonfigurasi nama tampilan untuk topik Amazon SNS Anda.

   d. (Opsional) Untuk FIFO topik, Anda dapat memilih deduplikasi pesan berbasis konten untuk mengaktifkan deduplikasi pesan default. Untuk informasi selengkapnya, lihat Deduplikasi SNS pesan Amazon untuk topik FIFO.
5. (Opsional) Perluas bagian Encryption (Enkripsi) dan lakukan hal-hal berikut ini. Untuk informasi selengkapnya, lihat Mengamankan SNS data Amazon dengan enkripsi sisi server.

   a. Pilih Enable encryption (Aktifkan enkripsi).

   b. Tentukan AWS KMS kuncinya. Untuk informasi selengkapnya, lihat Istilah kunci.

      Untuk setiap KMS jenis, Deskripsi, Akun, dan KMSARNditampilkan.

> **Important**
>
> Jika Anda bukan pemilikKMS, atau jika Anda masuk dengan akun yang tidak memiliki izin `kms:ListAliases` dan `kms:DescribeKey` izin, Anda tidak akan dapat melihat informasi tentang SNS konsol Amazon. KMS
> Minta pemilik KMS untuk memberi Anda izin ini. Untuk informasi selengkapnya, lihat AWS KMS APIIzin: Tindakan dan Referensi Sumber Daya di Panduan AWS Key Management Service Pengembang.

- AWS Dikelola KMS untuk Amazon SNS (Default) alias/aws/sns dipilih secara default.

  > **Note**
  >
  > Ingatlah hal-hal berikut ini:
  >
  > - Pertama kali Anda menggunakan AWS Management Console untuk menentukan yang AWS dikelola KMS untuk Amazon SNS untuk suatu topik, AWS KMS membuat yang AWS dikelola KMS untuk AmazonSNS.
  > - Atau, saat pertama kali Anda menggunakan `Publish` tindakan pada topik yang SSE diaktifkan, AWS KMS membuat yang AWS dikelola KMS untuk AmazonSNS.

- Untuk menggunakan kustom KMS dari AWS akun Anda, pilih bidang KMSkunci dan kemudian pilih kustom KMS dari daftar.

  > **Note**
  >
  > Untuk petunjuk cara membuat kustomKMSs, lihat Membuat Kunci di Panduan AWS Key Management Service Pengembang

- Untuk menggunakan kustom KMS ARN dari AWS akun Anda atau dari AWS akun lain, masukkan ke dalam bidang KMSkunci.

6. (Opsional) Secara default, hanya pemilik topik yang dapat menerbitkan atau berlangganan topik. Untuk mengkonfigurasi izin akses tambahan, perluas bagian Access policy (Kebijakan akses). Untuk informasi selengkapnya, silakan lihat Manajemen identitas dan akses di Amazon SNS dan Contoh kasus untuk kontrol SNS akses Amazon.

> **Note**
>
> Saat Anda membuat topik menggunakan konsol tersebut, kebijakan default menggunakan kunci syarat `aws:SourceOwner`. Kunci ini sama dengan `aws:SourceAccount`.

7. (Opsional) Untuk mengonfigurasi cara Amazon SNS mencoba ulang upaya pengiriman pesan yang gagal, perluas bagian Kebijakan coba ulang pengiriman (HTTP/S). Untuk informasi selengkapnya, lihat Mencoba lagi pengiriman SNS pesan Amazon.
8. (Opsional) Untuk mengonfigurasi cara Amazon SNS mencatat pengiriman pesan ke CloudWatch, perluas bagian Pencatatan status pengiriman. Untuk informasi selengkapnya, lihat Status pengiriman SNS pesan Amazon.
9. (Opsional) Untuk menambahkan tag metadata ke topik, perluas bagian Tag, masukkan Kunci dan Nilai (opsional) dan pilih Add tag (Tambahkan tag). Untuk informasi selengkapnya, lihat Penandaan SNS topik Amazon.
10. Pilih Create topic (Buat topik).

    Topik dibuat dan ***MyTopic***halaman ditampilkan.

    Nama topik, ARN, (opsional) Nama tampilan, dan ID AWS akun pemilik Topik ditampilkan di bagian Detail.
11. Salin topik ARN ke clipboard, misalnya:

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

Untuk membuat topik menggunakan AWS SDK

Untuk menggunakan AWS SDK, Anda harus mengkonfigurasinya dengan kredensi Anda. Untuk informasi selengkapnya, lihat File konfigurasi dan kredensial bersama di Panduan Referensi Alat AWS SDKs dan Alat.

Contoh kode berikut menunjukkan cara menggunakan`CreateTopic`.

.NET

AWS SDK for .NET

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

Buat topik dengan nama tertentu.

```
    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;
        }
    }
```

Buat topik baru dengan nama dan atribut spesifik FIFO dan de-duplikasi.

```
    /// <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;
    }
```

- Untuk API detailnya, lihat CreateTopicdi AWS SDK for .NET APIReferensi.

C++

SDKuntuk C ++

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

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

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

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

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

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

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

    return outcome.IsSuccess();
}
```

- Untuk API detailnya, lihat CreateTopicdi AWS SDK for C++ APIReferensi.

CLI

AWS CLI

Untuk membuat SNS topik

`create-topic`Contoh berikut membuat SNS topik bernama`my-topic`.

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

Output:

```
{
    "ResponseMetadata": {
        "RequestId": "1469e8d7-1642-564e-b85d-a19b4b341f83"
```

```
    },
    "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
}
```

Untuk informasi selengkapnya, lihat Menggunakan Antarmuka Baris AWS Perintah dengan Amazon SQS dan Amazon SNS di Panduan Pengguna Antarmuka Baris AWS Perintah.

- Untuk API detailnya, lihat CreateTopicdi Referensi AWS CLI Perintah.

Go

SDKuntuk Go V2

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

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(ctx context.Context, topicName string,
 isFifoTopic bool, contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
```

```
  topicAttributes["ContentBasedDeduplication"] = "true"
 }
 topic, err := actor.SnsClient.CreateTopic(ctx, &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}
```

- Untuk API detailnya, lihat CreateTopicdi AWS SDK for Go APIReferensi.

Java

SDKuntuk Java 2.x

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

```
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 "";
    }
}
```

- Untuk API detailnya, lihat CreateTopicdi AWS SDK for Java 2.x APIReferensi.

JavaScript

SDKuntuk JavaScript (v3)

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

Buat klien dalam modul terpisah dan ekspor klien tersebut.

```
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({});
```

Impor modul SDK dan klien dan panggil fileAPI.

```
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;
};
```

- Untuk informasi selengkapnya, lihat AWS SDK for JavaScript Panduan Developer.
- Untuk API detailnya, lihat CreateTopicdi AWS SDK for JavaScript APIReferensi.

Kotlin

SDKuntuk Kotlin

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

```
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()
    }
}
```

- Untuk API detailnya, lihat CreateTopic AWSSDKAPIreferensi Kotlin.

PHP

SDKuntuk PHP

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

```
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());
}
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for PHP.
- Untuk API detailnya, lihat CreateTopicdi AWS SDK for PHP APIReferensi.

Python

SDKuntuk Python (Boto3)

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

```
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
```

- Untuk API detailnya, lihat CreateTopic AWSSDKReferensi Python (Boto3). API

Ruby

SDKuntuk Ruby

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

```
# 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
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for Ruby.
- Untuk API detailnya, lihat CreateTopicdi AWS SDK for Ruby APIReferensi.

Rust

SDKuntuk Rust

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

```
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(())
}
```

- Untuk API detailnya, lihat CreateTopic AWSSDKuntuk API referensi Rust.

SAP ABAP

SDKuntuk SAP ABAP

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

```
   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.
```

- Untuk API detailnya, lihat CreateTopic AWSSDKuntuk SAP ABAP API referensi.

Membuat langganan ke SNS topik Amazon

Untuk menerima olahpesan yang dipublikasikan ke Topik, Anda harus berlangganan endpoint ke topik. Ketika Anda berlangganan endpoint untuk topik, endpoint mulai menerima olahpesan yang diterbitkan untuk topik terkait.

> **Note**
>
> HTTP(S) titik akhir, alamat email, dan AWS sumber daya lainnya Akun AWS memerlukan konfirmasi langganan sebelum mereka dapat menerima pesan.

Untuk berlangganan titik akhir ke topik Amazon SNS

Berlangganan titik akhir ke SNS topik Amazon memungkinkan pengiriman pesan ke titik akhir yang ditentukan, memastikan sistem atau pengguna yang tepat menerima pemberitahuan saat pesan

dipublikasikan ke topik tersebut. Langkah ini penting untuk menghubungkan topik ke konsumen — apakah itu aplikasi, penerima email, atau layanan lainnya — memungkinkan komunikasi tanpa batas di seluruh sistem.

1. Masuk ke SNSkonsol Amazon.
2. Di panel navigasi kiri, pilih Subscriptions (Langganan).
3. Di halaman Subscriptions (Langganan), pilih Create subscription (Buat langganan).
4. Di halaman Create subscription (Buat langganan), di bagian Details (Detail), lakukan:

   a. Untuk Topik ARN, pilih Amazon Resource Name (ARN) dari suatu topik. Nilai ini adalah AWS ARN yang dihasilkan saat Anda membuat SNS topik Amazon, misalnya`aarn:aws:sns:us-east-2:123456789012:your_topic`.

   b. Untuk Protocol (Protokol), pilih tipe endpoint. Tipe endpoint yang tersedia adalah:

      - HTTP/HTTPS
      - Email/Email- JSON
      - Amazon Data Firehose
      - Amazon SQS

      > **Note**
      >
      > Untuk berlangganan SNSFIFOtopik, pilih opsi ini.

      - AWS Lambda
      - Titik akhir aplikasi platform
      - SMS

   c. Untuk Endpoint, masukkan nilai endpoint, seperti alamat email atau ARN antrian AmazonSQS.

   d. Hanya titik akhir Firehose: Untuk peran Langganan ARN, tentukan IAM peran ARN yang Anda buat untuk menulis ke aliran pengiriman Firehose. Untuk informasi selengkapnya, lihat Prasyarat untuk berlangganan aliran pengiriman Firehose ke topik Amazon SNS.

   e. (Opsional) Untuk FirehoseSQS, Amazon, HTTP /S endpoint, Anda juga dapat mengaktifkan pengiriman pesan mentah. Untuk informasi selengkapnya, lihat Pengiriman pesan SNS mentah Amazon.

f. (Opsional) Untuk mengkonfigurasi kebijakan filter, perluas bagian Subscription filter policy (Kebijakan filter langganan). Untuk informasi selengkapnya, lihat Kebijakan filter SNS langganan Amazon.

g. (Opsional) Untuk mengaktifkan pemfilteran berbasis muatan, konfigurasikan ke. `Filter Policy Scope` `MessageBody` Untuk informasi selengkapnya, lihat Lingkup kebijakan filter SNS langganan Amazon.

h. (Opsional) Untuk mengonfigurasi antrean surat mati untuk berlanggana, perluas bagian Redrive policy (dead-letter queue) (Kebijakan redrive (antrean surat mati)). Untuk informasi selengkapnya, lihat Antrian SNS surat mati Amazon.

i. Pilih Create subscription (Buat langganan).

   Konsol tersebut membuat langganan dan membuka halaman Details (Detail) langganan.

Menerbitkan SNS pesan Amazon

Setelah membuat SNS topik Amazon dan berlangganan titik akhir, Anda dapat mempublikasikan pesan ke topik tersebut. Saat pesan dipublikasikan, Amazon SNS mencoba mengirimkan pesan ke titik akhir berlangganan.

Topik

- Untuk mempublikasikan pesan ke SNS topik Amazon menggunakan AWS Management Console
- Untuk mempublikasikan pesan ke topik menggunakan AWS SDK
- Menerbitkan pesan besar dengan Amazon SNS dan Amazon S3
- Atribut SNS pesan Amazon
- SNSPengelompokan pesan Amazon

Untuk mempublikasikan pesan ke SNS topik Amazon menggunakan AWS Management Console

1. Masuk ke SNSkonsol Amazon.
2. Di panel navigasi kiri, pilih Topics (Topik).
3. Di halaman Topics (Topik), pilih topik, lalu pilih Publish message (Terbitkan pesan).

   Konsol membuka halaman Publish message to topic (Terbitkan pesan ke topik).

4. Di bagian Message details (Detail pesan), lakukan hal berikut:

   a. (Opsional) Masukkan Subjek pesan.

   b. Untuk FIFOtopik, masukkan ID grup Pesan. Pesan-pesan dalam grup pesan yang sama akan dikirim sesuai urutan yang penerbitannya.

   c. Untuk FIFO topik, masukkan ID deduplikasi Pesan. ID ini bersifat opsional jika Anda mengaktifkan pengaturan Deduplikasi pesan berbasis konten untuk topik.

   d. (Opsional) Untuk pemberitahuan push seluler, masukkan nilai Time to Live (TTL) dalam hitungan detik. Ini adalah jumlah waktu yang dimiliki layanan notifikasi push — seperti Apple Push Notification Service (APNs) atau Firebase Cloud Messaging (FCM) —untuk mengirimkan pesan ke titik akhir.

5. Di bagian Message body (Isi pesan), lakukan salah satu hal berikut:

   a. Pilih Identical payload for all delivery protocols (Muatan identik untuk semua protokol pengiriman), lalu masukkan pesan.

   b. Pilih Payload khusus untuk setiap protokol pengiriman, lalu masukkan JSON objek untuk menentukan pesan yang akan dikirim untuk setiap protokol pengiriman.

      Untuk informasi selengkapnya, lihat Menerbitkan SNS notifikasi Amazon dengan muatan khusus platform.

6. Di bagian Atribut pesan, tambahkan atribut apa pun yang Anda SNS ingin Amazon cocokkan dengan atribut langganan `FilterPolicy` untuk memutuskan apakah titik akhir berlangganan tertarik pada pesan yang dipublikasikan.

   a. Untuk Type (Jenis), pilih jenis atribut, seperti String.Array.

      > **Note**
      >
      > Untuk jenis atribut String.Array, lampirkan array dalam tanda kurung siku (`[]`). Dalam array, lampirkan nilai string dalam tanda kutip ganda. Anda tidak memerlukan tanda kutip untuk angka atau kata kunci `true`, `false`, dan `null`.

   b. Masukkan atribut Name (Nama), seperti `customer_interests`.

   c. Masukkan atribut Value (Nilai), seperti `["soccer", "rugby", "hockey"]`.

Jika jenis atribut adalah String, String.Array, atau Number, Amazon SNS mengevaluasi atribut pesan terhadap kebijakan filter langganan (jika ada) sebelum mengirim pesan ke cakupan kebijakan filter yang diberikan langganan tidak disetel secara eksplisit. `MessageBody`

Untuk informasi selengkapnya, lihat Atribut SNS pesan Amazon.

7. Pilih Publish message (Terbitkan pesan).

   Pesan diterbitkan ke topik, dan konsol membuka halaman Detail topik.

Untuk mempublikasikan pesan ke topik menggunakan AWS SDK

Untuk menggunakan AWS SDK, Anda harus mengkonfigurasinya dengan kredensi Anda. Untuk informasi selengkapnya, lihat File konfigurasi dan kredensial bersama di Panduan Referensi Alat AWS SDKs dan Alat.

Contoh kode berikut menunjukkan cara menggunakan`Publish`.

.NET

AWS SDK for .NET

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

Publikasikan pesan ke topik.

```
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}");
        }
    }
```

Publikasikan pesan ke topik dengan opsi grup, duplikasi, dan atribut.

```
    /// <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);
        }
    }
```

Terapkan pilihan pengguna ke tindakan publikasi.

```
    /// <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;
    }
```

- Untuk API detailnya, lihat Publikasikan di AWS SDK for .NET APIReferensi.

C++

SDKuntuk C ++

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

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

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

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

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

    return outcome.IsSuccess();
}
```

Publikasikan pesan dengan atribut.

```
       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;
        }
```

- Untuk API detailnya, lihat Publikasikan di AWS SDK for C++ APIReferensi.

CLI

AWS CLI

Contoh 1: Untuk mempublikasikan pesan ke topik

`publish`Contoh berikut menerbitkan pesan yang ditentukan ke SNS topik yang ditentukan. Pesan berasal dari file teks, yang memungkinkan Anda untuk memasukkan jeda baris.

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

Isi dari `message.txt`:

```
Hello World
Second Line
```

Output:

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

Contoh 2: Untuk mempublikasikan SMS pesan ke nomor telepon

`publish`Contoh berikut menerbitkan pesan `Hello world!` ke nomor `+1-555-555-0100` telepon.

```
aws sns publish \
```

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

Output:

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

- Untuk API detailnya, lihat Publikasikan di Referensi AWS CLI Perintah.

Go

SDKuntuk Go V2

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

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
```

```
func (actor SnsActions) Publish(ctx context.Context, topicArn string, message
 string, groupId string, dedupId string, filterKey string, filterValue string)
 error {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(ctx, &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
 return err
}
```

- Untuk API detailnya, lihat Publikasikan di AWS SDK for Go APIReferensi.

Java

SDKuntuk Java 2.x

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

```
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);
        }
    }
}
```

- Untuk API detailnya, lihat Publikasikan di AWS SDK for Java 2.x APIReferensi.

JavaScript

SDKuntuk JavaScript (v3)

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

Buat klien dalam modul terpisah dan ekspor klien tersebut.

```
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({});
```

Impor modul SDK dan klien dan panggil fileAPI.

```
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;
};
```

Publikasikan pesan ke topik dengan opsi grup, duplikasi, dan atribut.

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

    let groupId;
    let deduplicationId;
```

```
    let 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();
    }
  }
```

- Untuk informasi selengkapnya, lihat AWS SDK for JavaScript Panduan Developer.
- Untuk API detailnya, lihat Publikasikan di AWS SDK for JavaScript APIReferensi.

Kotlin

SDKuntuk Kotlin

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

```
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.")
    }
```

```
}
```

- Untuk API detailnya, lihat Publish in AWS SDKuntuk API referensi Kotlin.

PHP

SDKuntuk PHP

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

```
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());
}
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for PHP.
- Untuk API detailnya, lihat Publikasikan di AWS SDK for PHP APIReferensi.

PowerShell

Alat untuk PowerShell

Contoh 1: Contoh ini menunjukkan penerbitan pesan dengan satu baris yang MessageAttribute dideklarasikan.

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

Contoh 2: Contoh ini menunjukkan penerbitan pesan dengan beberapa MessageAttributes dideklarasikan sebelumnya.

```
$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
```

- Untuk API detailnya, lihat Menerbitkan di AWS Tools for PowerShell Referensi Cmdlet.

Python

SDKuntuk Python (Boto3)

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

Publikasikan pesan dengan atribut sehingga langganan dapat memfilter berdasarkan atribut.

```
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
```

Publikasikan pesan yang mengambil bentuk berbeda berdasarkan protokol pelanggan.

```
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
```

- Untuk API detailnya, lihat Publikasikan AWS SDKuntuk Referensi Python (Boto3). API

Ruby

SDKuntuk Ruby

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

```
# 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
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for Ruby.
- Untuk API detailnya, lihat Publikasikan di AWS SDK for Ruby APIReferensi.

Rust

SDKuntuk Rust

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

```
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(())
}
```

- Untuk API detailnya, lihat Publikasikan AWS SDKuntuk API referensi Rust.

SAP ABAP

SDKuntuk SAP ABAP

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

```
    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.
```

- Untuk API detailnya, lihat Publikasikan AWS SDKuntuk SAP ABAP API referensi.

Menerbitkan pesan besar dengan Amazon SNS dan Amazon S3

Untuk mempublikasikan SNS pesan Amazon besar, Anda dapat menggunakan Amazon SNS Extended Client Library untuk Java, atau Amazon SNS Extended Client Library untuk Python. Pustaka ini berguna untuk pesan yang lebih besar dari maksimum 256 KB saat ini, dengan maksimum 2 GB. Kedua pustaka menyimpan muatan aktual ke bucket Amazon S3, dan mempublikasikan referensi objek Amazon S3 yang disimpan ke topik Amazon. SNS SQSAntrian Amazon berlangganan dapat menggunakan Amazon SQS Extended Client Library for Java untuk menghilangkan referensi dan mengambil muatan dari Amazon S3. Endpoint lain seperti Lambda dapat menggunakan Payload Offloading Java Common Library AWS untuk de-referensi dan mengambil payload.

> **Note**
>
> Amazon SNS Extended Client Libraries kompatibel dengan standar dan FIFO topik.

Topik

Amazon SNS Extended Client Library untuk Java

Topik

Prasyarat

Berikut ini adalah prasyarat untuk menggunakan Amazon SNS Extended Client Library for Java:

- Sebuah AWS SDK.

  Contoh pada halaman ini menggunakan AWS JavaSDK. Untuk menginstal dan mengaturSDK, lihat Mengatur AWS SDK untuk Java di Panduan AWS SDK for Java Pengembang.

- An Akun AWS dengan kredensi yang tepat.

  Untuk membuat Akun AWS, navigasikan ke AWS halaman beranda, lalu pilih Buat AWS Akun. Ikuti petunjuk online.

  Untuk informasi tentang kredensional, lihat Menyiapkan AWS Kredensial dan Wilayah untuk Pengembangan di Panduan Pengembang.AWS SDK for Java
- Java 8 atau lebih baik.
- Amazon SNS Extended Client Library untuk Java (juga tersedia dari Maven).

Mengkonfigurasi penyimpanan pesan

Library Amazon SNS Extended Client menggunakan Payload Offloading Java Common Library AWS untuk penyimpanan dan pengambilan pesan. Anda dapat mengkonfigurasi Amazon S3 berikut opsi penyimpanan pesan:

- Batas ukuran pesan kustom - Pesan dengan muatan dan atribut yang melebihi ukuran ini secara otomatis disimpan di Amazon S3.
- `alwaysThroughS3` flag - Mengatur nilai ini untuk `true` Untuk memaksa semua muatan pesan yang disimpan di Amazon S3. Sebagai contoh:

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

- KMSKunci khusus — Kunci yang digunakan untuk enkripsi sisi server di bucket Amazon S3 Anda.
- Nama bucket - Nama bucket Amazon S3 untuk menyimpan muatan pesan.

Contoh: Menerbitkan pesan ke Amazon SNS dengan muatan yang disimpan di Amazon S3

Contoh kode berikut ini menunjukkan cara:

- Buat contoh topik dan antrean.
- Berlangganan antrean untuk menerima pesan dari topik.
- Publikasikan pesan percobaan.

Muatan pesan disimpan di Amazon S3 dan referensi untuk diterbitkan. Amazon SQS Extended Client digunakan untuk menerima pesan.

SDKuntuk Java 1.x

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

Untuk mempublikasikan pesan besar, gunakan Amazon SNS Extended Client Library for Java. Pesan yang Anda kirim mereferensikan objek Amazon S3 yang berisi konten pesan yang sebenarnya.

```
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());
        }
}
```

Protokol titik akhir lainnya

SQSPustaka Amazon SNS dan Amazon menggunakan Payload Offloading Java Common Library AWS untuk menyimpan dan mengambil muatan pesan dengan Amazon S3. Setiap titik akhir yang mendukung Java (misalnya, HTTPS titik akhir yang diimplementasikan di Java) dapat menggunakan pustaka yang sama untuk menghilangkan referensi konten pesan.

Titik akhir yang tidak dapat menggunakan Payload Offloading Java Common Library untuk masih AWS dapat mempublikasikan pesan dengan muatan yang disimpan di Amazon S3. Berikut ini adalah contoh dari referensi Amazon S3 yang diterbitkan oleh contoh kode di atas:

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

Amazon SNS Extended Client Library untuk Python

Topik

- Prasyarat
- Mengkonfigurasi penyimpanan pesan
- Contoh: Menerbitkan pesan ke Amazon SNS dengan muatan yang disimpan di Amazon S3

Prasyarat

Berikut ini adalah prasyarat untuk menggunakan Amazon SNS Extended Client Library untuk Python:

- Sebuah AWS SDK.

  Contoh pada halaman ini menggunakan AWS Python SDK Boto3. Untuk menginstal dan mengaturSDK, lihat dokumentasi AWS SDKuntuk Python.
- An Akun AWS dengan kredensi yang tepat.

  Untuk membuat Akun AWS, navigasikan ke AWS halaman beranda, lalu pilih Buat AWS Akun. Ikuti petunjuk online.

  Untuk informasi tentang kredensional, lihat Kredensial di Panduan Pengembang untuk AWS SDKPython.
- Python 3.x (atau yang lebih baru) dan pip.
- Amazon SNS Extended Client Library untuk Python (juga tersedia dari PyPI).

Mengkonfigurasi penyimpanan pesan

Atribut di bawah ini tersedia di SNS Klien, Topik, dan PlatformEndpointobjek Amazon Boto3 untuk mengonfigurasi opsi penyimpanan pesan Amazon S3.

- `large_payload_support`— Nama bucket Amazon S3 untuk menyimpan pesan besar.
- `message_size_threshold`— Ambang batas untuk menyimpan pesan di ember pesan besar. Tidak boleh kurang dari 0, atau lebih besar dari 262144. Defaultnya adalah 262144.
- `always_through_s3`— Jika`True`, maka semua pesan disimpan di Amazon S3. Default-nya adalah `False`.
- `s3`— Objek Boto3 Amazon `resource` S3 yang digunakan untuk menyimpan objek di Amazon S3. Gunakan ini jika Anda ingin mengontrol sumber daya Amazon S3 (misalnya, konfigurasi atau kredenal Amazon S3 kustom). Jika sebelumnya tidak disetel pada penggunaan pertama, defaultnya adalah `boto3.resource("s3")`.

Contoh: Menerbitkan pesan ke Amazon SNS dengan muatan yang disimpan di Amazon S3

Contoh kode berikut ini menunjukkan cara:

- Buat contoh SNS topik Amazon dan SQS antrian Amazon.
- Berlangganan antrean untuk menerima pesan dari topik.
- Publikasikan pesan percobaan.
- Payload pesan disimpan di Amazon S3, dan referensi untuk itu diterbitkan.
- Cetak pesan yang diterbitkan dari antrian bersama dengan pesan asli yang diambil dari Amazon S3.

Untuk mempublikasikan pesan besar, gunakan Amazon SNS Extended Client Library untuk Python. Pesan yang Anda kirim mereferensikan objek Amazon S3 yang berisi konten pesan yang sebenarnya.

```
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)
```

Keluaran

```
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
```

Atribut SNS pesan Amazon

Amazon SNS mendukung pengiriman atribut pesan, yang memungkinkan Anda menyediakan item metadata terstruktur (seperti stempel waktu, data geospasial, tanda tangan, dan pengidentifikasi) tentang pesan tersebut. Untuk SQS langganan, maksimal 10 atribut pesan dapat dikirim saat Pengiriman Pesan Mentah diaktifkan. Untuk mengirim lebih dari 10 atribut pesan, Pengiriman Pesan Mentah harus dinonaktifkan. Pesan dengan lebih dari 10 atribut pesan yang diarahkan ke SQS langganan Amazon yang diaktifkan Pengiriman Pesan Mentah akan dibuang sebagai kesalahan sisi klien.

Atribut pesan bersifat opsional dan terpisah dari—tetapi dikirim bersama-sama dengan—isi pesan. Penerima dapat menggunakan informasi ini untuk memutuskan bagaimana menangani pesan tanpa harus memproses isi pesan terlebih dahulu.

Untuk informasi tentang mengirim pesan dengan atribut menggunakan AWS Management Console atau AWS SDK for Java, lihat Untuk mempublikasikan pesan ke SNS topik Amazon menggunakan AWS Management Console tutorialnya.

> **Note**
>
> Atribut pesan dikirim hanya ketika struktur pesan adalah String, bukanJSON.

Anda dapat menggunakan atribut pesan untuk menyusun pesan notifikasi push untuk endpoint seluler. Dalam skenario ini, atribut pesan hanya digunakan untuk menyusun pesan notifikasi push. Atribut tidak dikirim ke titik akhir seperti saat mengirim pesan dengan atribut pesan ke titik SQS akhir Amazon.

Anda juga dapat menggunakan atribut pesan untuk membuat pesan Anda dapat difilter menggunakan kebijakan filter langganan. Anda dapat menerapkan kebijakan filter untuk langganan topik. Jika kebijakan filter diterapkan dengan cakupan kebijakan filter yang disetel ke `MessageAttributes` (default), langganan hanya menerima pesan yang memiliki atribut yang diterima kebijakan tersebut. Untuk informasi selengkapnya, lihat SNSPemfilteran pesan Amazon.

> **Note**
>
> Ketika atribut pesan digunakan untuk pemfilteran, nilainya harus berupa JSON string yang valid. Melakukan hal ini memastikan bahwa pesan dikirimkan ke langganan dengan pemfilteran atribut pesan diaktifkan.

Pesan atribut item dan validasi

Setiap atribut pesan terdiri atas beberapa item berikut:

- Name (Nama)— Nama atribut pesan dapat berisi karakter berikut: A-Z, a-z, 0-9, garis bawah (_), tanda hubung (-), dan periode (.). Nama tidak harus dimulai atau diakhiri dengan titik, dan seharusnya tidak memiliki titik berturut-turut. Nama peka huruf besar/kecil dan harus unik di antara semua nama atribut untuk pesan. Panjang nama dapat mencapai 256 karakter. Nama tidak dapat dimulai dengan `AWS.` atau `Amazon.` (atau variasi dalam casing) karena awalan ini disediakan untuk digunakan oleh Amazon Web Services.
- Type (Jenis) — Jenis data atribut pesan yang didukung `String`, `String.Array`, `Number`, dan `Binary`. Tipe data memiliki batasan konten yang sama dengan isi pesan. Untuk informasi selengkapnya, lihat bagian Pesan jenis data atribut dan validasi.

- Value (Nilai) — Nilai atribut pesan yang ditentukan pengguna. Untuk jenis data string, atribut nilai memiliki batasan konten yang sama dengan isi pesan. Untuk informasi selengkapnya, lihat tindakan Publikasikan di APIReferensi Layanan Pemberitahuan Sederhana Amazon.

Nama, jenis, dan nilai tidak boleh kosong atau nol. Selain itu, isi pesan tidak boleh kosong atau nol. Semua bagian dari atribut pesan, termasuk nama, jenis, dan nilai, termasuk dalam pembatasan ukuran pesan, yaitu 256 KB.

Pesan jenis data atribut dan validasi

Tipe data atribut pesan mengidentifikasi bagaimana nilai atribut pesan ditangani oleh AmazonSNS. Misalnya, jika jenisnya adalah angka, Amazon SNS memvalidasi bahwa itu adalah angka.

Amazon SNS mendukung tipe data logis berikut untuk semua titik akhir kecuali seperti yang disebutkan:

- String — String adalah Unicode dengan UTF -8 pengkodean biner. Untuk daftar nilai kode, lihat http://en.wikipedia. org/wiki/ASCII# ASCII _printable_characters.

  > **Note**
  >
  > Nilai pengganti tidak didukung dalam atribut pesan. Misalnya, menggunakan nilai pengganti untuk mewakili emoji akan memberi Anda kesalahan berikut:`Invalid attribute value was passed in for message attribute`.

- String.Array – Array, diformat sebagai string, yang dapat berisi beberapa nilai. Nilai dapat berupa string, angka, atau kata kunci `true`, `false`, dan `null`. String.Array nomor atau tipe boolean tidak memerlukan tanda kutip. Beberapa nilai String.Array dipisahkan dengan koma.

  Tipe data ini tidak didukung untuk AWS Lambda langganan. Jika Anda menentukan tipe data ini untuk titik akhir Lambda, itu diteruskan sebagai tipe `String` data dalam JSON payload yang SNS dikirimkan Amazon ke Lambda.

- Number (Nomor) — Nomor adalah bilangan bulat positif atau negatif atau bilangan floating-point. Angka memiliki jangkauan dan presisi yang cukup untuk mencakup sebagian besar kemungkinan nilai yang biasanya didukung oleh bilangan bulat, float, dan ganda. Sejumlah dapat memiliki nilai dari -10^9 hingga 10^9, dengan 5 digit akurasi setelah titik desimal. Nol awal dan akhir dipangkas.

Tipe data ini tidak didukung untuk AWS Lambda langganan. Jika Anda menentukan tipe data ini untuk titik akhir Lambda, itu diteruskan sebagai tipe `String` data dalam JSON payload yang SNS dikirimkan Amazon ke Lambda.

- Binary (Biner) — Atribut jenis biner dapat menyimpan data biner apapun; misalnya, data terkompresi, data terenkripsi, atau gambar.

Atribut pesan yang dicadangkan untuk notifikasi push seluler

Tabel berikut mencantumkan atribut pesan dicadangkan untuk layanan notifikasi push seluler yang dapat Anda gunakan untuk struktur pesan notifikasi push Anda:

Layanan notifikasi push	Atribut pesan yang dicadangkan
ADM	`AWS.SNS.MOBILE.ADM.TTL`
APNs[1]	`AWS.SNS.MOBILE.APNS_MDM.TTL`
	`AWS.SNS.MOBILE.APNS_MDM_SANDBOX.TTL`
	`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`

Layanan notifikasi push	Atribut pesan yang dicadangkan
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`
	`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] Apple akan menolak SNS pemberitahuan Amazon jika atribut pesan tidak memenuhi persyaratannya. Untuk detail tambahan, lihat Mengirim Permintaan Pemberitahuan ke APNs situs web Pengembang Apple.

SNSPengelompokan pesan Amazon

Apa itu message batching?

Alternatif untuk menerbitkan pesan ke Standar atau FIFO topik dalam `Publish` API permintaan individual, menggunakan Amazon SNS `PublishBatch` API untuk mempublikasikan hingga 10 pesan dalam satu API permintaan. Mengirim pesan dalam batch dapat membantu Anda mengurangi biaya yang terkait dengan menghubungkan aplikasi terdistribusi (pesan A2A) atau mengirim pemberitahuan kepada orang-orang (pesan A2P) dengan Amazon SNS dengan faktor hingga 10. Amazon SNS memiliki kuota tentang berapa banyak pesan yang dapat Anda terbitkan ke topik per detik berdasarkan wilayah tempat Anda beroperasi. Lihat halaman SNStitik akhir dan kuota Amazon di Referensi Umum AWSpanduan untuk informasi lebih lanjut tentang API kuota.

> **Note**
>
> Ukuran agregat total semua pesan yang Anda kirim dalam satu `PublishBatch` API permintaan tidak dapat melebihi 262.144 byte (256 KB).
> `PublishBatch`APIMenggunakan `Publish` API tindakan yang sama untuk IAM kebijakan.

Bagaimana cara kerja pengelompokan pesan?

Menerbitkan pesan dengan `PublishBatch` API mirip dengan menerbitkan pesan dengan file `Publish`API. Perbedaan utama adalah bahwa setiap pesan dalam `PublishBatch` API permintaan harus diberi ID batch unik (hingga 80 karakter). Dengan cara ini, Amazon SNS dapat mengembalikan API respons individual untuk setiap pesan dalam batch untuk mengonfirmasi bahwa setiap pesan telah diterbitkan atau terjadi kegagalan. Untuk pesan yang dipublikasikan ke FIFO topik, selain menyertakan penetapan ID batch unik, Anda masih perlu menyertakan `MessageDeduplicationID` dan `MessageGroupId` untuk setiap pesan individual.

Contoh

Menerbitkan kumpulan 10 pesan ke suatu FIFO topik

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

Menerbitkan kumpulan 10 pesan ke suatu FIFO topik

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

Menghapus SNS topik dan langganan Amazon

Saat topik dihapus, langganan terkaitnya dihapus secara asinkron. Meskipun pelanggan masih dapat mengakses langganan ini, langganan tidak lagi terkait dengan topik tersebut—bahkan jika Anda membuat ulang topik menggunakan nama yang sama. Jika penayang mencoba mempublikasikan pesan ke topik yang dihapus, penerbit akan menerima pesan kesalahan yang menunjukkan bahwa topik tersebut tidak ada. Demikian pula, setiap upaya untuk berlangganan topik yang dihapus juga akan menghasilkan pesan kesalahan. Anda tidak dapat menghapus langganan yang menunggu konfirmasi. Amazon SNS secara otomatis menghapus langganan yang belum dikonfirmasi setelah 48 jam.

Topik

- Untuk menghapus SNS topik atau langganan Amazon menggunakan AWS Management Console

- Untuk menghapus langganan dan topik menggunakan AWS SDK

Untuk menghapus SNS topik atau langganan Amazon menggunakan AWS Management Console

Menghapus SNS topik atau langganan Amazon memastikan pengelolaan sumber daya yang efisien, mencegah penggunaan yang tidak perlu, dan menjaga SNS konsol Amazon tetap teratur. Langkah ini membantu menghindari potensi biaya dari sumber daya yang tidak digunakan dan merampingkan administrasi dengan menghapus topik atau langganan yang tidak lagi diperlukan.

Untuk menghapus topik menggunakan AWS Management Console

1. Masuk ke SNSkonsol Amazon.
2. Di panel navigasi kiri, pilih Topics (Topik).
3. Di halaman Topics (Topik), pilih topik, lalu pilih Delete (Hapus).
4. Di kotak dialog Delete topic (Hapus topik), masukkan `delete me`, lalu pilih Delete (Hapus).

   Konsol tersebut menghapus topik.

Untuk menghapus langganan menggunakan AWS Management Console

1. Masuk ke SNSkonsol Amazon.
2. Di panel navigasi kiri, pilih Subscriptions (Langganan).
3. Pada halaman Langganan, pilih langganan dengan status Dikonfirmasi, lalu pilih Hapus.
4. Di kotak dialog Delete subscription (Hapus langganan), pilih Delete (Hapus).

   Konsol tersebut menghapus langganan.

Untuk menghapus langganan dan topik menggunakan AWS SDK

Untuk menggunakan AWS SDK, Anda harus mengkonfigurasinya dengan kredensi Anda. Untuk informasi selengkapnya, lihat File konfigurasi dan kredensial bersama di Panduan Referensi Alat AWS SDKs dan Alat.

Contoh kode berikut menunjukkan cara menggunakan`DeleteTopic`.

.NET

AWS SDK for .NET

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

Hapus topik berdasarkan topiknyaARN.

```
/// <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;
}
```

- Untuk API detailnya, lihat DeleteTopicdi AWS SDK for .NET APIReferensi.

C++

SDKuntuk C ++

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

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

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

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

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

    return outcome.IsSuccess();
}
```

- Untuk API detailnya, lihat DeleteTopicdi AWS SDK for C++ APIReferensi.

CLI

AWS CLI

Untuk menghapus SNS topik

`delete-topic`Contoh berikut menghapus SNS topik yang ditentukan.

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

Perintah ini tidak menghasilkan output.

- Untuk API detailnya, lihat DeleteTopicdi Referensi AWS CLI Perintah.

Go

SDKuntuk Go V2

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

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(ctx context.Context, topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(ctx, &sns.DeleteTopicInput{
  TopicArn: aws.String(topicArn)})
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}
```

- Untuk API detailnya, lihat DeleteTopicdi AWS SDK for Go APIReferensi.

Java

SDKuntuk Java 2.x

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

```
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);
        }
    }
}
```

- Untuk API detailnya, lihat DeleteTopicdi AWS SDK for Java 2.x APIReferensi.

JavaScript

SDKuntuk JavaScript (v3)

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

Buat klien dalam modul terpisah dan ekspor klien tersebut.

```
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({});
```

Impor modul SDK dan klien dan panggil fileAPI.

```
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
  //   }
  // }
};
```

- Untuk informasi selengkapnya, lihat AWS SDK for JavaScript Panduan Developer.
- Untuk API detailnya, lihat DeleteTopicdi AWS SDK for JavaScript APIReferensi.

Kotlin

SDKuntuk Kotlin

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

```
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.")
    }
}
```

- Untuk API detailnya, lihat DeleteTopic AWSSDKAPIreferensi Kotlin.

PHP

SDKuntuk PHP

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

```
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());
}
```

- Untuk API detailnya, lihat DeleteTopicdi AWS SDK for PHP APIReferensi.

Python

SDKuntuk Python (Boto3)

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

```
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
```

- Untuk API detailnya, lihat DeleteTopic AWSSDKReferensi Python (Boto3). API

SAP ABAP

SDKuntuk SAP ABAP

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

```
    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.
```

- Untuk API detailnya, lihat DeleteTopic AWSSDKuntuk SAP ABAP API referensi.

Langkah selanjutnya

Sekarang setelah Anda membuat topik dengan langganan dan mengirim pesan ke topik tersebut, Anda mungkin ingin mencoba hal berikut:

- Jelajahi AWS Pusat Developer.
- Pelajari tentang melindungi data Anda di bagian Keamanan.
- Aktifkan Enkripsi sisi server untuk topik.
- Aktifkan enkripsi sisi server untuk topik dengan antrean Amazon Simple Queue Service (Amazon) terenkripsi yang dilanggan. SQS
- Berlangganan AWS Event Fork Pipelines ke suatu topik.

Strategi pemesanan dan deduplikasi pesan menggunakan topik Amazon SNS FIFO

Topik ini memberikan informasi tentang fitur dan fungsionalitas topik Amazon SNS FIFO (First-In-First-Out) dan bagaimana mereka berintegrasi dengan antrian Amazon. SQS FIFO Anda akan belajar cara menggunakan layanan ini bersama-sama untuk memastikan pemesanan dan deduplikasi pesan yang ketat, penting untuk aplikasi yang memerlukan konsistensi data. Konten ini mencakup kasus penggunaan spesifik di mana SNS FIFO topik Amazon bermanfaat, memberikan wawasan tentang skenario di mana urutan pesan dan keunikan sangat penting.

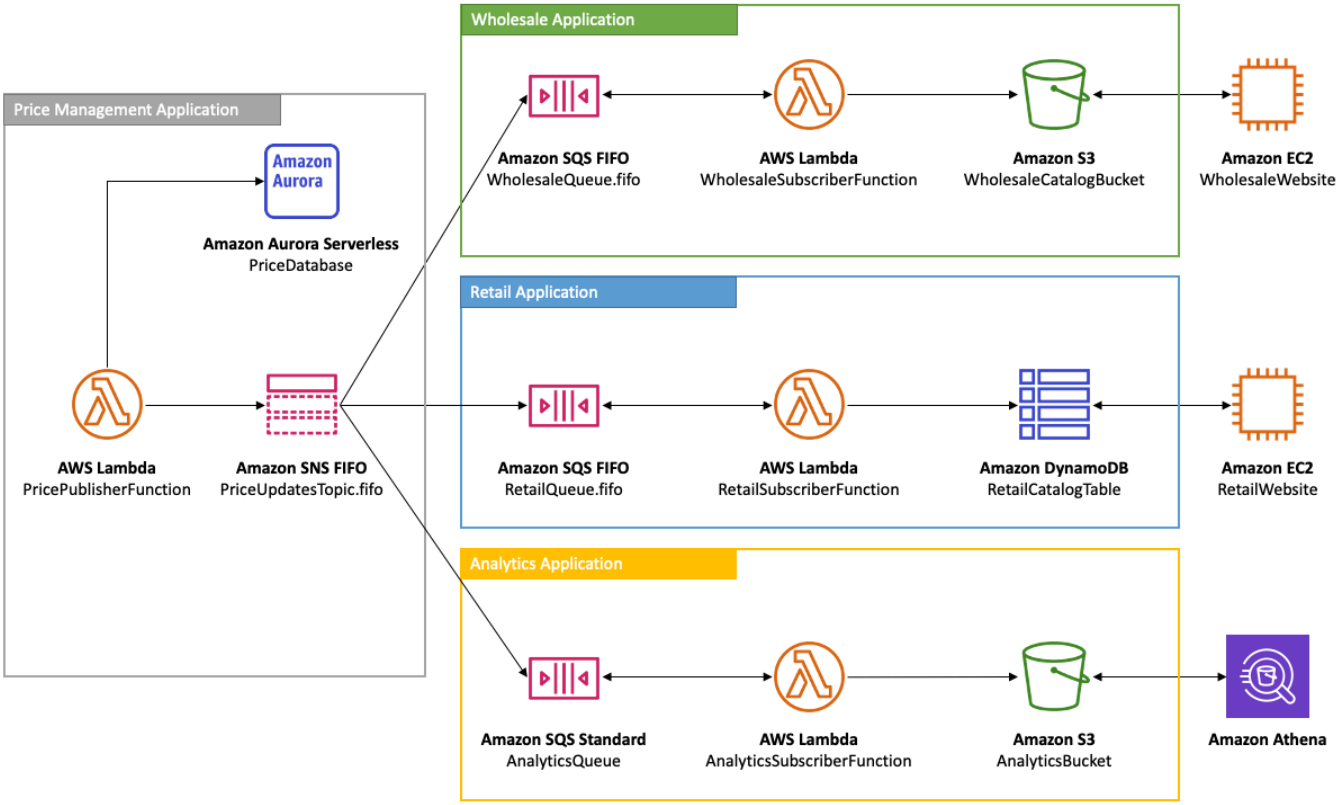
Anda juga akan mempelajari detail teknis pemesanan pesan, pengelompokan pesan, dan bagaimana hal ini memengaruhi pengiriman pesan. Topik deduplikasi pesan menjelaskan mekanisme yang mencegah duplikat pesan, memastikan bahwa setiap pesan diproses hanya sekali. Selain itu, Anda akan belajar tentang penyaringan pesan, keamanan, dan daya tahan, yang penting untuk menjaga integritas dan keandalan sistem pesan Anda. Konten ini juga mencakup informasi tentang pengarsipan dan pemutaran ulang pesan, menawarkan strategi untuk mengelola riwayat pesan. Contoh kode praktis juga disediakan untuk membantu Anda menerapkan fitur-fitur ini di aplikasi Anda sendiri, memberi Anda pengalaman langsung dengan SNS FIFO topik Amazon dan integrasinya dengan antrian Amazon SQSFIFO.

Contoh kasus penggunaan SNS FIFO topik Amazon

Contoh berikut menjelaskan platform e-niaga yang dibangun oleh produsen suku cadang mobil menggunakan SNS FIFO topik Amazon dan SQS antrian Amazon. Platform ini terdiri dari empat aplikasi tanpa server:

- Manajer inventaris menggunakan aplikasi manajemen harga untuk mengatur harga untuk setiap item dalam stok. Di perusahaan ini, harga produk dapat berubah berdasarkan fluktuasi pertukaran mata uang, permintaan pasar, dan pergeseran strategi penjualan. Aplikasi manajemen harga menggunakan AWS Lambda fungsi yang menerbitkan pembaruan harga ke SNS FIFO topik Amazon setiap kali harga berubah.
- Aplikasi grosir menyediakan backend untuk situs web tempat toko bodi mobil dan produsen mobil dapat membeli suku cadang mobil perusahaan dalam jumlah besar. Untuk mendapatkan pemberitahuan perubahan harga, aplikasi grosir berlangganan SQS FIFO antrian Amazon ke topik Amazon SNS FIFO aplikasi manajemen harga.

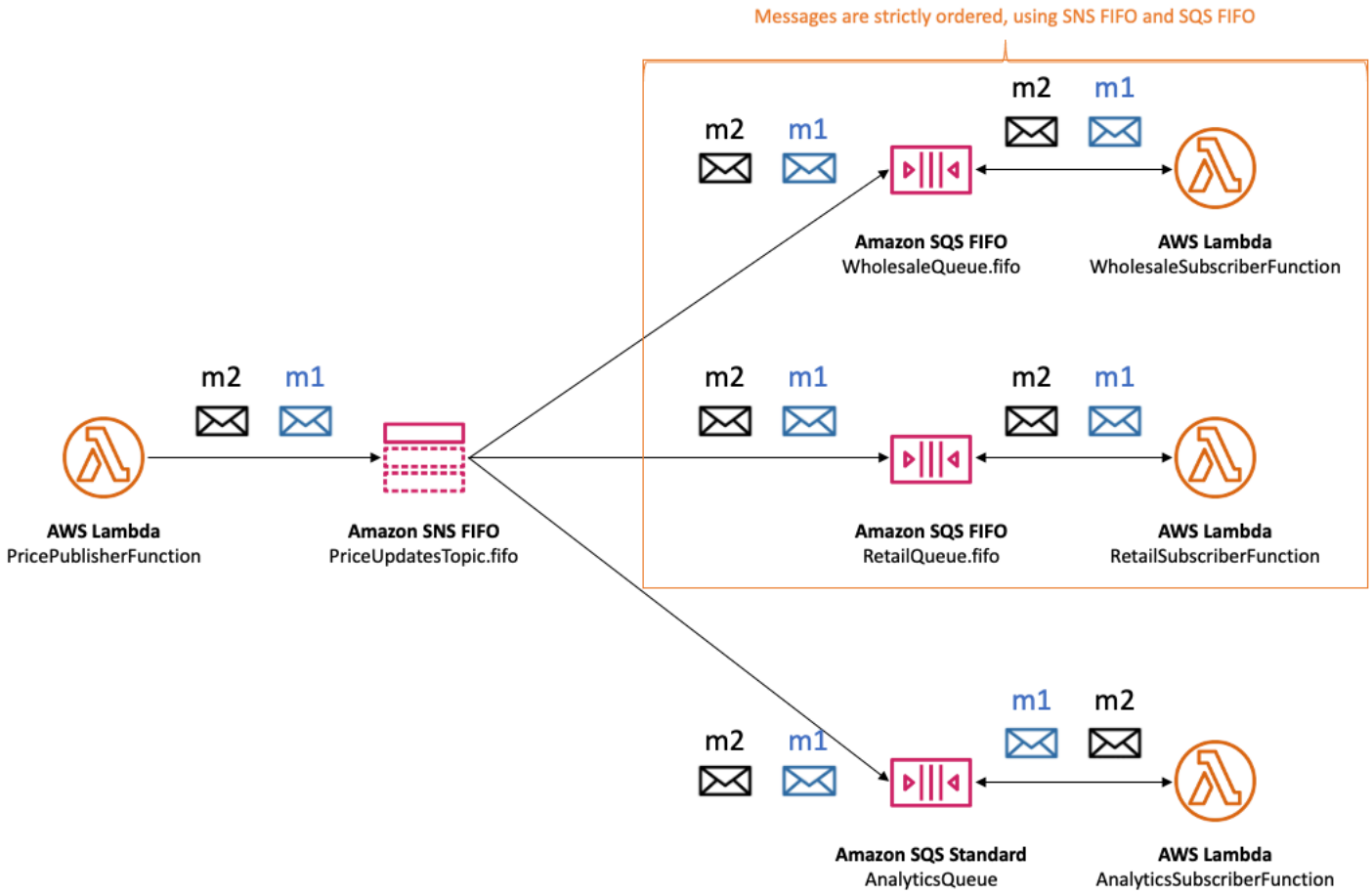
- Sebuah aplikasi ritel menyediakan backend untuk situs web lain tempat pemilik mobil dan penggemar penyetelan mobil dapat membeli suku cadang mobil individu untuk kendaraan mereka. Untuk mendapatkan pemberitahuan perubahan harga, aplikasi ritel juga berlangganan SQS FIFO antrian Amazon ke topik Amazon SNS FIFO aplikasi manajemen harga.
- Aplikasi analitik yang menggabungkan pembaruan harga dan menyimpannya ke dalam bucket Amazon S3, memungkinkan Amazon Athena untuk menanyakan bucket untuk tujuan intelijen bisnis (BI). Untuk mendapatkan pemberitahuan perubahan harga, aplikasi analitik berlangganan antrian SQS standar Amazon ke topik Amazon SNS FIFO aplikasi manajemen harga. Berbeda dengan aplikasi lain, analitik tidak memerlukan pembaruan harga untuk dipesan secara ketat.



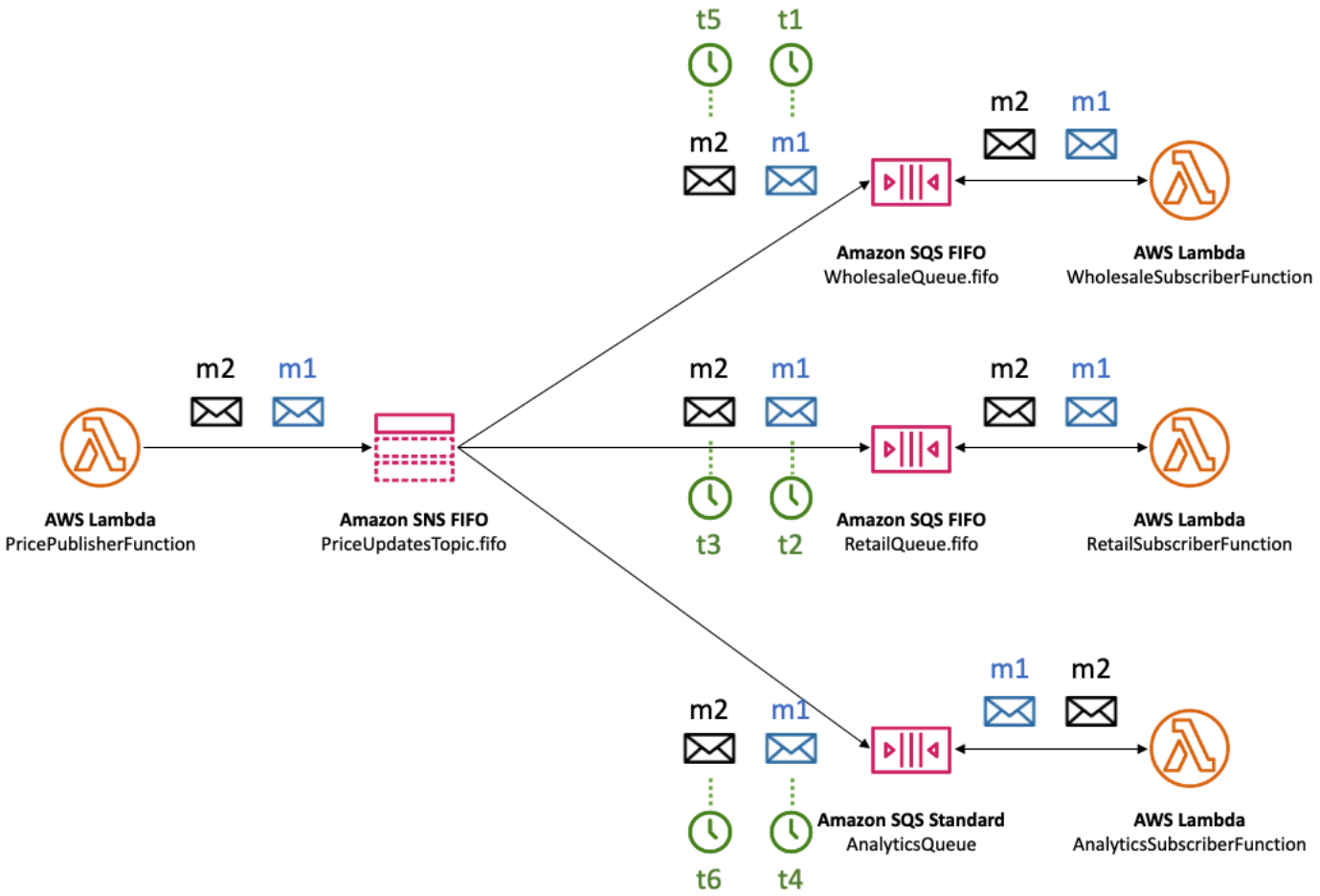
Agar aplikasi grosir dan ritel menerima pembaruan harga dalam urutan yang benar, aplikasi manajemen harga harus menggunakan sistem distribusi pesan yang diurutkan secara ketat. Menggunakan SNS FIFO topik Amazon dan SQS FIFO antrian Amazon memungkinkan pemrosesan pesan secara berurutan dan tanpa duplikasi. Untuk informasi selengkapnya, lihat Detail pemesanan SNS pesan Amazon untuk FIFO topik. Untuk cuplikan kode yang menerapkan kasus penggunaan ini, lihat Contoh SNS kode Amazon untuk FIFO topik.

Detail pemesanan SNS pesan Amazon untuk FIFO topik

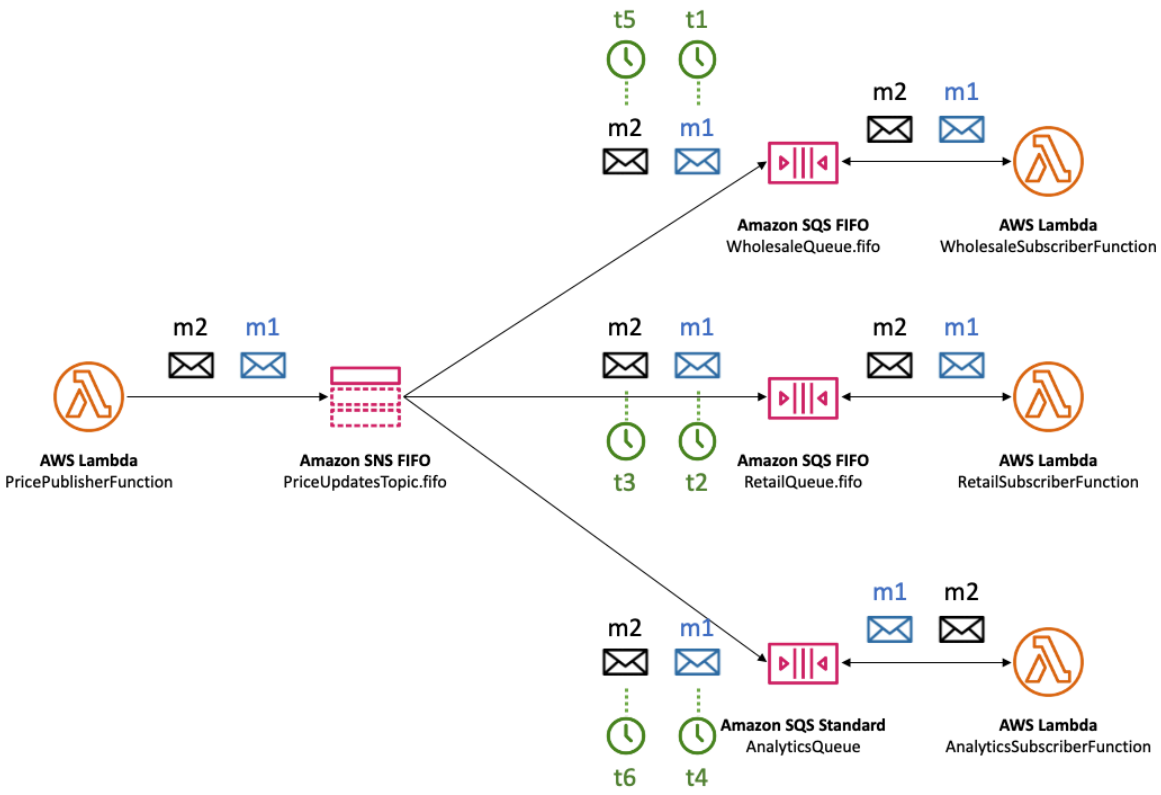
SNSFIFOTopik Amazon selalu mengirimkan pesan ke SQS antrian Amazon berlangganan dalam urutan yang tepat di mana pesan dipublikasikan ke topik, dan hanya sekali. Dengan SQS FIFO antrian Amazon berlangganan, konsumen antrian menerima pesan dalam urutan yang tepat di mana pesan dikirim ke antrian, dan tidak ada duplikat. Namun, dengan antrian SQS standar Amazon yang berlangganan, konsumen antrian dapat menerima pesan yang rusak, dan lebih dari sekali. Hal ini memungkinkan pemisahan pelanggan lebih lanjut dari penerbit, memberikan pelanggan lebih banyak fleksibilitas dalam hal konsumsi pesan dan pengoptimalan biaya, seperti yang ditunjukkan pada diagram berikut, berdasarkan pada. Contoh kasus penggunaan SNS FIFO topik Amazon



Perhatikan bahwa tidak ada pemesanan tersirat dari pelanggan. Contoh berikut menunjukkan bahwa pesan m1 dikirim pertama ke pelanggan grosir dan kemudian ke pelanggan ritel dan kemudian ke pelanggan analitik. Pesan m2 dikirim pertama ke pelanggan ritel dan kemudian ke pelanggan grosir dan akhirnya ke pelanggan analitik. Meskipun kedua pesan dikirim ke pelanggan dalam urutan yang berbeda, pemesanan pesan dipertahankan untuk setiap SQS FIFO pelanggan Amazon. Setiap pelanggan dianggap terpisah dari pelanggan lain.

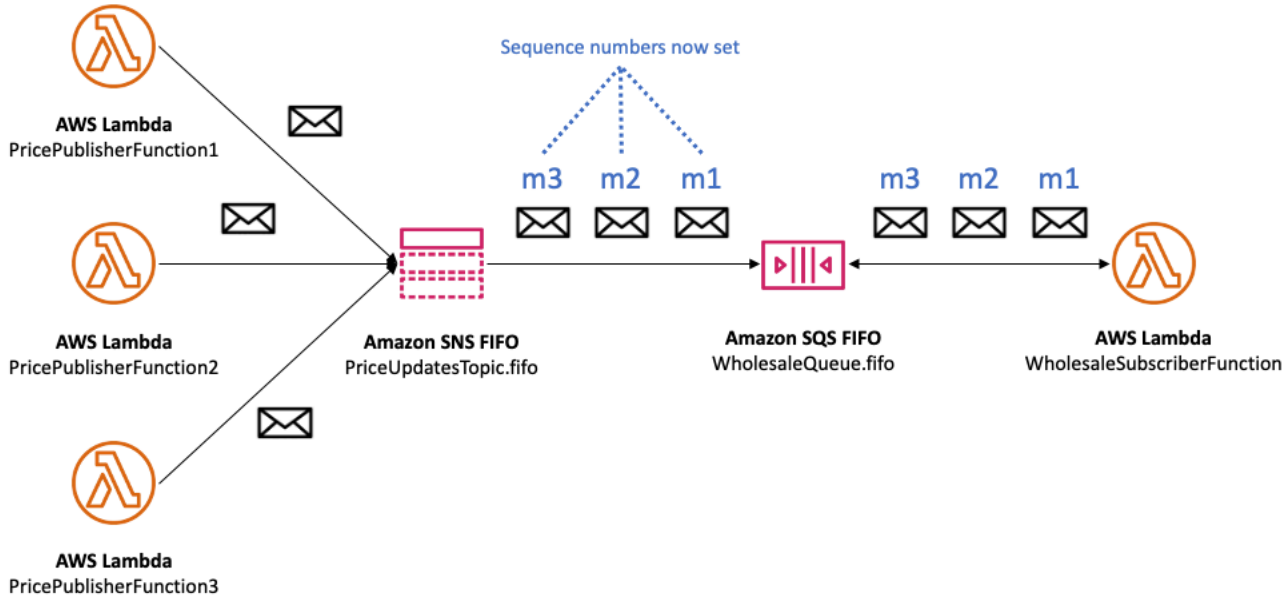


Jika pelanggan SQS antrian Amazon menjadi tidak dapat dijangkau, itu bisa keluar dari sinkron. Misalnya, pemilik antrian aplikasi grosir secara keliru mengubah kebijakan SQS antrian Amazon dengan cara yang mencegah kepala SNS layanan Amazon mengirimkan pesan ke antrian. Dalam hal ini, pengiriman pembaruan harga ke antrian grosir gagal, sedangkan antrian ritel dan analitik berhasil, menyebabkan pelanggan tidak sinkron. Ketika pemilik antrian aplikasi grosir mengoreksi kebijakan SNS antreannya, Amazon melanjutkan pengiriman pesan ke antrian berlangganan. Setiap pesan yang dipublikasikan ke topik yang menargetkan antrian yang tidak dikonfigurasi dengan benar akan dihapus, kecuali langganan terkait memiliki antrian huruf mati yang dikonfigurasi.



Anda dapat memiliki beberapa aplikasi (atau beberapa utas dalam aplikasi yang sama) menerbitkan pesan ke SNS FIFO topik secara paralel. Ketika Anda melakukan ini, Anda secara efektif mendelegasikan pengurutan pesan ke layanan AmazonSNS. Untuk menentukan urutan pesan yang telah ditetapkan, Anda dapat memeriksa nomor urutannya.

Nomor urut adalah angka besar dan tidak berurutan yang diberikan Amazon untuk setiap SNS pesan. Panjang nomor urut adalah 128-bit, dan terus meningkat untuk setiap Grup Pesan. Nomor urut diteruskan ke SQS antrian Amazon berlangganan sebagai bagian dari badan pesan. Namun, jika Anda mengaktifkan pengiriman pesan mentah, pesan yang dikirim ke SQS antrian Amazon tidak menyertakan nomor urut atau metadata SNS pesan Amazon lainnya.



SNSFIFOTopik Amazon menentukan urutan dalam konteks grup pesan. Untuk informasi selengkapnya, lihat Pengelompokan SNS pesan Amazon untuk topik FIFO.

Pengelompokan SNS pesan Amazon untuk topik FIFO

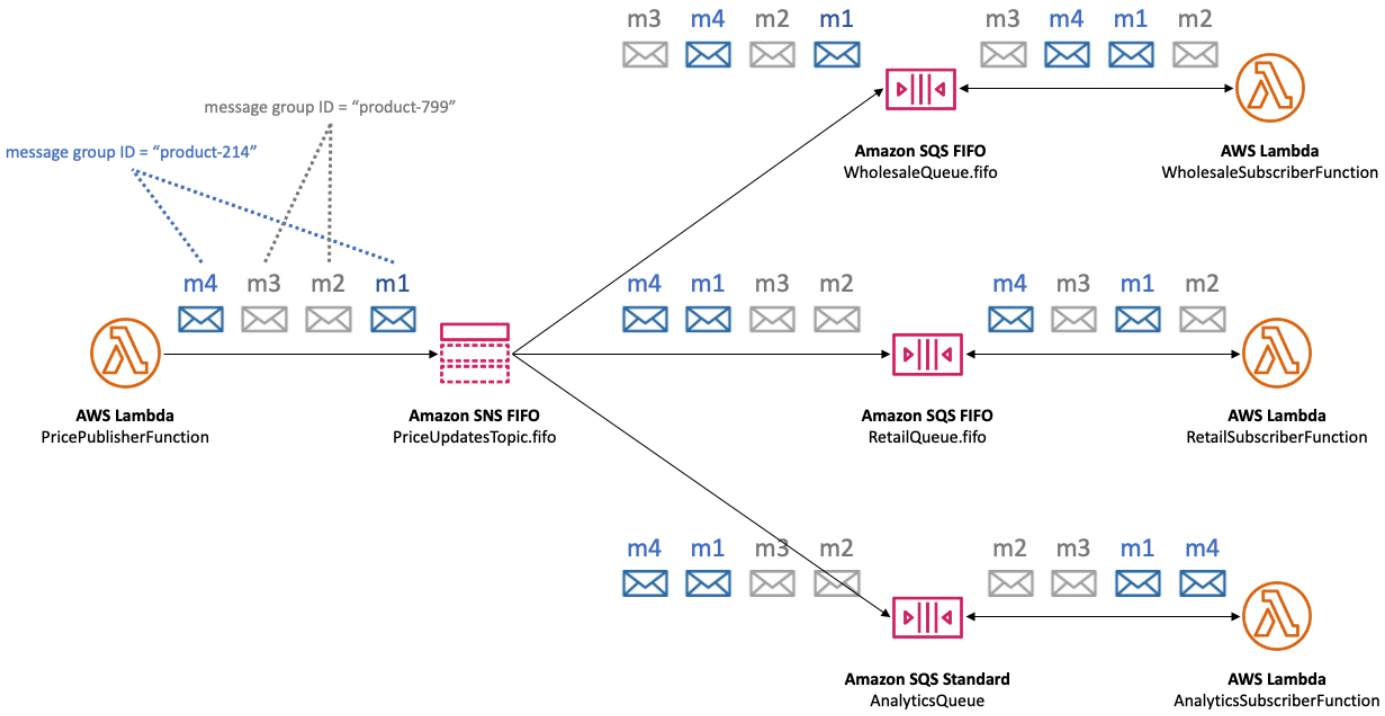
Pesan yang termasuk dalam grup yang sama diproses satu per satu, dalam urutan yang ketat relatif terhadap grup.

Saat memublikasikan pesan ke SNS FIFO topik Amazon, Anda menyetel ID grup pesan. ID grup adalah token wajib yang menentukan bahwa pesan milik grup pesan tertentu. SNSFIFOTopik meneruskan ID grup ke SQS FIFO antrian Amazon berlangganan. Tidak ada batasan jumlah grup IDs dalam SNS FIFO topik atau SQS FIFO antrian. ID grup pesan tidak diteruskan ke SQS antrian standar Amazon.

Tidak ada afinitas antara grup pesan dan langganan. Oleh karena itu, pesan yang dipublikasikan ke grup pesan akan dikirim ke semua antrian langganan, tunduk pada kebijakan filter yang dilampirkan ke langganan. Untuk informasi lebih lanjut, lihat Pengiriman SNS pesan Amazon untuk FIFO topik dan Pemfilteran SNS pesan Amazon untuk topik FIFO.

Di bagian auto manajemen harga contoh kasus penggunaan, ada grup pesan khusus untuk setiap produk yang dijual di platform. SNSFIFOTopik Amazon yang sama digunakan untuk memproses semua pembaruan harga. Urutan pembaruan harga dipertahankan dalam konteks produk suku cadang mobil tunggal, namun tidak di berbagai produk. Diagram berikut menunjukkan bagaimana inii bekerja. Perhatikan bahwa, untuk produk yang ID grup pesannya adalah product-214, pesan

m4 diproses sebelum pesan m1. Urutan ini dipertahankan di seluruh alur kerja yang menggunakan Amazon SNS FIFO ke Amazon SQSFIFO. Demikian juga, untuk produk yang ID grup pesannya adalah product-799, pesan m3 diproses sebelum pesan m2, selama alur kerjanya menggunakan Amazon dan Amazon. SNS FIFO SQS FIFO Namun, saat menggunakan antrian SQS standar Amazon, urutan pesan tidak lagi dijamin dan grup pesan tidak ada. Grup pesan produk-214 dan produk-799 terpisah satu sama lain, sehingga tidak ada hubungan antara urutan pesan mereka.



Mendistribusikan data dengan grup pesan IDs untuk meningkatkan kinerja

Untuk mengoptimalkan throughput pengiriman, SNS FIFO topik Amazon mengirimkan pesan dari grup pesan yang berbeda secara paralel, sementara urutan pesan dijaga ketat dalam setiap grup pesan. Setiap grup pesan individu dapat mengirimkan maksimal 300 pesan per detik. Oleh karena itu, untuk mencapai throughput tinggi untuk satu topik, gunakan sejumlah besar grup IDs pesan yang berbeda. Dengan memanfaatkan kumpulan grup pesan yang beragam, SNS FIFO topik Amazon secara otomatis mendistribusikan pesan di sejumlah besar partisi paralel.

> **Note**
>
> SNSFIFOTopik Amazon dioptimalkan untuk distribusi pesan yang seragam di seluruh grup pesanIDs, terlepas dari jumlah grup. AWS merekomendasikan agar Anda menggunakan sejumlah besar grup pesan yang berbeda IDs untuk kinerja yang dioptimalkan.

Saat memublikasikan ke SNS FIFO topik Amazon Anda dengan throughput tinggi dan satu atau lebih SQS FIFO antrian Amazon berlangganan, Anda disarankan untuk mengaktifkan throughput tinggi pada antrian Anda. Untuk selengkapnya lihat Throughput tinggi untuk FIFO antrian di Panduan Pengembang Layanan Antrian Sederhana Amazon.

Pengiriman SNS pesan Amazon untuk FIFO topik

Topik Amazon SNS FIFO (pertama masuk, keluar pertama) mendukung pengiriman ke SQS standar Amazon dan FIFO antrian untuk memberikan fleksibilitas dan kontrol kepada pelanggan saat mengintegrasikan aplikasi terdistribusi yang memerlukan konsistensi data dalam waktu dekat.

Untuk beban kerja yang perlu mempertahankan urutan pesan atau de-duplikasi yang ketat, kombinasi topik Amazon SNS FIFO dengan SQSFIFOantrian Amazon berlangganan sebagai titik akhir pengiriman memberikan peningkatan pesan antar aplikasi saat urutan operasi dan peristiwa sangat penting, atau di mana duplikat tidak dapat ditoleransi.

Untuk beban kerja yang mentolerir pemesanan dan at-least-once pengiriman upaya terbaik, berlangganan antrian standar SQS Amazon ke topik SNS FIFO Amazon memberikan kemampuan untuk menurunkan biaya, selain berbagi antrian di seluruh beban kerja yang tidak digunakan. FIFO

> **Note**
>
> Untuk menyebarkan pesan dari SNS FIFO topik Amazon ke AWS Lambda fungsi, diperlukan langkah-langkah tambahan. Pertama, berlangganan Amazon SQS FIFO atau antrian standar ke topik. Kemudian konfigurasikan antrean untuk memicu fungsi. Untuk informasi lebih lanjut, lihat posting SQSFIFOsebagai sumber acara di AWS Compute Blog.

SNSFIFOtopik tidak dapat mengirimkan pesan ke titik akhir yang dikelola pelanggan, seperti alamat email, aplikasi seluler, nomor telepon untuk pesan teks (SMS), atau titik akhir HTTP (S). Jenis

titik akhir ini tidak dijamin untuk mempertahankan pengurutan pesan yang ketat. Upaya untuk berlangganan titik akhir yang dikelola pelanggan ke SNS FIFO topik menghasilkan kesalahan.

SNSFIFOtopik mendukung kemampuan penyaringan pesan yang sama dengan topik standar. Untuk informasi selengkapnya, lihat Pemfilteran SNS pesan Amazon untuk topik FIFO dan posting Sederhanakan Pesan Pub/Sub Anda dengan Pemfilteran SNS Pesan Amazon di Blog Komputasi.AWS

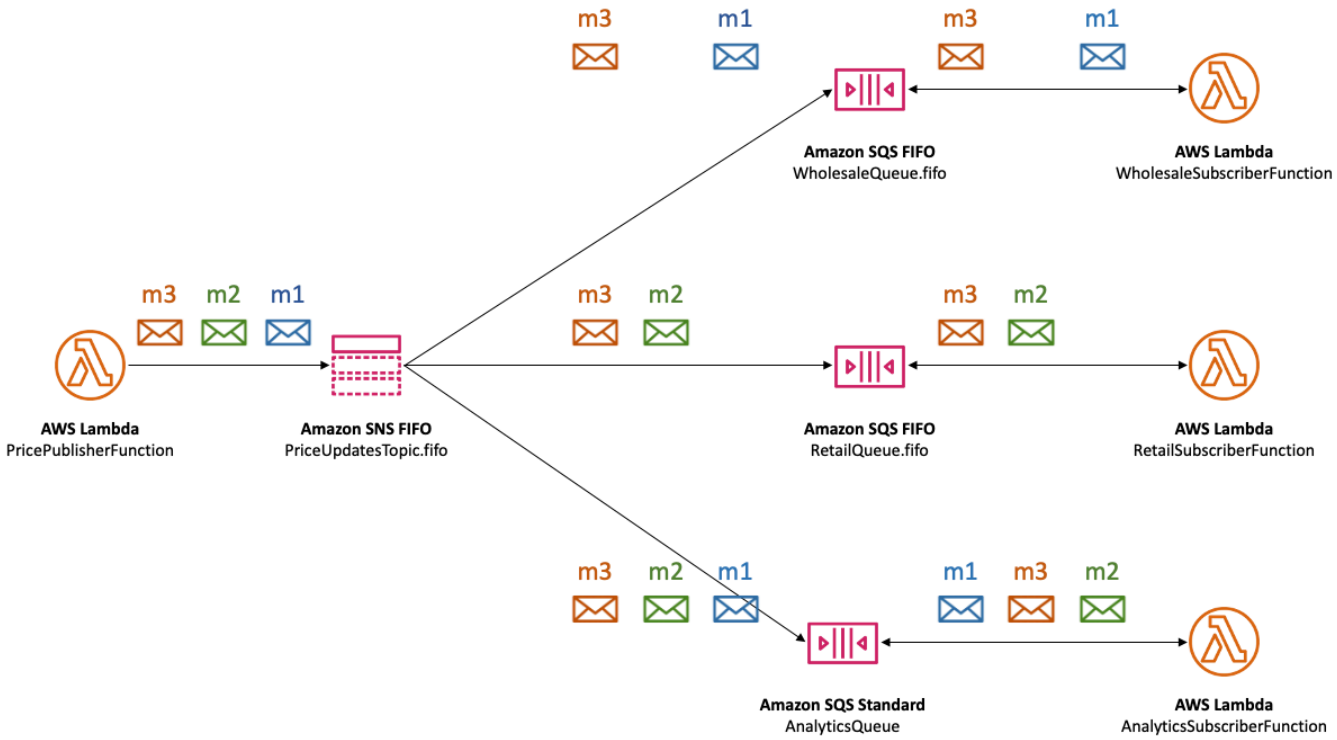
Pemfilteran SNS pesan Amazon untuk topik FIFO

SNSFIFOTopik Amazon mendukung pemfilteran pesan. Menggunakan pemfilteran pesan menyederhanakan arsitektur Anda dengan membongkar logika perutean pesan dari sistem penerbit Anda dan logika pemfilteran pesan dari sistem pelanggan Anda.

Saat Anda berlangganan Amazon SQS FIFO atau antrian standar ke suatu SNS FIFO topik, Anda dapat menggunakan pemfilteran pesan untuk menentukan bahwa pelanggan menerima subset pesan, bukan semuanya. Setiap pelanggan dapat menetapkan kebijakan filternya sendiri sebagai atribut langganan. Berdasarkan cakupan kebijakan filter, kebijakan filter dicocokkan dengan atribut pesan atau isi pesan yang masuk. Jika kebijakan filter cocok, topik akan mengirimkan salinan pesan ke pelanggan. Jika tidak ada kecocokan, topik tidak akan mengirimkan salinan pesan.

Dalam kasus penggunaan contoh manajemen harga suku cadang mobil, asumsikan bahwa kebijakan SNS filter Amazon berikut ditetapkan dan cakupan kebijakan filter adalah`MessageBody`:

- Untuk antrian grosir, kebijakan filter `{"business":["wholesale"]}` cocok dengan setiap pesan yang berisi kunci bernama `business` dan dengan `wholesale` dalam kumpulan nilai. Dalam diagram berikut, salah satu kunci dalam pesan m1 adalah `business` dengan nilai. `wholesale` Salah satu kunci dalam pesan m3 adalah `business` dengan nilai. `["wholesale,retail"]` Dengan demikian, m1 dan m3 sesuai dengan kriteria kebijakan filter, dan kedua pesan tersebut dikirim ke antrean grosir.
- Untuk antrian ritel, kebijakan filter `{"business":["retail"]}` cocok dengan setiap pesan yang berisi kunci bernama `business` dan dengan `retail` dalam kumpulan nilai. Dalam diagram, salah satu kunci dalam pesan m2 adalah `business` dengan nilai`retail`. Salah satu kunci dalam pesan m3 adalah `business` dengan nilai. `["wholesale,retail"]` Dengan demikian, m2 dan m3 sesuai dengan kriteria kebijakan filter, dan kedua pesan tersebut dikirim ke antrean ritel.
- Untuk antrian analitik, kami ingin Amazon Athena menerima semua catatan, jadi tidak ada kebijakan filter yang diterapkan.



SNSFIFOtopik mendukung berbagai operator yang cocok, termasuk nilai string atribut, nilai numerik atribut, dan kunci atribut. Untuk informasi selengkapnya, lihat SNSPemfilteran pesan Amazon.

SNSFIFOtopik tidak mengirimkan pesan duplikat ke titik akhir berlangganan. Untuk informasi selengkapnya, lihat Deduplikasi SNS pesan Amazon untuk topik FIFO.

Deduplikasi SNS pesan Amazon untuk topik FIFO

SNSFIFOTopik Amazon dan SQS FIFO antrian Amazon mendukung deduplikasi pesan, yang menyediakan pengiriman dan pemrosesan pesan tepat sekali selama kondisi berikut terpenuhi:

- SQSFIFOAntrian Amazon berlangganan ada dan memiliki izin yang memungkinkan kepala SNS layanan Amazon untuk mengirimkan pesan ke antrian.
- Konsumen SQS FIFO antrian Amazon memproses pesan dan menghapusnya dari antrian sebelum batas waktu visibilitas berakhir.
- Topik SNS berlangganan Amazon tidak memiliki pemfilteran pesan. Saat Anda mengonfigurasi pemfilteran pesan, SNS FIFO topik Amazon mendukung at-most-once pengiriman, karena pesan dapat disaring berdasarkan kebijakan filter langganan Anda.
- Tidak ada gangguan jaringan yang mencegah perizinan pengiriman pesan.

> **Note**
>
> Deduplikasi pesan berlaku untuk seluruh SNS FIFO topik Amazon, bukan untuk grup pesan individual.

Saat Anda memublikasikan pesan ke SNS FIFO topik Amazon, pesan tersebut harus menyertakan ID deduplikasi. ID ini disertakan dalam pesan yang dikirimkan SNS FIFO topik Amazon ke antrian Amazon SQS FIFO yang berlangganan.

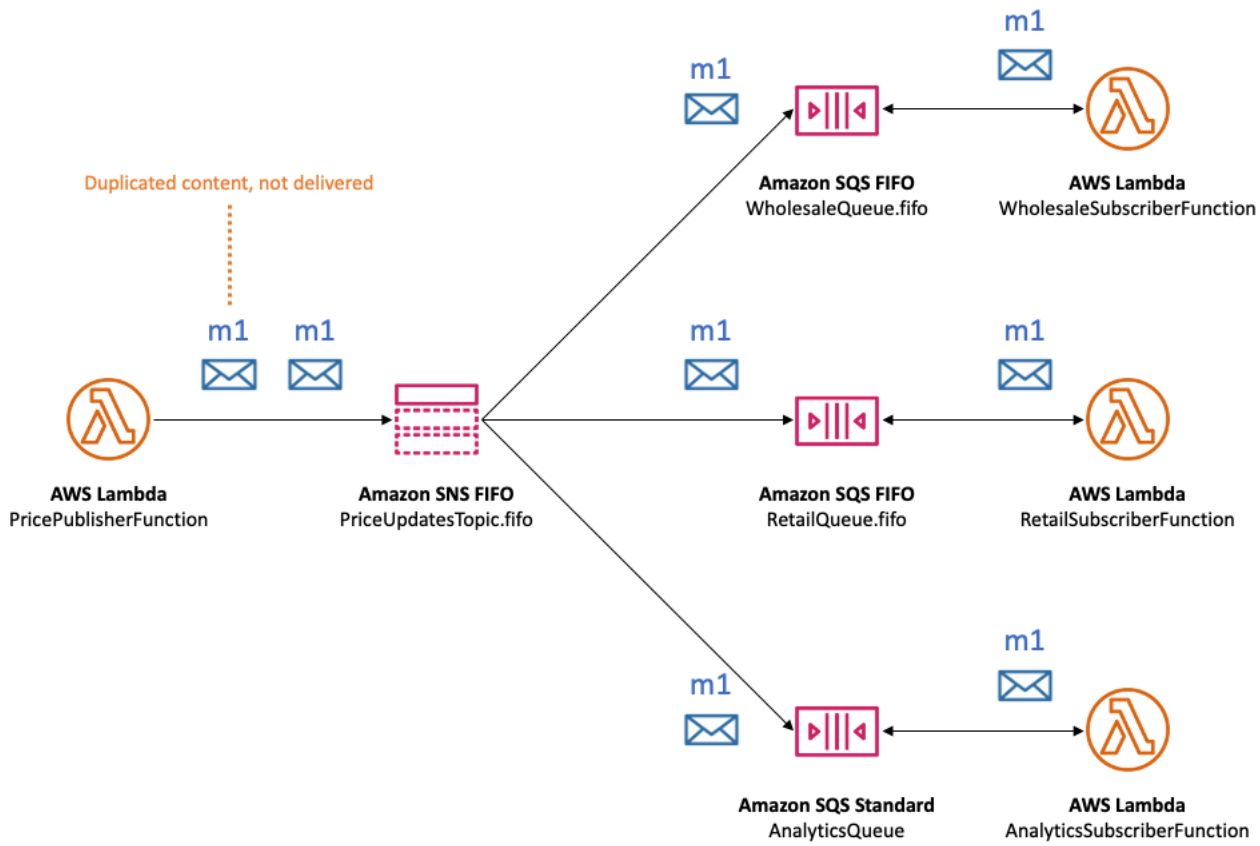
Jika pesan dengan ID deduplikasi tertentu berhasil dipublikasikan ke SNS FIFO topik Amazon, pesan apa pun yang diterbitkan dengan ID deduplikasi yang sama, dalam interval deduplikasi lima menit, diterima tetapi tidak dikirim. SNSFIFOTopik Amazon terus melacak ID deduplikasi pesan, bahkan setelah pesan dikirim ke titik akhir berlangganan.

Jika badan pesan dijamin unik untuk setiap pesan yang dipublikasikan, Anda dapat mengaktifkan deduplikasi berbasis konten untuk SNS FIFO topik Amazon dan antrian Amazon berlangganan. SQS FIFO Amazon SNS menggunakan badan pesan untuk menghasilkan nilai hash unik untuk digunakan sebagai ID deduplikasi untuk setiap pesan, jadi Anda tidak perlu menyetel ID deduplikasi saat mengirim setiap pesan.

> **Note**
>
> Atribut pesan tidak termasuk dalam perhitungan hash.

Saat deduplikasi berbasis konten diaktifkan untuk SNS FIFO topik Amazon, dan pesan dipublikasikan dengan ID deduplikasi, ID deduplikasi yang dipublikasikan akan mengganti ID deduplikasi berbasis konten yang dihasilkan.

Di contoh kasus penggunaan manajemen harga suku cadang mobil, perusahaan harus mengatur ID deduplikasi yang unik secara universal untuk setiap pembaruan harga. Hal ini karena isi pesan bisa jadi identik bahkan ketika atribut pesan berbeda untuk grosir dan ritel. Namun, jika perusahaan menambahkan jenis bisnis (grosir atau eceran) ke badan pesan di samping ID produk dan harga produk, mereka dapat mengaktifkan duplikasi berbasis konten dalam SNS FIFO topik Amazon dan antrian Amazon berlangganan. SQS FIFO



Selain pemesanan pesan dan deduplikasi, SNS FIFO topik Amazon mendukung enkripsi sisi server pesan (SSE) dengan AWS KMS kunci, dan privasi pesan melalui titik akhir dengan. VPC AWS PrivateLink Untuk informasi selengkapnya, lihat Keamanan SNS pesan Amazon untuk FIFO topik.

Keamanan SNS pesan Amazon untuk FIFO topik

Anda dapat memilih agar Amazon SNS dan Amazon SQS mengenkripsi pesan yang dikirim ke FIFO topik dan antrian, menggunakan AWS Key Management Service (AWS KMS) kunci master pelanggan (). CMKs Anda dapat membuat FIFO topik dan antrian terenkripsi, atau memilih untuk mengenkripsi topik dan antrian yang ada. FIFO Amazon SNS dan Amazon hanya SQS mengenkripsi isi pesan. Mereka tidak mengenkripsi atribut pesan, metadata sumber daya, atau metrik sumber daya.

> **Note**
>
> Menambahkan enkripsi ke FIFO topik atau antrian yang ada tidak mengenkripsi pesan yang di-backlog, dan menghapus enkripsi dari topik atau antrian membuat pesan backlog dienkripsi.

SNSFIFOtopik mendekripsi pesan segera sebelum mengirimkannya ke titik akhir berlangganan. SQSFIFOantrian mendekripsi pesan sebelum mengembalikannya ke aplikasi konsumen. Untuk informasi selengkapnya, lihat Enkripsi SNS data Amazon dan Enkripsi pesan yang dipublikasikan ke Amazon SNS dengan AWS KMS posting di AWS Compute Blog.

Selain itu, SNS FIFO topik dan SQS FIFO antrian mendukung privasi pesan dengan VPCtitik akhir antarmuka yang didukung oleh. AWS PrivateLink Menggunakan titik akhir antarmuka, Anda dapat mengirim pesan dari subnet Amazon Virtual Private Cloud (AmazonVPC) ke FIFO topik dan antrian tanpa melintasi internet publik. Model ini menyimpan pesan Anda dalam AWS infrastruktur dan jaringan, yang meningkatkan keamanan keseluruhan aplikasi Anda. Saat Anda menggunakan AWS PrivateLink, Anda tidak perlu menyiapkan gateway internet, terjemahan alamat jaringan (NAT), atau jaringan pribadi virtual (VPN). Untuk informasi selengkapnya, lihat Mengamankan SNS lalu lintas Amazon dengan titik akhir VPC dan Mengamankan pesan yang dipublikasikan ke Amazon SNS dengan AWS PrivateLink posting di Blog AWS Keamanan.

SNSFIFOtopik juga mendukung antrian surat mati dan penyimpanan pesan di seluruh Availability Zone. Untuk informasi selengkapnya, lihat Daya tahan SNS pesan Amazon untuk FIFO topik.

Daya tahan SNS pesan Amazon untuk FIFO topik

SNSFIFOTopik Amazon dan SQS antrian Amazon tahan lama. Kedua jenis sumber daya ini menyimpan pesan secara redundan di beberapa Availability Zone, dan menyediakan antrean surat mati untuk menangani kasus luar biasa.

Di AmazonSNS, pengiriman pesan gagal saat SNS topik Amazon tidak dapat mengakses SQS antrian Amazon berlangganan karena kesalahan sisi klien atau sisi server:

- Kesalahan sisi klien terjadi ketika SNS FIFO topik Amazon memiliki metadata langganan basi. Dua penyebab umum kesalahan sisi klien adalah ketika pemilik SQS antrian Amazon melakukan salah satu hal berikut:
  - Menghapus antrean.

  - Mengubah kebijakan antrian dengan cara yang mencegah prinsipal SNS layanan Amazon mengirimkan pesan ke sana.

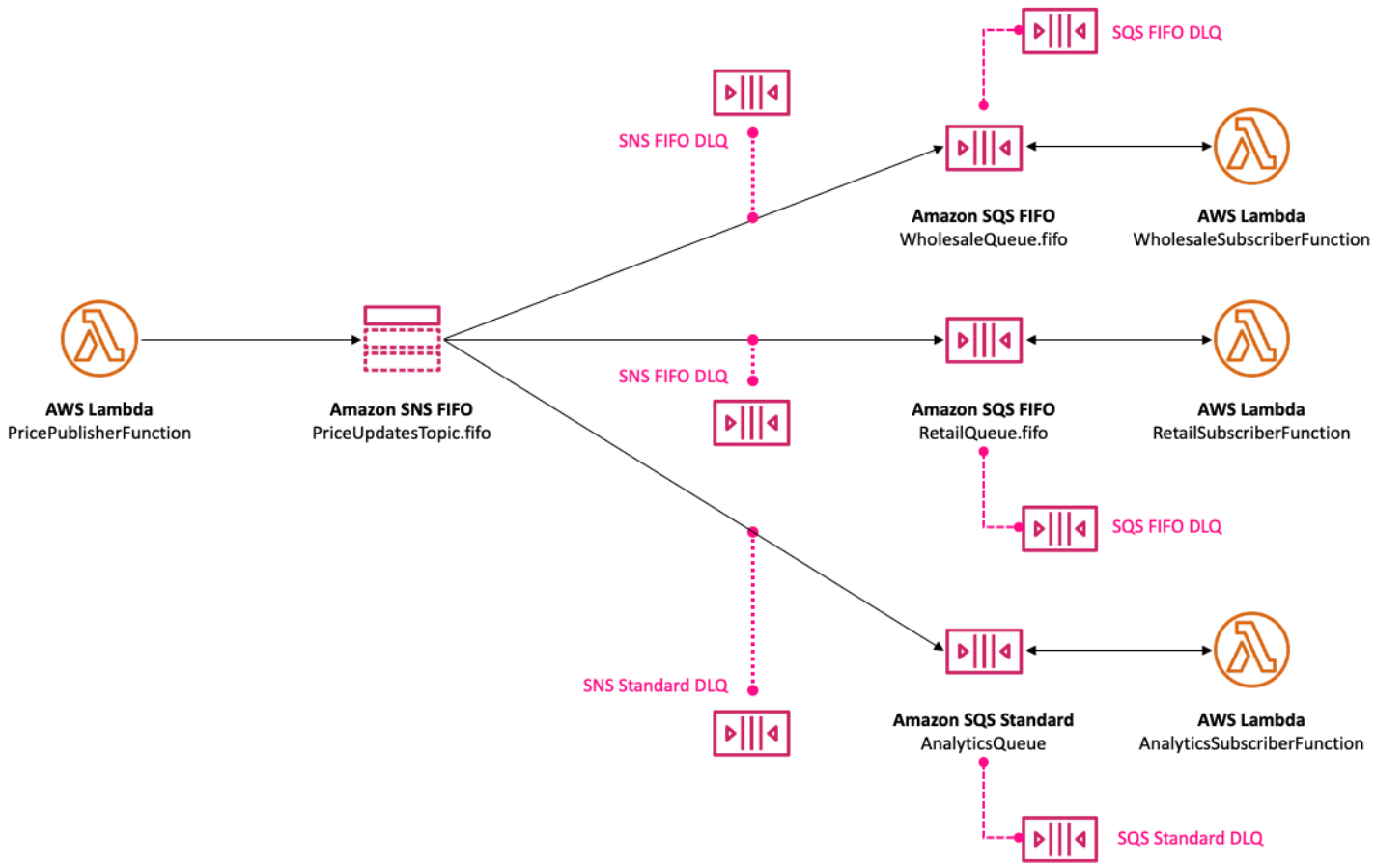
  Amazon SNS tidak mencoba lagi mengirimkan pesan yang gagal karena kesalahan sisi klien.
- Kesalahan sisi server dapat terjadi dalam situasi ini:
  - SQSLayanan Amazon tidak tersedia.
  - Amazon SQS gagal memproses permintaan yang valid dari SNS layanan Amazon.

  Ketika kesalahan sisi server terjadi, SNS FIFO topik Amazon mencoba kembali pengiriman yang gagal hingga 100.015 kali selama 23 hari. Untuk informasi selengkapnya, lihat Mencoba lagi pengiriman SNS pesan Amazon.

Untuk semua jenis kesalahan, Amazon SNS dapat mengesampingkan pesan ke antrian SQS surat mati Amazon sehingga data tidak hilang.

Di AmazonSQS, pemrosesan pesan gagal ketika aplikasi konsumen gagal menerima pesan, memprosesnya, dan menghapusnya dari antrian. Ketika jumlah maksimum permintaan terima gagal, Amazon SQS dapat mengesampingkan pesan ke antrian surat mati sehingga data tidak hilang.

Dalam kasus penggunaan contoh manajemen harga suku cadang mobil, perusahaan dapat menetapkan antrian SQS surat mati Amazon () DLQ untuk setiap langganan SNS FIFO topik Amazon, serta untuk setiap antrian Amazon yang berlangganan. SQS Ini melindungi perusahaan dari kerugian pembaruan harga.



Antrian surat mati yang terkait dengan SNS langganan Amazon harus berupa SQS antrian Amazon dengan jenis yang sama dengan antrian berlangganan. Misalnya, SNS FIFO langganan Amazon untuk SQS FIFO antrian Amazon harus memiliki antrian Amazon sebagai SQS FIFO antrian surat mati. Demikian pula, SNS FIFO langganan Amazon untuk antrian SQS standar Amazon harus memiliki antrian SQS standar Amazon sebagai antrian huruf mati. Untuk informasi selengkapnya, lihat Antrian SNS surat mati Amazon dan Mendesain aplikasi tanpa server yang tahan lama dengan DLQs AmazonSQS, SNS Amazon, AWS Lambda posting di Blog AWS Komputasi.

Untuk ketahanan yang lebih lama untuk membantu pemulihan dari kegagalan hilir, pemilik topik juga dapat menggunakan FIFO topik untuk mengarsipkan pesan hingga 365 hari. Pelanggan topik kemudian dapat memutar ulang pesan tersebut ke titik akhir berlangganan untuk memulihkan pesan yang hilang karena kegagalan dalam aplikasi hilir, atau untuk mereplikasi status aplikasi yang ada. Untuk lebih lanjut, lihatPengarsipan dan pemutaran ulang SNS pesan Amazon untuk topik FIFO.

Pengarsipan dan pemutaran ulang SNS pesan Amazon untuk topik FIFO

Apa itu pengarsipan dan pemutaran ulang pesan?

Amazon SNS menyediakan fitur pengarsipan dan pemutaran ulang pesan tanpa kode, yang dirancang khusus untuk topik FIFO (First-In-First-Out). Fitur ini memungkinkan pemilik topik untuk menyimpan pesan langsung dalam arsip topik hingga 365 hari dan memutarnya kembali ke pelanggan bila diperlukan. Pengarsipan dan pemutaran ulang pesan sangat penting untuk memulihkan pesan yang hilang dan menyinkronkan aplikasi di seluruh wilayah atau sistem dengan mereplikasi status.

Fungsi ini dapat diakses melalui AWS API,SDK, AWS CloudFormation, dan AWS Management Console.

Kasus penggunaan kunci

- Pemulihan pesan - Memulihkan pesan yang hilang karena kegagalan aplikasi hilir dengan memutar ulang ke titik akhir pelanggan.
- Replikasi status - Mereplikasi status sistem yang ada di lingkungan baru dengan memutar ulang pesan yang dimulai dari stempel waktu tertentu.
- Koreksi kesalahan - Kirim ulang pesan yang tidak terjawab selama pemadaman untuk memastikan semua peristiwa diproses dengan benar.

Komponen pengarsipan dan pemutaran ulang pesan

Kelola pengarsipan dan pemutaran ulang pesan untuk SNS FIFO topik Amazon, termasuk menyetel periode penyimpanan, memantau pesan yang diarsipkan menggunakan CloudWatch, memulai pemutaran ulang melalui atribut langganan, dan memahami izin yang diperlukan untuk memodifikasi dan memulai pemutaran ulang.

Pengarsipan pesan

- Pemilik topik mengaktifkan fitur pengarsipan dan menetapkan periode penyimpanan pesan, yang bisa sampai 365 hari. Untuk lebih lanjut, lihat Pengarsipan SNS pesan Amazon untuk pemilik FIFO topik
- CloudWatch metrik membantu memantau pesan yang diarsipkan.

Putar ulang pesan

- Pelanggan memulai pemutaran ulang, memilih jendela waktu untuk pesan yang akan diproses ulang ke titik akhir berlangganan. Untuk lebih lanjut lihat,Putar ulang SNS pesan Amazon untuk pelanggan FIFO topik.
- Anda mengelola pemutaran ulang melalui atribut berlangganan menggunakan `ReplayPolicy` fitur tersebut.

Izin yang relevan

- `SetSubscriptionAttributes`— Diperlukan untuk mengkonfigurasi atau memodifikasi pengaturan replay menggunakan `ReplayPolicy` atribut pada langganan.
- `Subscribe`— Diperlukan untuk melampirkan langganan baru dan memulai pemutaran ulang.
- `GetTopicAttributes`— Memungkinkan melihat properti topik, tetapi inisiasi replay terutama berkisar pada manajemen berlangganan.

Pengarsipan SNS pesan Amazon untuk pemilik FIFO topik

Pengarsipan pesan menyediakan kemampuan untuk mengarsipkan satu salinan dari semua pesan yang dipublikasikan ke topik Anda. Anda dapat menyimpan pesan yang dipublikasikan dalam topik Anda dengan mengaktifkan kebijakan arsip pesan pada topik, yang memungkinkan pengarsipan pesan untuk semua langganan yang ditautkan ke topik tersebut. Pesan dapat diarsipkan minimal satu hari hingga maksimal 365 hari.

Biaya tambahan berlaku saat menetapkan kebijakan arsip. Untuk informasi harga, lihat SNSharga Amazon.

Membuat kebijakan arsip pesan menggunakan AWS Management Console

Gunakan opsi ini untuk membuat kebijakan arsip pesan baru menggunakan AWS Management Console.

1. Masuk ke SNSkonsol Amazon.
2. Pilih topik atau buat yang baru. Untuk mempelajari lebih lanjut tentang membuat topik, lihat Membuat SNS topik Amazon.

> **Note**
>
> Pengarsipan dan pemutaran ulang SNS pesan Amazon hanya tersedia untuk topik application-to-application (A2A). FIFO

3. Pada halaman Edit topik, perluas bagian Kebijakan arsip.
4. Aktifkan fitur Kebijakan arsip, dan masukkan jumlah hari yang ingin Anda arsipkan pesan dalam topik.
5. Pilih Simpan perubahan.

Untuk melihat, mengedit, dan menonaktifkan kebijakan topik pengarsipan pesan

- Pada halaman Detail topik, kebijakan Retensi menampilkan status kebijakan arsip, termasuk jumlah hari yang disetel. Pilih tab Kebijakan arsip untuk melihat detail arsip pesan berikut:
  - Status — Status arsip dan pemutaran ulang muncul sebagai aktif saat kebijakan arsip diterapkan. Status arsip dan pemutaran ulang muncul sebagai tidak aktif saat kebijakan arsip disetel ke objek kosongJSON.
  - Periode penyimpanan pesan — Jumlah hari yang ditentukan untuk penyimpanan pesan.
  - Tanggal mulai arsip — Tanggal dari mana pelanggan dapat memutar ulang pesan.
  - JSONpratinjau — JSON Pratinjau kebijakan arsip.
- (Opsional) Untuk mengedit kebijakan arsip, buka halaman ringkasan topik dan pilih Edit.
- (Opsional) Untuk menonaktifkan kebijakan arsip, buka halaman ringkasan topik dan pilih Edit. Nonaktifkan Kebijakan Arsip dan pilih Simpan perubahan.
- (Opsional) Untuk menghapus topik dengan kebijakan arsip, Anda harus terlebih dahulu menonaktifkan kebijakan arsip seperti yang dijelaskan sebelumnya.

> **Important**
>
> Untuk menghindari penghapusan pesan yang tidak disengaja, Anda tidak dapat menghapus topik dengan kebijakan arsip pesan aktif. Kebijakan arsip pesan topik harus dinonaktifkan sebelum topik dapat dihapus. Saat Anda menonaktifkan kebijakan arsip pesan, Amazon SNS menghapus semua pesan yang diarsipkan. Saat menghapus topik, langganan dihapus, dan pesan apa pun yang sedang transit mungkin tidak terkirim.

Membuat kebijakan arsip pesan menggunakan API

Untuk membuat kebijakan arsip pesan menggunakanAPI, Anda perlu menambahkan atribut `ArchivePolicy` ke topik Anda. Anda dapat mengatur `ArchivePolicy` menggunakan API tindakan `CreateTopic` dan`SetTopicAttributes`. `ArchivePolicy`memiliki nilai tunggal,`MessageRetentionPeriod`, yang mewakili jumlah hari Amazon SNS menyimpan pesan. Untuk mengaktifkan pengarsipan pesan untuk topik Anda, setel `MessageRetentionPeriod` ke nilai integer lebih besar dari nol. Misalnya, untuk menyimpan pesan dalam arsip Anda selama 30 hari, setel `ArchivePolicy` ke:

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

Untuk menonaktifkan pengarsipan pesan untuk topik Anda, dan menghapus arsip, batalkan pengaturan`ArchivePolicy`, sebagai berikut:

```
{}
```

Membuat kebijakan arsip pesan menggunakan SDK

Untuk menggunakan AWS SDK, Anda harus mengkonfigurasinya dengan kredensil Anda. Untuk informasi selengkapnya, lihat Berbagi `config` dan `credentials` file di Panduan Referensi Alat AWS SDKs dan Alat.

Contoh kode berikut menunjukkan cara mengatur SNS topik Amazon `ArchivePolicy` untuk menyimpan semua pesan yang dipublikasikan ke topik selama 30 hari.

```
// 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);
```

Membuat kebijakan arsip pesan menggunakan AWS CloudFormation

Untuk membuat kebijakan arsip menggunakan AWS CloudFormation lihat `AWS::SNS::Topic`di Panduan AWS CloudFormation Pengguna.

Berikan akses ke arsip terenkripsi

Sebelum pelanggan dapat mulai memutar ulang pesan dari topik terenkripsi, Anda harus menyelesaikan langkah-langkah berikut. Karena pesan sebelumnya diputar ulang, Amazon SNS perlu menyediakan `Decrypt` akses ke KMS kunci yang digunakan untuk mengenkripsi pesan dalam arsip.

1. Saat Anda mengenkripsi pesan dengan KMS kunci dan menyimpannya dalam topik, Anda harus memberi Amazon SNS kemampuan untuk mendekripsi pesan ini melalui Kebijakan Utama. Untuk lebih lanjut, lihatBerikan izin dekripsi ke Amazon SNS.
2. Aktifkan AWS KMS untuk AmazonSNS. Untuk lebih lanjut, lihatMengkonfigurasi izin AWS KMS.

> **Important**
>
> Ketika Anda menambahkan bagian baru ke kebijakan KMS utama Anda, jangan mengubah bagian yang ada dalam kebijakan. Jika enkripsi diaktifkan pada suatu topik, dan KMS kuncinya dinonaktifkan atau dihapus, atau kebijakan KMS kunci tidak dikonfigurasi dengan benar untuk AmazonSNS, Amazon SNS tidak dapat memutar ulang pesan ke pelanggan Anda.

Berikan izin dekripsi ke Amazon SNS

SNSAgar Amazon dapat mengakses pesan terenkripsi dari dalam arsip topik Anda dan memutarnya kembali ke titik akhir berlangganan, Anda harus mengaktifkan prinsip SNS layanan Amazon untuk mendekripsi pesan ini.

Berikut ini adalah contoh kebijakan yang diperlukan untuk mengizinkan kepala SNS layanan Amazon mendekripsi pesan yang disimpan selama pemutaran ulang pesan historis dari dalam topik Anda.

```
{
    "Sid": "Allow SNS to decrypt archived messages",
    "Effect": "Allow",
    "Principal": {
        "Service": "sns.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey"
    ],
    "Resource": "*"
}
```

Pantau metrik arsip pesan menggunakan Amazon CloudWatch

Anda dapat memantau pesan yang diarsipkan menggunakan Amazon CloudWatch menggunakan metrik berikut. Agar diberi tahu tentang anomali pada beban kerja Anda dan membantu menghindari dampak, Anda dapat mengonfigurasi CloudWatch alarm Amazon pada metrik ini. Untuk detail selengkapnya, lihat Pencatatan dan pemantauan di Amazon SNS.

Metrik	Deskripsi
ApproximateNumberOfMessagesArchived	Menyediakan pemilik topik dengan jumlah agregat pesan yang diarsipkan dalam arsip topik, pada resolusi 60 menit.
ApproximateNumberOfBytesArchived	Menyediakan pemilik topik dengan jumlah agregat byte yang diarsipkan, di semua pesan dalam arsip topik, pada resolusi 60 menit.
NumberOfMessagesArchiveProcessing	Memberikan pemilik topik dengan jumlah pesan yang disimpan ke arsip topik selama interval dalam resolusi 1 menit.
NumberOfBytesArchiveProcessing	Menyediakan pemilik topik dengan jumlah agregat byte yang disimpan ke arsip topik selama interval dalam resolusi 1 menit.

Ini `GetTopicAttributes` API memiliki `BeginningArchiveTime` properti, yang mewakili stempel waktu tertua di mana pelanggan dapat memulai pemutaran ulang. Berikut ini merupakan contoh respons untuk API tindakan ini:

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

Putar ulang SNS pesan Amazon untuk pelanggan FIFO topik

Amazon SNS replay memungkinkan pelanggan topik mengambil pesan yang diarsipkan dari penyimpanan data topik dan mengirimkan ulang (atau memutar ulang) mereka ke titik akhir berlangganan. Pesan dapat diputar ulang segera setelah langganan dibuat. Pesan yang diputar ulang memiliki konten yang sama`MessageId`, dan `Timestamp` sebagai salinan asli, dan juga berisi atribut`Replayed`, untuk membantu Anda mengidentifikasi bahwa itu adalah pesan yang diputar ulang. Untuk hanya memutar ulang pesan tertentu, Anda dapat menambahkan kebijakan filter ke langganan Anda. Untuk informasi lebih lanjut tentang memfilter pesan, lihatFilter pesan yang diputar ulang.

Membuat kebijakan pemutaran ulang pesan menggunakan AWS Management Console

Gunakan opsi ini untuk membuat kebijakan replay baru menggunakan. AWS Management Console

1. Masuk ke SNSkonsol Amazon.
2. Pilih langganan topik atau buat yang baru. Untuk mempelajari lebih lanjut tentang membuat langganan, lihatMembuat langganan ke SNS topik Amazon.
3. Untuk memulai pemutaran ulang pesan, buka drop-down Putar ulang dan pilih Mulai replay.
4. Dari modal Replay timeframe, buat pilihan berikut:

    a. Pilih tanggal dan waktu mulai putar ulang - Pilih tanggal (YYYY/MM/DDformat) dan waktu (format 24 jam jam: mm: ss) dari mana Anda ingin mulai memutar ulang pesan yang diarsipkan. Waktu mulai harus lebih lambat dari awal perkiraan waktu arsip.

b. (Opsional) Pilih tanggal dan waktu akhir pemutaran ulang - Pilih tanggal (YYYY/MM/DDformat) dan waktu (format 24 jam jam: mm: ss) saat Anda ingin berhenti memutar ulang pesan yang diarsipkan.

c. Pilih Mulai putar ulang.

5. (Opsional) Untuk menghentikan pemutaran ulang pesan, buka halaman Detail langganan dan pilih Hentikan pemutaran ulang dari tarik-turun Putar Ulang.
6. (Opsional) Untuk memantau metrik pemutaran ulang pesan dari dalam alur kerja ini menggunakan CloudWatch, lihat. Pantau metrik pemutaran ulang pesan menggunakan Amazon CloudWatch

Untuk melihat dan mengedit kebijakan pemutaran ulang pesan

Anda dapat melakukan tindakan berikut dari halaman Detail langganan:

- Untuk melihat status pemutaran ulang pesan, bidang status Putar ulang menampilkan nilai berikut:
  - Selesai — Putar ulang telah berhasil mengirim ulang semua pesan, dan sekarang mengirimkan pesan yang baru diterbitkan.
  - Sedang berlangsung - Putar ulang saat ini memutar ulang pesan yang dipilih.
  - Gagal - Pemutaran ulang tidak dapat diselesaikan.
  - Tertunda - Status default saat pemutaran ulang dimulai.
- (Opsional) Untuk mengubah kebijakan pemutaran ulang pesan, buka halaman Detail langganan dan pilih Mulai memutar ulang dari tarik-turun Putar Ulang. Memulai replay akan menggantikan replay yang ada.

Tambahkan kebijakan pemutaran ulang ke langganan menggunakan API

Untuk memutar ulang pesan yang diarsipkan, gunakan atribut. `ReplayPolicy` `ReplayPolicy`dapat digunakan dengan `Subscribe` dan `SetSubscriptionAttributes` API tindakan. Kebijakan ini memiliki nilai-nilai berikut:

- `StartingPoint`(Wajib) — Sinyal dari mana harus mulai memutar ulang pesan.
- `EndingPoint`(Opsional) — Sinyal kapan harus berhenti memutar ulang pesan. Jika `EndingPoint` dihilangkan, maka pemutaran ulang akan berlanjut hingga terjebak hingga waktu saat ini.

- `PointType`(Wajib) - Mengatur jenis titik awal dan akhir. Saat ini, nilai yang didukung untuk `PointType` adalah`Timestamp`.

Misalnya, untuk memulihkan dari kegagalan hilir dan mengirim ulang semua pesan untuk jangka waktu dua jam pada 1 Oktober 2023, gunakan `SetSubscriptionAttributes` API tindakan untuk menetapkan sebagai `ReplayPolicy` berikut:

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

Untuk memutar ulang semua pesan yang dikirim ke topik per 1 Oktober 2023, dan terus menerima semua pesan yang baru diterbitkan ke topik Anda, gunakan `SetSubscriptionAttributes` API tindakan untuk menetapkan langganan Anda sebagai berikut: `ReplayPolicy`

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

Untuk memverifikasi bahwa pesan telah diputar ulang, atribut boolean ditambahkan ke setiap pesan yang `Replayed` diputar ulang.

Tambahkan kebijakan pemutaran ulang ke langganan menggunakan SDK

Untuk menggunakan AWS SDK, Anda harus mengkonfigurasinya dengan kredensil Anda. Untuk informasi selengkapnya, lihat Berbagi `config` dan `credentials` file di Panduan Referensi Alat AWS SDKs dan Alat.

Contoh kode berikut menunjukkan cara menyetel langganan untuk mengirim ulang pesan dari arsip SNS FIFO topik Amazon untuk jangka waktu 2 jam pada 1 Oktober 2023. `ReplayPolicy`

```
// 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);
```

Memahami EndingPoint

Saat Anda menerapkan SNS langganan Amazon `ReplayPolicy` ke, `EndingPoint` nilainya opsional. Jika tidak `EndingPoint` disediakan, pemutaran ulang akan dimulai dari yang ditentukan `StartingPoint` dan berlanjut hingga mengikuti waktu saat ini, termasuk memproses pesan yang baru diterbitkan. Setelah tertangkap, langganan akan berfungsi sebagai langganan reguler, menerima pesan baru saat dipublikasikan.

Jika `EndingPoint` ditentukan, layanan akan memutar ulang pesan dari `StartingPoint` atas ke `EndingPoint` dan kemudian berhenti. Tindakan ini secara efektif menghentikan langganan. Saat langganan dijeda, pesan yang baru diterbitkan tidak akan dikirimkan ke titik akhir berlangganan.

Untuk melanjutkan pengiriman pesan, terapkan yang baru `ReplayPolicy` tanpa memberikan`EndingPoint`, dan atur `StartingPoint` ke titik waktu yang diinginkan untuk terus menerima pesan. Misalnya, untuk melanjutkan langganan dari tempat pemutaran ulang sebelumnya selesai, atur yang baru `StartingPoint` ke yang disediakan `EndingPoint` sebelumnya.

Filter pesan yang diputar ulang

Pemfilteran SNS pesan Amazon memungkinkan Anda mengontrol pesan yang diputar ulang yang SNS diputar ulang Amazon ke titik akhir pelanggan Anda. Saat pemfilteran pesan dan pengarsipan pesan diaktifkan, Amazon SNS pertama-tama mengambil pesan dari penyimpanan data topik, lalu menerapkan pesan terhadap langganan. `FilterPolicy` Pesan dikirim ke titik akhir berlangganan saat ada kecocokan, jika tidak, pesan akan disaring. Untuk informasi selengkapnya, lihat Kebijakan filter SNS langganan Amazon.

Pantau metrik pemutaran ulang pesan menggunakan Amazon CloudWatch

Anda dapat memantau pesan replay menggunakan Amazon CloudWatch menggunakan metrik berikut. Agar diberi tahu tentang anomali pada beban kerja Anda dan membantu menghindari dampak, Anda dapat mengonfigurasi CloudWatch alarm Amazon pada metrik ini. Untuk detail selengkapnya, lihat Pencatatan dan pemantauan di Amazon SNS.

Metrik	Deskripsi
NumberOfReplayedNotificationsDelivered	Menyediakan pelanggan dengan jumlah agregat pesan yang diputar ulang dari arsip topik, pada resolusi 1 menit.
NumberOfReplayedNotificationsFailed	Menyediakan pelanggan dengan jumlah agregat pesan yang diputar ulang yang gagal disampaikan dari arsip topik, pada resolusi 1 menit.

Contoh SNS kode Amazon untuk FIFO topik

Anda dapat menggunakan contoh kode berikut untuk mengintegrasikan kasus penggunaan contoh manajemen harga suku cadang mobil menggunakan SNS FIFO topik Amazon dengan SQS FIFO antrian Amazon atau antrian standar.

Menggunakan sebuah AWS SDK

Dengan menggunakan AWS SDK, Anda membuat SNS FIFO topik Amazon dengan menyetel `FifoTopic` atributnya**`true`**. Anda membuat SQS FIFO antrian Amazon dengan menyetel `FifoQueue` atributnya. **`true`** Juga, Anda harus menambahkan **`.fifo`** akhiran ke nama setiap FIFO sumber daya. Setelah membuat FIFO topik atau antrian, Anda tidak dapat mengubahnya menjadi topik atau antrian standar.

Contoh kode berikut membuat sumber daya antrian ini FIFO dan standar:

- SNSFIFOTopik Amazon yang mendistribusikan pembaruan harga
- SQSFIFOAntrian Amazon yang menyediakan pembaruan ini untuk aplikasi grosir dan eceran
- Antrian SQS standar Amazon untuk aplikasi analitik yang menyimpan catatan, yang dapat ditanyakan untuk intelijen bisnis (BI)

- SNSFIFOLangganan Amazon yang menghubungkan tiga antrian ke topik

Contoh ini menetapkan kebijakan filter dalam langganan. Jika Anda menguji contoh dengan menerbitkan pesan ke topik, pastikan Anda mempublikasikan pesan dengan `business` atribut tersebut. Tentukan baik `retail` atau `wholesale` untuk nilai atribut. Jika tidak, pesan difilter dan tidak dikirim ke antrean berlangganan. Untuk informasi selengkapnya, lihat Pemfilteran SNS pesan Amazon untuk topik FIFO.

Java

SDKuntuk Java 2.x

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

Contoh ini

- membuat SNS FIFO topik Amazon, dua SQS FIFO antrian Amazon, dan satu antrian Standar.
- berlangganan antrian ke topik dan menerbitkan pesan ke topik tersebut.

Tes memverifikasi penerimaan pesan ke setiap antrian. Contoh lengkap juga menunjukkan penambahan kebijakan akses dan menghapus sumber daya di akhir.

```
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);
        }
    }
```

- Untuk API detailnya, lihat topik berikut di AWS SDK for Java 2.x APIReferensi.

- CreateTopic
- Publikasikan
- Berlangganan

Python

SDKuntuk Python (Boto3)

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

Buat SNS FIFO topik Amazon, berlangganan Amazon SQS FIFO dan antrian standar ke topik, dan publikasikan pesan ke topik tersebut.

```
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
```

- Untuk API detailnya, lihat topik berikut AWS SDKuntuk Referensi Python (Boto3). API
  - CreateTopic
  - Publikasikan
  - Berlangganan

SAP ABAP

SDKuntuk SAP ABAP

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

Buat FIFO topik, berlangganan SQS FIFO antrian Amazon ke topik, dan publikasikan pesan ke SNS topik Amazon.

```
    " 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.
```

- Untuk API detailnya, lihat topik berikut AWS SDKuntuk SAP ABAP API referensi.
  - CreateTopic
  - Publikasikan
  - Berlangganan

Menerima pesan dari FIFO langganan

Anda sekarang dapat menerima pembaruan harga di tiga aplikasi berlangganan. Seperti yang ditunjukkan padathe section called “FIFOkasus penggunaan topik”, titik masuk untuk setiap aplikasi konsumen adalah SQS antrian Amazon, yang AWS Lambda fungsinya yang sesuai dapat polling secara otomatis. Ketika SQS antrian Amazon adalah sumber peristiwa untuk fungsi Lambda, Lambda menskalakan armada poller sesuai kebutuhan untuk mengkonsumsi pesan secara efisien.

Untuk informasi selengkapnya, lihat Menggunakan AWS Lambda dengan Amazon SQS di Panduan AWS Lambda Pengembang. Untuk informasi tentang menulis poller antrian Anda sendiri, lihat Rekomendasi untuk SQS standar Amazon dan FIFO antrian di Panduan Pengembang Layanan Antrian Sederhana Amazon dan di ReceiveMessageReferensi Layanan Antrian Sederhana Amazon. API

Menggunakan AWS CloudFormation

AWS CloudFormation memungkinkan Anda untuk menggunakan file template untuk membuat dan mengkonfigurasi kumpulan AWS sumber daya bersama-sama sebagai satu unit. Bagian ini memiliki contoh templat yang menciptakan berikut ini:

- SNSFIFOTopik Amazon yang mendistribusikan pembaruan harga
- SQSFIFOAntrian Amazon yang menyediakan pembaruan ini untuk aplikasi grosir dan eceran
- Antrian SQS standar Amazon untuk aplikasi analitik yang menyimpan catatan, yang dapat ditanyakan untuk intelijen bisnis (BI)
- SNSFIFOLangganan Amazon yang menghubungkan tiga antrian ke topik
- Sebuah kebijakan filter yang menentukan bahwa aplikasi pelanggan hanya menerima pembaruan harga yang mereka butuhkan

> **Note**
>
> Jika Anda menguji contoh kode ini dengan menerbitkan pesan ke topik, pastikan Anda mempublikasikan pesan dengan `business` atribut. Tentukan baik `retail` atau `wholesale` untuk nilai atribut. Jika tidak, pesan difilter dan tidak dikirim ke antrean berlangganan.

```
{
  "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"
          }
        ]
      }
    }
  }
}
```

Untuk informasi selengkapnya tentang penerapan AWS sumber daya menggunakan AWS CloudFormation templat, lihat Memulai di Panduan AWS CloudFormation Pengguna.

SNSPemfilteran pesan Amazon

Secara default, pelanggan SNS topik Amazon menerima setiap pesan yang dipublikasikan ke topik tersebut. Untuk hanya menerima sebagian pesan, pelanggan harus menetapkan kebijakan filter ke langganan topik.

Kebijakan filter adalah JSON objek yang berisi properti yang menentukan pesan mana yang diterima pelanggan. Amazon SNS mendukung kebijakan yang bekerja pada atribut pesan atau pada badan pesan, sesuai dengan cakupan kebijakan filter yang Anda tetapkan untuk langganan. Kebijakan filter untuk badan pesan mengasumsikan bahwa payload pesan adalah objek yang terbentuk dengan baikJSON.

Jika langganan tidak memiliki kebijakan filter, pelanggan menerima setiap pesan yang dipublikasikan ke topiknya. Saat Anda memublikasikan pesan ke topik dengan kebijakan filter, Amazon akan SNS membandingkan atribut pesan atau badan pesan dengan properti dalam kebijakan filter untuk setiap langganan topik. Jika semua atribut pesan atau properti isi pesan memenuhi kondisi yang ditentukan dalam kebijakan filter, Amazon akan SNS mengirimkan pesan ke pelanggan. Jika tidak, Amazon SNS tidak mengirim pesan ke pelanggan itu.

Untuk informasi selengkapnya, lihat Memfilter Pesan yang Dipublikasikan ke Topik.

Lingkup kebijakan filter SNS langganan Amazon

Atribut `FilterPolicyScope` subscription memungkinkan Anda memilih lingkup pemfilteran dengan menetapkan salah satu nilai berikut:

- `MessageAttributes`— Kebijakan filter diterapkan pada atribut pesan. Ini adalah opsi default.
- `MessageBody`— Kebijakan filter diterapkan ke badan pesan.

> **Note**
>
> Jika tidak ada cakupan kebijakan filter yang ditentukan untuk kebijakan filter yang ada, cakupan default akan ditetapkan. `MessageAttributes`

Kebijakan filter SNS langganan Amazon

Kebijakan filter langganan memungkinkan Anda menentukan nama properti dan menetapkan daftar nilai untuk setiap nama properti. Untuk informasi selengkapnya, lihat SNSPemfilteran pesan Amazon.

Saat Amazon SNS mengevaluasi atribut pesan atau properti isi pesan terhadap kebijakan filter langganan, Amazon akan mengabaikan atribut pesan yang tidak ditentukan dalam kebijakan.

> **Important**
>
> AWS Layanan seperti IAM dan Amazon SNS menggunakan model komputasi terdistribusi yang disebut konsistensi akhirnya. Penambahan atau perubahan kebijakan filter langganan memerlukan hingga 15 menit untuk diterapkan sepenuhnya.

Langganan menerima pesan di bawah ketentuan berikut:

- Bila cakupan kebijakan filter disetel ke`MessageAttributes`, setiap nama properti dalam kebijakan filter cocok dengan nama atribut pesan. Untuk setiap nama properti yang cocok dalam kebijakan filter, setidaknya satu nilai properti cocok dengan nilai atribut pesan.
- Bila cakupan kebijakan filter disetel ke`MessageBody`, setiap nama properti dalam kebijakan filter cocok dengan nama properti isi pesan. Untuk setiap nama properti yang cocok dalam kebijakan filter, setidaknya satu nilai properti cocok dengan nilai properti isi pesan.

Amazon SNS saat ini mendukung operator filter berikut:

- ANDlogika
- Logika ATAU
- ATAU operator
- Pencocokan kunci
- Nilai numerik pencocokan tepat
- Nilai numerik apa pun-tapi cocok
- Pencocokan rentang nilai numerik
- Nilai string pencocokan tepat
- String menghargai apa pun-tapi cocok

- Pencocokan string menggunakan awalan dengan operator apa pun kecuali
- Nilai string sama dengan kasus abaikan
- Pencocokan alamat IP nilai string
- Pencocokan awalan nilai string
- Pencocokan akhiran nilai string

Kebijakan filter SNS contoh Amazon

Contoh berikut menunjukkan payload pesan yang dikirimkan oleh SNS topik Amazon yang memproses transaksi pelanggan.

Contoh pertama mencakup `MessageAttributes` bidang dengan atribut yang menggambarkan transaksi:

- Minat pelanggan
- Nama penyimpanan
- State kejadian
- Harga pembelian di USD

Karena pesan ini menyertakan `MessageAttributes` bidang, langganan topik apa pun yang menetapkan a `FilterPolicy` dapat menerima atau menolak pesan secara selektif, selama `FilterPolicyScope` disetel ke `MessageAttributes` dalam langganan. Untuk informasi tentang menerapkan atribut pada olahpesan, lihat Atribut SNS pesan Amazon.

```
{
   "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"
      }
   }
}
```

Contoh berikut menunjukkan atribut yang sama termasuk dalam `Message` bidang, juga disebut sebagai payload pesan atau isi pesan. Langganan topik apa pun yang menyertakan a `FilterPolicy` dapat menerima atau menolak pesan secara selektif, selama `FilterPolicyScope` diatur `MessageBody` dalam langganan.

```
{
"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"
}
```

Kebijakan filter berikut menerima atau menolak pesan berdasarkan nama dan nilai properti mereka.

Kebijakan yang menerima contoh olahpesan

Properti dalam kebijakan filter langganan berikut cocok dengan atribut yang ditetapkan ke pesan contoh. Perhatikan bahwa kebijakan filter yang sama berfungsi untuk `FilterPolicyScope` apakah

itu disetel ke `MessageAttributes` atau`MessageBody`. Setiap pelanggan memilih ruang lingkup penyaringan mereka sesuai dengan komposisi pesan yang mereka terima dari topik tersebut.

Jika satu properti dalam kebijakan ini tidak cocok dengan atribut yang ditetapkan ke pesan, kebijakan akan menolak pesan tersebut.

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

Kebijakan yang menolak contoh olahpesan

Kebijakan filter langganan berikut memiliki beberapa ketidakcocokan antara propertinya dan atribut yang ditetapkan ke pesan contoh. Misalnya, karena nama `encrypted` properti tidak ada dalam atribut pesan, properti kebijakan ini menyebabkan pesan ditolak terlepas dari nilai yang ditetapkan padanya.

Jika terjadi ketidakcocokan, kebijakan menolak pesan.

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

Memfilter kendala kebijakan di Amazon SNS

Saat Anda membuat kebijakan filter, ingatlah batasan berikut.

Topik

- Kendala kebijakan umum
- Kendala kebijakan untuk penyaringan berbasis atribut
- Kendala kebijakan untuk penyaringan berbasis muatan

Kendala kebijakan umum

- Pencocokan String - Untuk pencocokan string dalam kebijakan filter, perbandingannya peka huruf besar/kecil.
- Pencocokan Numerik — Untuk pencocokan numerik, nilainya dapat berkisar dari -10^9 hingga 10^9 (-1 miliar hingga 1 miliar), dengan lima digit akurasi setelah titik desimal.
- Kompleksitas Kebijakan Filter — Untuk kompleksitas kebijakan filter, total kombinasi nilai tidak boleh melebihi 150. Untuk menghitung kombinasi total, kalikan jumlah nilai di setiap array dalam kebijakan filter.

Perhatikan contoh kebijakan berikut:

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

Array pertama memiliki tiga nilai, yang kedua memiliki satu nilai, dan yang ketiga memiliki dua nilai. Kombinasi total dihitung sebagai berikut:

```
3 x 1 x 2 = 6
```

- Kebijakan filter dapat berisi yang berikut: JSON
  - String terlampir dalam tanda kutip
  - Nomor
  - Kata kunci`true`,`false`, dan`null`, tanpa tanda kutip

- Saat menggunakan Amazon SNSAPI, Anda harus meneruskan kebijakan filter sebagai string UTF -8 yang valid. JSON

- Ukuran maksimum kebijakan filter adalah 256 KB.
- Secara default, Anda dapat memiliki hingga 200 kebijakan filter per topik, dan 10.000 kebijakan filter per AWS akun.

  Batas kebijakan ini tidak akan menghentikan langganan SQS antrian Amazon dibuat dengan. `Subscribe` API Namun, itu akan gagal ketika Anda melampirkan kebijakan filter dalam `Subscribe` API panggilan (atau `SetSubscriptionAttributes` API panggilan).

  Untuk meningkatkan kuota ini, Anda dapat menggunakan AWS Service Quotas.

Kendala kebijakan untuk penyaringan berbasis atribut

- Pemfilteran berbasis atribut adalah opsi default. `FilterPolicyScope`diatur ke `MessageAttributes` dalam langganan.
- Amazon SNS tidak menerima kebijakan filter bersarang untuk pemfilteran berbasis atribut.
- Amazon SNS membandingkan properti kebijakan hanya dengan atribut pesan yang memiliki tipe data berikut:
  - `String`
  - `String.Array`

    > **Important**
    >
    > Melewati objek dalam array tidak disarankan karena dapat menghasilkan hasil yang tidak terduga karena bersarang, yang tidak didukung oleh pemfilteran berbasis atribut. Gunakan pemfilteran berbasis muatan untuk kebijakan bersarang.

  - `Number`
- Amazon SNS mengabaikan atribut pesan dengan tipe `Binary` data.
- Kebijakan filter dapat memiliki maksimal lima nama atribut.

Kendala kebijakan untuk penyaringan berbasis muatan

- Amazon SNS menerima kebijakan filter bersarang untuk pemfilteran berbasis muatan. Untuk menghitung total kombinasi nilai dalam kebijakan filter, kalikan jumlah nilai di setiap array bersarang.

Perhatikan contoh kebijakan berikut:

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

- Array pertama memiliki empat nilai dalam kunci bersarang tiga tingkat, dan yang kedua memiliki tiga nilai dalam kunci bersarang dua tingkat. Kombinasi total dihitung sebagai berikut:

```
4 x 3 x 3 x 2 = 72
```

- Kebijakan filter dapat memiliki maksimal lima nama atribut. Untuk kebijakan bersarang, hanya kunci induk yang dihitung.
- Untuk beralih dari pemfilteran berbasis atribut (default) ke pemfilteran berbasis muatan, Anda harus menyetel ke dalam `FilterPolicyScope` langganan. `MessageBody`

AND/ATAU logika

Anda dapat menggunakan operasi yang menyertakan logika AND /OR untuk mencocokkan atribut pesan atau properti isi pesan.

Topik

- ANDlogika
- Logika ATAU
- ATAU operator

ANDlogika

Anda dapat menerapkan AND logika menggunakan beberapa nama properti.

Pertimbangkan kebijakan berikut:

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

Ini cocok dengan atribut pesan atau properti badan pesan dengan nilai `customer_interests` set ke `rugby` dan nilai `price_usd` set ke angka yang lebih besar dari 100.

> **Note**
>
> Anda tidak dapat menerapkan AND logika pada nilai atribut yang sama.

Logika ATAU

Anda dapat menerapkan logika OR dengan menetapkan beberapa nilai ke nama properti.

Pertimbangkan kebijakan berikut:

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

Ini cocok dengan atribut pesan atau properti badan pesan dengan nilai `customer_interests` set ke`rugby`,`football`, atau`baseball`.

ATAU operator

Anda dapat menggunakan `"$or"` operator untuk secara eksplisit menentukan kebijakan filter untuk mengekspresikan hubungan OR antara beberapa atribut dalam kebijakan.

Amazon SNS hanya mengakui `"$or"` hubungan ketika kebijakan telah memenuhi semua persyaratan berikut. Ketika semua kondisi ini tidak terpenuhi, `"$or"` diperlakukan sebagai nama atribut reguler, sama seperti string lain dalam kebijakan.

- Ada atribut `"$or"` bidang dalam aturan diikuti dengan array, misalnya`“$or” : []`.
- Setidaknya ada 2 objek dalam `"$or"` array:`"$or": [{}, {}]`.
- Tak satu pun dari objek dalam `"$or"` array memiliki nama bidang yang merupakan kata kunci cadangan.

Jika `"$or"` tidak, diperlakukan sebagai nama atribut normal, sama seperti string lain dalam kebijakan.

Kebijakan berikut tidak diuraikan sebagai hubungan OR karena numerik dan awalan adalah kata kunci yang dicadangkan.

```
{
   "$or": [ {"numeric" : 123}, {"prefix": "abc"} ]
}
```

ORcontoh operator

Standar`OR`:

```
{
  "source": [ "aws.cloudwatch" ],
  "$or": [
    { "metricName": [ "CPUUtilization" ] },
    { "namespace": [ "AWS/EC2" ] }
  ]
}
```

Logika filter untuk kebijakan ini adalah:

```
"source" && ("metricName" || "namespace")
```

Ini cocok dengan salah satu dari set atribut pesan berikut:

```
"source": {"Type": "String", "Value": "aws.cloudwatch"},
"metricName": {"Type": "String", "Value": "CPUUtilization"}
```

atau

```
"source": {"Type": "String", "Value": "aws.cloudwatch"},
```

```
"namespace": {"Type": "String", "Value": "AWS/EC2"}
```

Ini juga cocok dengan salah satu badan pesan berikut:

```
{
    "source": "aws.cloudwatch",
    "metricName": "CPUUtilization"
}
```

atau

```
{
    "source": "aws.cloudwatch",
    "namespace": "AWS/EC2"
}
```

Kendala kebijakan yang mencakup hubungan **OR**

Pertimbangkan kebijakan berikut:

```
{
    "source": [ "aws.cloudwatch" ],
    "$or": [
      { "metricName": [ "CPUUtilization", "ReadLatency" ] },
      {
        "metricType": [ "MetricType" ] ,
        "$or" : [
          { "metricId": [ 1234, 4321 ] },
          { "spaceId": [ 1000, 2000, 3000 ] }
        ]
      }
    ]
  }
```

Logika untuk kebijakan ini juga dapat disederhanakan sebagai:

```
("source" AND "metricName")
OR
("source" AND "metricType" AND "metricId")
OR
("source" AND "metricType" AND "spaceId")
```

Perhitungan kompleksitas untuk kebijakan dengan hubungan OR dapat disederhanakan sebagai jumlah kompleksitas kombinasi untuk setiap pernyataan OR.

Kombinasi total dihitung sebagai berikut:

```
(source * metricName) + (source * metricType * metricId) + (source * metricType *
 spaceId)
= (1 * 2) + (1 * 1 * 2) + (1 * 1 * 3)
= 7
```

`source`memiliki satu nilai, `metricName` memiliki dua nilai, `metricType` memiliki satu nilai, `metricId` memiliki dua nilai dan `spaceId` memiliki tiga nilai.

Pertimbangkan kebijakan filter bersarang berikut:

```
{
    "$or": [
      { "metricName": [ "CPUUtilization", "ReadLatency" ] },
      { "namespace": [ "AWS/EC2", "AWS/ES" ] }
    ],
    "detail" : {
      "scope" : [ "Service" ],
      "$or": [
        { "source": [ "aws.cloudwatch" ] },
        { "type": [ "CloudWatch Alarm State Change"] }
      ]
    }
  }
```

Logika untuk kebijakan ini dapat disederhanakan sebagai:

```
("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")
```

Perhitungan untuk kombinasi total adalah sama untuk kebijakan non-bersarang kecuali kita perlu mempertimbangkan tingkat bersarang kunci.

Kombinasi total dihitung sebagai berikut:

```
(2 * 2 * 2) + (2 * 2 * 2) + (2 * 2 * 2) + (2 * 2 * 2) = 32
```

`metricName`memiliki dua nilai, `namespace` memiliki dua nilai, `scope` adalah kunci bersarang dua tingkat dengan satu nilai, `source` adalah kunci bersarang dua tingkat dengan satu nilai, dan `type` merupakan kunci bersarang dua tingkat dengan satu nilai.

Pencocokan kunci

Anda dapat menggunakan `exists` operator untuk mencocokkan pesan masuk dengan atau tanpa properti tertentu dalam kebijakan filter. `exists`pencocokan hanya berfungsi pada simpul daun. Pencocokan yang ada tidak berfungsi pada simpul intermediet.

- Gunakan `"exists": true` untuk mencocokkan pesan masuk yang menyertakan properti yang ditentukan. Kunci harus memiliki nilai non-null dan non-kosong.

  Misalnya, properti kebijakan berikut menggunakan `exists` operator dengan nilai`true`:

  ```
  "store": [{"exists": true}]
  ```

  Ini cocok dengan daftar atribut pesan yang berisi kunci `store` atribut, seperti berikut ini:

  ```
  "store": {"Type": "String", "Value": "fans"}
  "customer_interests": {"Type": "String.Array", "Value": "[\"baseball\", \"basketball
  \"]"}
  ```

  Ini juga cocok dengan salah satu dari badan pesan berikut:

  ```
  {
      "store": "fans"
      "customer_interests": ["baseball", "basketball"]
  }
  ```

  Namun, itu tidak cocok dengan daftar atribut pesan apa pun tanpa kunci `store` atribut, seperti berikut ini:

  ```
  "customer_interests": {"Type": "String.Array", "Value": "[\"baseball\", \"basketball
  \"]"}
  ```

Juga tidak cocok dengan badan pesan berikut:

```
{
    "customer_interests": ["baseball", "basketball"]
}
```

- Gunakan `"exists": false` untuk mencocokkan pesan masuk yang tidak menyertakan properti yang ditentukan.

> **Note**
>
> `"exists": false`hanya cocok jika setidaknya ada satu atribut. Kumpulan atribut kosong menghasilkan filter yang tidak cocok.

Misalnya, properti kebijakan berikut menggunakan `exists` operator dengan nilai`false`:

```
"store": [{"exists": false}]
```

Itu tidak cocok dengan daftar atribut pesan yang berisi kunci `store` atribut, seperti berikut:

```
"store": {"Type": "String", "Value": "fans"}
"customer_interests": {"Type": "String.Array", "Value": "[\"baseball\", \"basketball
\"]"}
```

Itu juga tidak cocok dengan badan pesan berikut:

```
{
    "store": "fans"
    "customer_interests": ["baseball", "basketball"]
}
```

Namun, ini cocok dengan daftar atribut pesan apa pun tanpa kunci `store` atribut, seperti berikut ini:

```
"customer_interests": {"Type": "String.Array", "Value": "[\"baseball\", \"basketball
\"]"}
```

Ini juga cocok dengan badan pesan berikut:

```
{
    "customer_interests": ["baseball", "basketball"]
}
```

Pencocokan nilai numerik

Anda dapat memfilter pesan dengan mencocokkan nilai numerik dengan nilai atribut pesan atau ke nilai properti isi pesan. Nilai numerik tidak terlampir dalam tanda kutip ganda dalam kebijakan. JSON Anda dapat menggunakan operasi numerik berikut untuk pemfilteran.

> **Note**
>
> Awalan didukung hanya untuk pencocokan string.

Topik

- Pencocokan tepat
- Apa saja tapi tidak cocok
- Pencocokan rentang nilai

Pencocokan tepat

Jika nilai properti kebijakan menyertakan kata kunci `numeric` dan operator=, nilai properti tersebut cocok dengan atribut pesan atau nilai properti isi pesan yang memiliki nama yang sama dan nilai numerik yang sama.

Pertimbangkan properti kebijakan berikut:

```
"price_usd": [{"numeric": ["=",301.5]}]
```

Cocok dengan salah satu dari atribut olahpesan berikut:

```
"price_usd": {"Type": "Number", "Value": 301.5}
```

```
"price_usd": {"Type": "Number", "Value": 3.015e2}
```

Ini juga cocok dengan salah satu badan pesan berikut:

```
{
   "price_usd": 301.5
}
```

```
{
   "price_usd": 3.015e2
}
```

Apa saja tapi tidak cocok

Jika nilai properti kebijakan menyertakan kata kunci`anything-but`, nilai properti tersebut cocok dengan atribut pesan atau nilai properti isi pesan apa pun yang tidak menyertakan nilai properti kebijakan apa pun.

Pertimbangkan properti kebijakan berikut:

```
"price": [{"anything-but": [100, 500]}]
```

Cocok dengan salah satu dari atribut olahpesan berikut:

```
"price": {"Type": "Number", "Value": 101}
```

```
"price": {"Type": "Number", "Value": 100.1}
```

Ini juga cocok dengan salah satu badan pesan berikut:

```
{
   "price": 101
}
```

```
{
   "price": 100.1
}
```

Selain itu, ini cocok dengan atribut pesan berikut (karena berisi nilai yang bukan `100` atau`500`):

```
"price": {"Type": "Number.Array", "Value": "[100, 50]"}
```

Dan itu juga cocok dengan badan pesan berikut (karena berisi nilai yang bukan `100` atau`500`):

```
{
   "price": [100, 50]
}
```

Namun, tidak cocok dengan atribut olahpesan berikut:

```
"price": {"Type": "Number", "Value": 100}
```

Juga tidak cocok dengan badan pesan berikut:

```
{
   "price": 100
}
```

Pencocokan rentang nilai

Selain operator=, properti kebijakan numerik dapat mencakup operator berikut:<,, <=>, dan>=.

Pertimbangkan properti kebijakan berikut:

```
"price_usd": [{"numeric": ["<", 0]}]
```

Ini cocok dengan atribut pesan atau properti badan pesan dengan nilai numerik negatif.

Pertimbangkan atribut olahpesan lain:

```
"price_usd": [{"numeric": [">", 0, "<=", 150]}]
```

Ini cocok dengan atribut pesan atau properti badan pesan dengan angka positif hingga dan termasuk 150.

Pencocokan nilai string

Anda dapat memfilter pesan dengan mencocokkan nilai string dengan nilai atribut pesan atau nilai properti isi pesan. Nilai string diapit tanda kutip ganda dalam kebijakan. JSON Anda dapat menggunakan operasi string berikut untuk mencocokkan atribut pesan atau isi pesan.

Topik

Pencocokan tepat

Pencocokan yang tepat terjadi ketika nilai properti kebijakan cocok dengan satu atau beberapa nilai atribut pesan.

Pertimbangkan properti kebijakan berikut:

```
"customer_interests": ["rugby", "tennis"]
```

Cocok dengan atribut olahpesan berikut:

```
"customer_interests": {"Type": "String", "Value": "rugby"}
```

```
"customer_interests": {"Type": "String", "Value": "tennis"}
```

Ini juga cocok dengan badan pesan berikut:

```
{
   "customer_interests": "rugby"
}
```

```
{
   "customer_interests": "tennis"
}
```

Namun, tidak cocok dengan atribut olahpesan berikut:

```
"customer_interests": {"Type": "String", "Value": "baseball"}
```

Juga tidak cocok dengan badan pesan berikut:

```
{
   "customer_interests": "baseball"
}
```

Apa saja tapi tidak cocok

Jika nilai properti kebijakan menyertakan kata kuncianything-but, nilai tersebut cocok dengan atribut pesan atau nilai isi pesan apa pun yang tidak menyertakan nilai properti kebijakan apa pun. anything-butdapat dikombinasikan dengan"exists": false.

Pertimbangkan properti kebijakan berikut:

```
"customer_interests": [{"anything-but": ["rugby", "tennis"]}]
```

Cocok dengan salah satu dari atribut olahpesan berikut:

```
"customer_interests": {"Type": "String", "Value": "baseball"}
```

```
"customer_interests": {"Type": "String", "Value": "football"}
```

Ini juga cocok dengan salah satu badan pesan berikut:

```
{
   "customer_interests": "baseball"
}
```

```
{
   "customer_interests": "football"
}
```

Selain itu, ini cocok dengan atribut pesan berikut (karena berisi nilai yang bukan rugby atautennis):

```
"customer_interests": {"Type": "String.Array", "Value": "[\"rugby\", \"baseball\"]"}
```

Dan itu juga cocok dengan badan pesan berikut (karena berisi nilai yang bukan `rugby` atau`tennis`):

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

Namun, tidak cocok dengan atribut olahpesan berikut:

```
"customer_interests": {"Type": "String", "Value": "rugby"}
```

Juga tidak cocok dengan badan pesan berikut:

```
{
   "customer_interests": ["rugby"]
}
```

Menggunakan prefiks dengan operator **anything-but**

Untuk pencocokan string, Anda juga dapat menggunakan awalan dengan `anything-but` operator. Misalnya, properti kebijakan berikut menyangkal `order-` awalan:

```
"event":[{"anything-but": {"prefix": "order-"}}]
```

Cocok salah satu dari atribut berikut:

```
"event": {"Type": "String", "Value": "data-entry"}
```

```
"event": {"Type": "String", "Value": "order_number"}
```

Ini juga cocok dengan salah satu badan pesan berikut:

```
{
   "event": "data-entry"
}
```

```
{
   "event": "order_number"
}
```

Namun, tidak cocok dengan atribut olahpesan berikut:

```
"event": {"Type": "String", "Value": "order-cancelled"}
```

Juga tidak cocok dengan badan pesan berikut:

```
{
   "event": "order-cancelled"
}
```

Equals-ignore-case pencocokan

Ketika properti kebijakan menyertakan kata kuncie`quals-ignore-case`, properti tersebut akan melakukan kecocokan case-insensitive dengan atribut pesan atau nilai properti body apa pun.

Pertimbangkan properti kebijakan berikut:

```
"customer_interests": [{"equals-ignore-case": "tennis"}]
```

Cocok dengan salah satu dari atribut olahpesan berikut:

```
"customer_interests": {"Type": "String", "Value": "TENNIS"}
```

```
"customer_interests": {"Type": "String", "Value": "Tennis"}
```

Ini juga cocok dengan salah satu badan pesan berikut:

```
{
    "customer_interests": "TENNIS"
}
```

```
{
    "customer_interests": "teNnis"
{
```

Pencocokan alamat IP

Anda dapat menggunakan operator `cidr` untuk memeriksa apakah olahpesan masuk berasal dari alamat IP tertentu atau subnet.

Pertimbangkan properti kebijakan berikut:

```
"source_ip":[{"cidr": "10.0.0.0/24"}]
```

Cocok dengan salah satu dari atribut olahpesan berikut:

```
"source_ip": {"Type": "String", "Value": "10.0.0.0"}
```

```
"source_ip": {"Type": "String", "Value": "10.0.0.255"}
```

Ini juga cocok dengan salah satu badan pesan berikut:

```
{
   "source_ip": "10.0.0.0"
}
```

```
{
   "source_ip": "10.0.0.255"
}
```

Namun, tidak cocok dengan atribut olahpesan berikut:

```
"source_ip": {"Type": "String", "Value": "10.1.1.0"}
```

Juga tidak cocok dengan badan pesan berikut:

```
{
   "source_ip": "10.1.1.0"
}
```

Pencocokan prefiks

Jika properti kebijakan menyertakan kata kunci`prefix`, properti tersebut cocok dengan atribut pesan atau nilai properti isi apa pun yang dimulai dengan karakter yang ditentukan.

Pertimbangkan properti kebijakan berikut:

```
"customer_interests": [{"prefix": "bas"}]
```

Cocok dengan salah satu dari atribut olahpesan berikut:

```
"customer_interests": {"Type": "String", "Value": "baseball"}
```

```
"customer_interests": {"Type": "String", "Value": "basketball"}
```

Ini juga cocok dengan salah satu badan pesan berikut:

```
{
   "customer_interests": "baseball"
}
```

```
{
   "customer_interests": "basketball"
}
```

Namun, tidak cocok dengan atribut olahpesan berikut:

```
"customer_interests": {"Type": "String", "Value": "rugby"}
```

Juga tidak cocok dengan badan pesan berikut:

```
{
   "customer_interests": "rugby"
}
```

Pencocokan akhiran

Jika properti kebijakan menyertakan kata kunci`suffix`, properti tersebut cocok dengan atribut pesan atau nilai properti isi yang diakhiri dengan karakter yang ditentukan.

Pertimbangkan properti kebijakan berikut:

```
"customer_interests": [{"suffix": "ball"}]
```

Cocok dengan salah satu dari atribut olahpesan berikut:

```
"customer_interests": {"Type": "String", "Value": "baseball"}
```

```
"customer_interests": {"Type": "String", "Value": "basketball"}
```

Ini juga cocok dengan salah satu badan pesan berikut:

```
{
    "customer_interests": "baseball"
}
```

```
{
    "customer_interests": "basketball"
}
```

Namun, tidak cocok dengan atribut olahpesan berikut:

```
"customer_interests": {"Type": "String", "Value": "rugby"}
```

Juga tidak cocok dengan badan pesan berikut:

```
{
    "customer_interests": "rugby"
}
```

Menerapkan kebijakan filter langganan di Amazon SNS

Pemfilteran pesan di Amazon SNS memungkinkan Anda mengirimkan pesan secara selektif ke pelanggan berdasarkan kebijakan filter. Kebijakan ini menentukan kondisi yang harus dipenuhi pesan agar dikirimkan ke langganan. Meskipun pengiriman pesan mentah adalah opsi yang dapat memengaruhi pemrosesan pesan, filter langganan tidak diperlukan untuk berfungsi.

Anda dapat menerapkan kebijakan filter ke SNS langganan Amazon menggunakan SNS konsol Amazon. Atau, untuk menerapkan kebijakan secara terprogram, Anda dapat menggunakan Amazon SNSAPI, AWS Command Line Interface (AWS CLI), atau apa pun AWS SDK yang mendukung Amazon. SNS Anda juga bisa menggunakan AWS CloudFormation.

Mengaktifkan Pengiriman Pesan Mentah

Pengiriman pesan mentah memastikan bahwa muatan pesan dikirimkan apa adanya kepada pelanggan tanpa pengkodean atau transformasi tambahan. Ini dapat berguna ketika pelanggan

memerlukan format pesan asli untuk diproses. Namun, pengiriman pesan mentah tidak terkait langsung dengan fungsionalitas filter berlangganan.

Menerapkan Filter Berlangganan

Untuk menerapkan filter pesan ke langganan, Anda menentukan kebijakan filter menggunakan JSON sintaks. Kebijakan ini menetapkan kondisi yang harus dipenuhi pesan untuk dikirimkan ke langganan. Filter dapat didasarkan pada atribut pesan, seperti atribut pesan, struktur pesan, atau bahkan konten pesan.

Hubungan antara Pengiriman Pesan Mentah dan Filter Berlangganan

Meskipun mengaktifkan pengiriman pesan mentah dapat memengaruhi cara pesan dikirim dan diproses oleh pelanggan, ini bukan prasyarat untuk menggunakan filter berlangganan. Namun, dalam skenario di mana pelanggan memerlukan format pesan asli tanpa modifikasi apa pun, mengaktifkan pengiriman pesan mentah mungkin bermanfaat bersama filter berlangganan.

Pertimbangan untuk Penyaringan Efektif

Saat menerapkan pemfilteran pesan, pertimbangkan persyaratan spesifik aplikasi dan pelanggan Anda. Tentukan kebijakan filter yang secara akurat sesuai dengan kriteria pengiriman pesan untuk memastikan distribusi pesan yang efisien dan ditargetkan.

> **Important**
>
> AWS Layanan seperti IAM dan Amazon SNS menggunakan model komputasi terdistribusi yang disebut konsistensi akhirnya. Penambahan atau perubahan kebijakan filter langganan memerlukan hingga 15 menit untuk diterapkan sepenuhnya.

AWS Management Console

1. Masuk ke SNSkonsol Amazon.
2. Di panel navigasi, pilih Berlangganan.
3. Pilih langganan dan kemudian pilih Edit.
4. Pada halaman Edit, perluas bagian Kebijakan filter Langganan.
5. Pilih antara pemfilteran berbasis atribut atau pemfilteran berbasis muatan.
6. Di bidang JSONeditor, berikan JSONisi kebijakan filter Anda.
7. Pilih Simpan perubahan.

Amazon SNS menerapkan kebijakan filter Anda ke langganan.

AWS CLI

Untuk menerapkan kebijakan filter dengan AWS Command Line Interface (AWS CLI), gunakan `set-subscription-attributes`perintah, seperti yang ditunjukkan pada contoh berikut. Untuk opsi `--attribute-name`, tentukan `FilterPolicy`. Untuk`--attribute-value`, tentukan JSONkebijakan Anda.

```
$ aws sns set-subscription-attributes --subscription-arn arn:aws:sns: ... --
attribute-name FilterPolicy --attribute-value '{"store":["example_corp"],"event":
["order_placed"]}'
```

Untuk memberikan valid JSON bagi kebijakan Anda, lampirkan nama dan nilai atribut dalam tanda kutip ganda. Anda juga harus menyertakan seluruh argumen kebijakan dalam tanda kutip. Untuk menghindari tanda kutip lolos, Anda dapat menggunakan tanda kutip tunggal untuk melampirkan kebijakan dan tanda kutip ganda untuk melampirkan JSON nama dan nilai, seperti yang ditunjukkan pada contoh di atas.

Jika Anda ingin beralih dari pemfilteran pesan berbasis atribut (default) ke pemfilteran pesan berbasis muatan, Anda juga dapat menggunakan perintah tersebut. set-subscription-attributes Untuk opsi `--attribute-name`, tentukan `FilterPolicyScope`. Untuk `--attribute-value`, tentukan `MessageBody`.

```
$ aws sns set-subscription-attributes --subscription-arn arn:aws:sns: ... --attribute-
name FilterPolicyScope --attribute-value MessageBody
```

Untuk memverifikasi bahwa kebijakan filter diterapkan, gunakan perintah `get-subscription-attributes`. Atribut dalam output terminal harus menunjukkan kebijakan filter Anda untuk kunci `FilterPolicy`, seperti yang ditunjukkan dalam contoh berikut:

```
$ 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 SDKs

Contoh kode berikut menunjukkan cara menggunakan`SetSubscriptionAttributes`.

> **Important**
>
> Jika Anda menggunakan contoh SDK untuk Java 2.x, kelas tidak `SNSMessageFilterPolicy` tersedia di luar kotak. Untuk petunjuk tentang cara menginstal kelas ini, lihat contoh dari situs GitHub web.

CLI

AWS CLI

Untuk mengatur atribut langganan

`set-subscription-attributes`Contoh berikut menetapkan `RawMessageDelivery` atribut ke SQS langganan.

```
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
```

Perintah ini tidak menghasilkan output.

`set-subscription-attributes`Contoh berikut menetapkan `FilterPolicy` atribut ke SQS langganan.

```
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\"] }"
```

Perintah ini tidak menghasilkan output.

`set-subscription-attributes`Contoh berikut menghapus `FilterPolicy` atribut dari SQS langganan.

```
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 "{}"
```

Perintah ini tidak menghasilkan output.

- Untuk API detailnya, lihat SetSubscriptionAttributesdi Referensi AWS CLI Perintah.

Java

SDKuntuk Java 2.x

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

```
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);
        }
    }
}
```

- Untuk API detailnya, lihat SetSubscriptionAttributesdi AWS SDK for Java 2.x APIReferensi.

Python

SDKuntuk Python (Boto3)

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

```
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
```

- Untuk API detailnya, lihat SetSubscriptionAttributes AWSSDKReferensi Python (Boto3). API

Amazon SNS API

Untuk menerapkan kebijakan filter dengan Amazon SNSAPI, buat permintaan untuk `SetSubscriptionAttributes`tindakan tersebut. Setel `AttributeName` parameter ke`FilterPolicy`, dan setel `AttributeValue` parameter ke kebijakan filter AndaJSON.

Jika Anda ingin beralih dari pemfilteran pesan berbasis atribut (default) ke pemfilteran pesan berbasis muatan, Anda juga dapat menggunakan tindakan tersebut. `SetSubscriptionAttributes` Atur `AttributeName` parameter ke`FilterPolicyScope`, dan atur `AttributeValue` parameternya ke`MessageBody`.

AWS CloudFormation

Untuk menerapkan kebijakan filter menggunakan AWS CloudFormation, gunakan YAML templat JSON atau untuk membuat AWS CloudFormation tumpukan. Untuk informasi selengkapnya, lihat `FilterPolicy`properti `AWS::SNS::Subscription` sumber daya di Panduan AWS CloudFormation Pengguna dan AWS CloudFormation templat contoh.

1. Masuk ke konsol AWS CloudFormation tersebut.
2. Pilih Buat Tumpukan.
3. Pada halaman Pilihan Templat, pilih Unggah templat ke Amazon S3, pilih templat file, dan pilih Selanjutnya.
4. Di halaman Tentukan Detail, lakukan hal berikut:
   a. Untuk Nama Tumpukan, ketik `MyFilterPolicyStack`.
   b. Untuk myHttpEndpoint, ketik HTTP titik akhir untuk berlangganan topik Anda.

   > **Tip**
   >
   > Jika Anda tidak memiliki HTTP titik akhir, buat satu.

5. Pada halaman Opsi, pilih Selanjutnya.
6. Di halaman Tinjau, pilih Buat.

Menghapus kebijakan filter langganan di Amazon SNS

Untuk berhenti memfilter pesan yang dikirim ke langganan, hapus kebijakan filter langganan dengan menimpa dengan badan kosongJSON. Setelah Anda menghapus kebijakan, langganan akan menerima setiap pesan yang dipublikasikan.

Menggunakan AWS Management Console

1. Masuk ke SNSkonsol Amazon.
2. Di panel navigasi, pilih Berlangganan.
3. Pilih langganan dan kemudian pilih Edit.
4. Pada Edit ***EXAMPLE1-23bc-4567-d890-ef12g3hij456***halaman, perluas bagian Kebijakan filter Langganan.

5. Di bidang JSONeditor, berikan JSON isi kosong untuk kebijakan filter Anda:`{}`.
6. Pilih Simpan perubahan.

   Amazon SNS menerapkan kebijakan filter Anda ke langganan.

Menggunakan AWS CLI

Untuk menghapus kebijakan filter dengan AWS CLI, gunakan `set-subscription-attributes`perintah dan berikan JSON isi kosong untuk `--attribute-value` argumen:

```
$ aws sns set-subscription-attributes --subscription-arn arn:aws:sns: ... --attribute-name FilterPolicy --attribute-value "{}"
```

Menggunakan Amazon SNS API

Untuk menghapus kebijakan filter dengan Amazon SNSAPI, buat permintaan untuk `SetSubscriptionAttributes`tindakan tersebut. Atur `AttributeName` parameter ke`FilterPolicy`, dan berikan JSON badan kosong untuk `AttributeValue` parameter.

Perlindungan data pesan di Amazon SNS

Topik

Apa itu perlindungan data pesan?

Perlindungan data pesan melindungi data yang dipublikasikan ke SNS topik Amazon Anda dengan menggunakan kebijakan perlindungan data untuk mengaudit, menutupi, menyunting, atau memblokir informasi sensitif yang berpindah antar aplikasi atau layanan. AWS

Perlindungan data pesan memindai data yang bergerak untuk informasi yang dapat diidentifikasi secara pribadi (PII) dan informasi kesehatan yang dilindungi (PHI) menggunakan pengidentifikasi data. Anda dapat memilih untuk menggunakan pengidentifikasi data yang telah ditentukan (atau SNS dikelola Amazon) (misalnya, nama, alamat, nomor kartu kredit, dan kode obat resep), atau Anda dapat membuat pengidentifikasi data kustom Anda sendiri, khusus untuk kasus penggunaan bisnis Anda. Menggunakan informasi yang dipindai, perlindungan data pesan menyediakan log audit terperinci, dan memungkinkan Anda mengambil tindakan untuk melindungi data tersebut.

Perlindungan data pesan mendukung tindakan berikut untuk membantu melindungi informasi sensitif pelanggan:

- Audit — Audit hingga 99% data yang dipublikasikan ke SNS topik Amazon. Anda kemudian dapat memilih untuk mengirim temuan ke Amazon CloudWatch, Amazon S3, atau Amazon Data Firehose.
- De-identifikasi - Menyembunyikan atau menyunting data sensitif tanpa mengganggu penerbitan atau pengiriman pesan.
- Tolak — Blokir transmisi data antara aplikasi dan AWS sumber daya jika data sensitif ada dalam muatan.

> **Note**
>
> Amazon SNS mendukung perlindungan data pesan untuk topik SNS standar Amazon saja.

Mengapa saya harus menggunakan perlindungan data pesan?

Dengan memperkenalkan perlindungan data pesan ke dalam program tata kelola, manajemen risiko, dan kepatuhan Anda, Anda dapat menerapkan kebijakan perlindungan data yang membantu Anda mengidentifikasi dan mencegah kebocoran data. Ini memberi tim Anda alat yang dapat membantu mengurangi risiko keuangan, hukum, dan peraturan dengan mematuhi peraturan privasi sepertiHIPAA,, GDPRPCI, dan FedRAMP. Ini juga membebaskan pengembang Anda dari overhead operasional yang terkait dengan membangun dan mengelola alat Anda sendiri untuk melindungi data sensitif.

Misalnya, Anda dapat menggunakan perlindungan data pesan untuk membuat kebijakan audit guna menentukan apakah ada sistem yang secara tidak sengaja mengirim atau menerima data sensitif. Jika hasil audit Anda menunjukkan bahwa sistem mengirimkan informasi kartu kredit ke sistem yang tidak memerlukannya, Anda dapat menggunakan kebijakan pemblokiran untuk mencegah pengiriman data.

> **Note**
>
> Amazon SNS mendukung perlindungan data pesan untuk topik SNS standar Amazon saja.

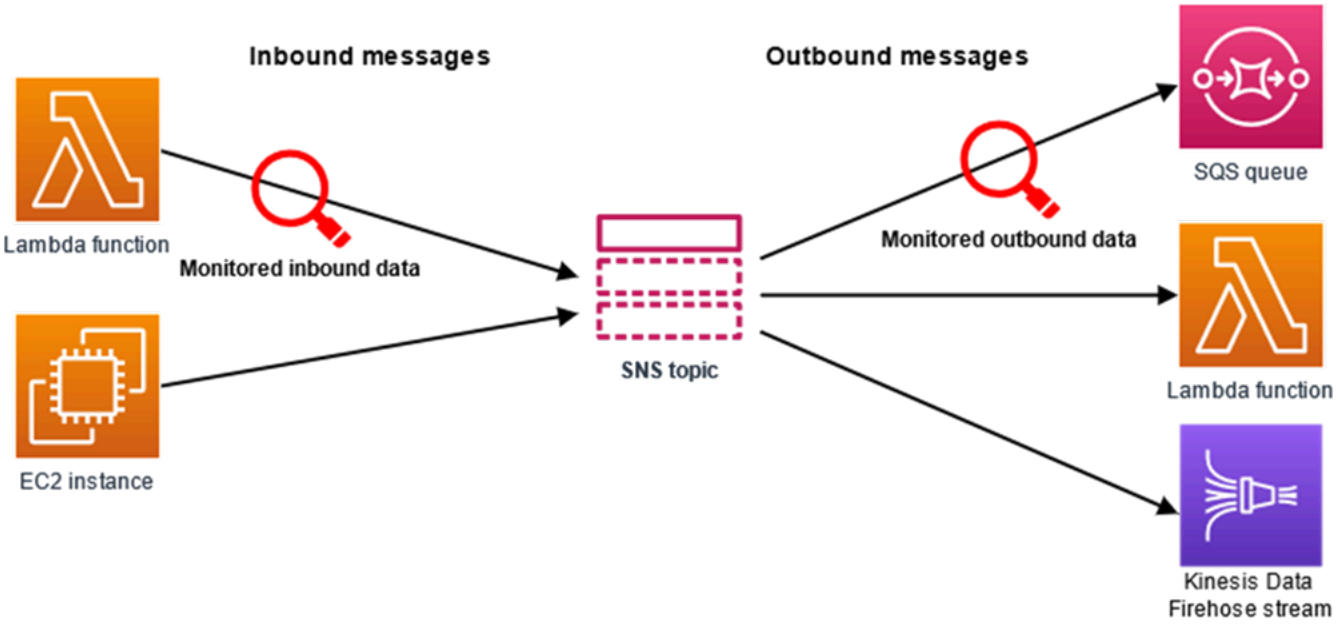
Memahami kebijakan perlindungan SNS data Amazon

Topik

- Menghapus kebijakan perlindungan data di Amazon SNS

Apa itu kebijakan perlindungan data?

Amazon SNS menggunakan kebijakan perlindungan data untuk memilih data sensitif yang ingin Anda pindai, dan tindakan yang ingin Anda ambil untuk melindungi data tersebut agar tidak dipertukarkan oleh SNS topik Amazon Anda. Untuk memilih data sensitif yang menarik, Anda menggunakan pengidentifikasi data. Perlindungan data SNS pesan Amazon kemudian mendeteksi data sensitif dengan menggunakan pembelajaran mesin dan pencocokan pola. Untuk bertindak atas pengidentifikasi data yang ditemukan, Anda dapat menentukan audit, de-identifikasi, atau menolak operasi. Operasi ini memungkinkan Anda mencatat data sensitif yang ditemukan (atau tidak ditemukan), menutupi atau menyunting data sensitif, atau menolak pengiriman pesan.

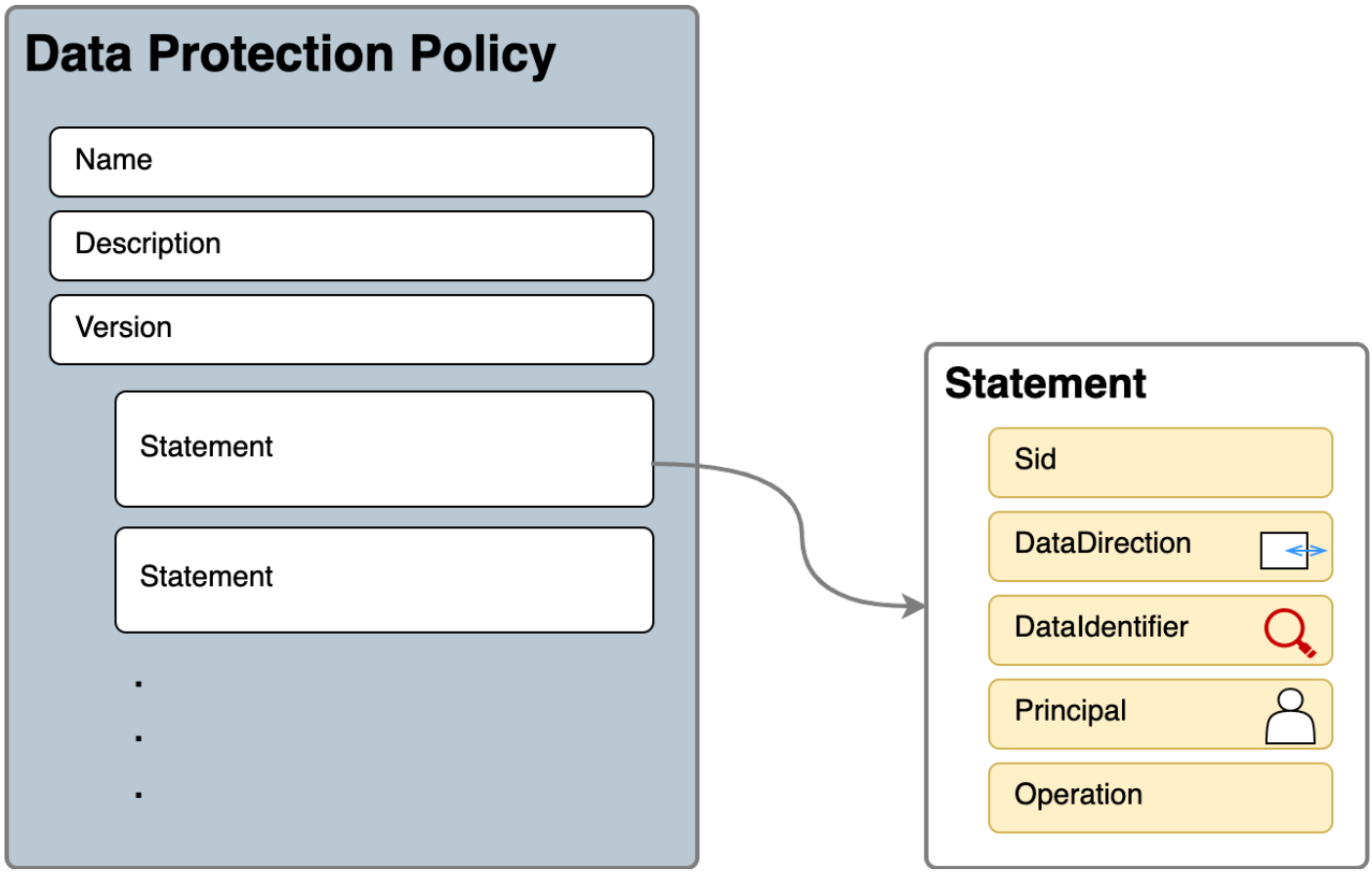


Bagaimana kebijakan perlindungan data terstruktur?

Seperti yang diilustrasikan pada gambar berikut, dokumen kebijakan perlindungan data mencakup elemen-elemen berikut:

- Informasi opsional untuk seluruh kebijakan di bagian atas dokumen
- Satu atau lebih pernyataan individu

Setiap pernyataan mencakup informasi tentang satu izin.



Hanya satu kebijakan perlindungan data yang dapat ditentukan per SNS topik Amazon. Kebijakan perlindungan data dapat memiliki satu atau lebih pernyataan penolakan atau de-identifikasi, tetapi hanya satu pernyataan audit.

JSONproperti untuk kebijakan perlindungan data

Kebijakan perlindungan data memerlukan informasi kebijakan dasar berikut untuk identifikasi:

- Nama — Nama kebijakan.
- Deskripsi (Opsional) — Deskripsi kebijakan.
- Versi - Versi bahasa kebijakan. Versi saat ini adalah 2021-06-01.
- Pernyataan — Daftar pernyataan yang menentukan tindakan kebijakan perlindungan data.

```
{
  "Name": "basicPII-protection",
  "Description": "Protect basic types of sensitive data",
  "Version": "2021-06-01",
```

```
  "Statement": [
        ...
  ]
}
```

JSONproperti untuk pernyataan kebijakan

Pernyataan kebijakan menetapkan konteks deteksi untuk operasi perlindungan data.

- Sid (Opsional) - Pengidentifikasi pernyataan.
- DataDirection— Masuk (untuk API permintaan Publikasikan) atau Keluar (untuk pengiriman pemberitahuan) sehubungan dengan topik Amazon. SNS
- DataIdentifier— Data sensitif yang harus dipindai oleh SNS topik Amazon. Misalnya, nama, alamat, atau nomor telepon.
- Principal — IAM Kepala sekolah yang diterbitkan untuk topik, atau IAM kepala sekolah yang berlangganan topik.
- Operasi — Tindakan tindak lanjut, baik Audit, De-identify (mask atau redact), atau Deny (block), yang dijalankan oleh SNS topik Amazon setelah menemukan data sensitif.

```
{
    "Sid": "basicPII-inbound-protection",
    "DataDirection": "Inbound",
    "Principal": ["*"],
    "DataIdentifier": [
        "arn:aws:dataprotection::aws:data-identifier/Name",
        "arn:aws:dataprotection::aws:data-identifier/PhoneNumber-US"
    ],
    "Operation": {
        ...
    }
}
```

JSONproperti untuk operasi pernyataan kebijakan

Pernyataan kebijakan menetapkan salah satu operasi perlindungan data berikut.

- Audit — Memancarkan metrik dan menemukan log tanpa mengganggu penerbitan atau pengiriman pesan.

- De-identifikasi - Menyembunyikan atau menyunting data sensitif tanpa mengganggu penerbitan pesan.
- Tolak - Memblokir permintaan SNS publikasi Amazon atau gagal pengiriman pesan.

Bagaimana cara menentukan IAM prinsip kebijakan perlindungan data saya?

Perlindungan data pesan menggunakan dua IAM prinsip yang berinteraksi dengan Amazon. SNS

1. Publikasikan API Principal (Inbound) — IAM Prinsipal yang diautentikasi yang memanggil Amazon. SNS `Publish` API
2. Subscription Principal (Outbound) — IAM Prinsipal yang diautentikasi yang disebut `Subscribe` API selama pembuatan langganan.

`SubscriptionPrincipal`Ini adalah properti SNS berlangganan Amazon yang tersedia untuk umum yang dapat diambil dari. `GetSubscriptionAttributes` API

```
{
  "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"
  }
}
```

Operasi kebijakan perlindungan data di Amazon SNS

Berikut ini adalah contoh kebijakan perlindungan data yang dapat Anda gunakan untuk mengaudit dan menolak data sensitif. Untuk tutorial lengkap yang menyertakan aplikasi contoh, lihat Memperkenalkan perlindungan data pesan untuk posting SNS blog Amazon.

Topik

Operasi audit

Operasi Audit mengambil sampel topik pesan masuk, dan mencatat temuan data sensitif di suatu AWS tujuan. Laju sampel dapat berupa bilangan bulat antara 0-99. Operasi ini membutuhkan salah satu dari jenis tujuan logging berikut:

1. FindingsDestination— Tujuan pencatatan saat SNS topik Amazon menemukan data sensitif di payload.
2. NoFindingsDestination— Tujuan pencatatan saat SNS topik Amazon tidak menemukan data sensitif di payload.

Anda dapat menggunakan yang berikut ini Layanan AWS di setiap jenis tujuan log berikut:

- Amazon CloudWatch Logs (Opsional) - `LogGroup` Harus ada di wilayah topik dan nama harus dimulai dengan /aws/vendedlogs/.
- Amazon Data Firehose (Opsional) — `DeliveryStream` Harus ada di wilayah topik dan memiliki Direct PUT sebagai sumber aliran pengiriman. Untuk detail tambahan, lihat Sumber, Tujuan, dan Nama di Panduan Pengembang Firehose Data Amazon.
- Amazon S3 (Opsional) - Nama bucket Amazon S3. Tindakan tambahan diperlukan untuk menggunakan bucket Amazon S3 dengan KMS enkripsi SSE - diaktifkan.

```
{
  "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"
            }
        }
    }
  }
}
```

Izin yang diperlukan saat menentukan tujuan log

Saat menentukan tujuan pencatatan dalam kebijakan perlindungan data, Anda harus menambahkan izin berikut ke kebijakan IAM identitas IAM prinsipal yang memanggil Amazon SNS `PutDataProtectionPolicy`API, atau `--data-protection-policy` parameter `CreateTopic` API dengan.

Tujuan audit	IAMizin
Default	logs:CreateLogDelivery logs:GetLogDelivery logs:UpdateLogDelivery logs:DeleteLogDelivery logs:ListLogDeliveries
CloudWatchLogs	logs:PutResourcePolicy logs:DescribeResourcePolicies

Tujuan audit	IAMizin
	logs:DescribeLogGroups
Firehose	iam:CreateServiceLinkedRole firehose:TagDeliveryStream
S3	s3:PutBucketPolicy s3:GetBucketPolicy Tindakan tambahan diperlukan untuk menggunakan bucket Amazon S3 dengan KMS enkripsi SSE - diaktifkan.

```
{
  "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"
      ]
    }
  ]
}
```

Kebijakan kunci yang diperlukan untuk digunakan dengan SSE - KMS

Jika Anda menggunakan bucket Amazon S3 sebagai tujuan log, Anda dapat melindungi data di bucket dengan mengaktifkan Enkripsi Sisi Server dengan Amazon S3-Managed Keys (SSE-S3), atau Enkripsi Sisi Server dengan (-). AWS KMS keys SSE KMS Untuk informasi selengkapnya, lihat Melindungi data menggunakan enkripsi sisi server di Panduan Pengguna Amazon S3.

Jika Anda memilih SSE -S3, tidak diperlukan konfigurasi tambahan. Amazon S3 menangani kunci enkripsi.

Jika Anda memilih SSE -KMS, Anda harus menggunakan kunci yang dikelola pelanggan. Anda harus memperbarui kebijakan kunci untuk kunci terkelola pelanggan Anda sehingga akun pengiriman log dapat menulis ke bucket S3 Anda. Untuk informasi selengkapnya tentang kebijakan kunci yang diperlukan untuk digunakan dengan SSE -KMS, lihat enkripsi sisi server bucket Amazon S3 di Panduan Pengguna Log Amazon. CloudWatch

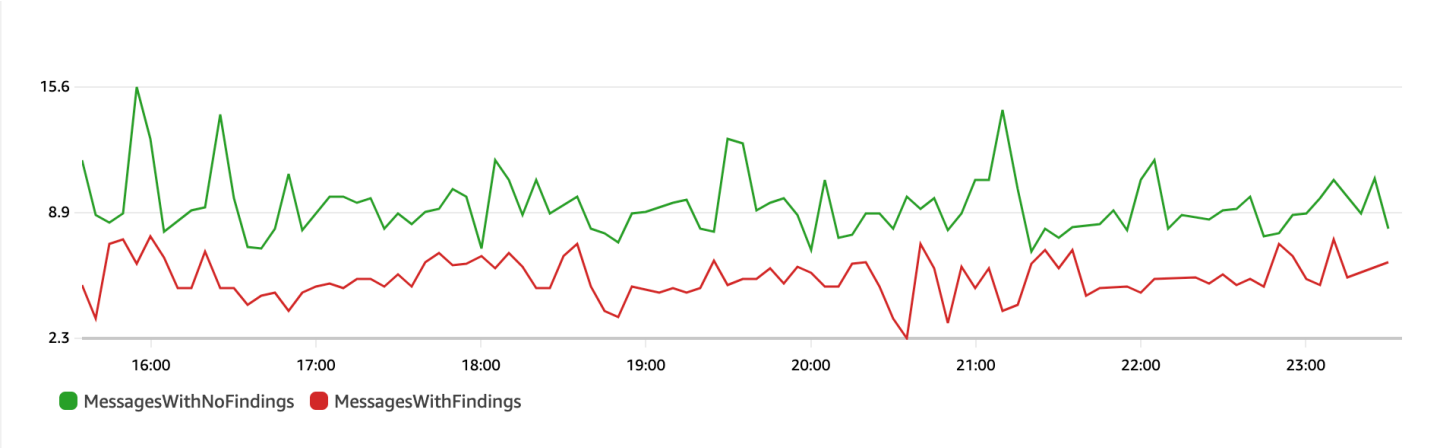
Contoh log tujuan audit

Dalam contoh berikut, `callerPrincipal` digunakan untuk mengidentifikasi sumber konten sensitif, dan `messageID` digunakan sebagai referensi untuk memeriksa terhadap `Publish` API respon.

```
{
  "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
        }
      ]
    }
  ]
}
```

Metrik operasi audit

Ketika operasi audit telah menentukan `FindingsDestination` atau `NoFindingsDestination` properti, pemilik topik juga menerima CloudWatch `MessagesWithFindings` dan `MessagesWithNoFindings` metrik.



De-identifikasi operasi

Operasi De-identifikasi menutupi atau menyunting data sensitif dari pesan yang dipublikasikan atau dikirim. Operasi ini tersedia untuk pesan masuk dan keluar, dan memerlukan salah satu jenis konfigurasi berikut:

- MaskConfig— Topeng menggunakan karakter yang didukung dari tabel berikut. Misalnya, ssn: `123-45-6789` menjadi ssn:. `##########`

```
{
"Operation": {
    "Deidentify": {
        "MaskConfig": {
            "MaskWithCharacter": "#"
          }
    }
}
```

Karakter topeng yang didukung	Nama
*	Tanda bintang
A-Z, a-z, dan 0-9	Alfanumerik
	Spasi
!	Tanda seru

Karakter topeng yang didukung	Nama
$	Tanda dolar
%	Tanda persen
&	Ampersand
()	Tanda kurung
+	Tanda plus
,	Koma
-	Tanda hubung
.	Periode
/\	Slash, tebasan belakang
#	Tanda angka
:	Usus besar
;	Titik koma
=, <>	Sama dengan kurang atau lebih besar dari
@	Pada tanda
[]	Kurung
^	Simbol tanda sisipan
_	menggarisbawahi
`	Backtick
|	Bilah vertikal
~	Simbol Tilde

- RedactConfig— Menyunting dengan menghapus data seluruhnya. Misalnya, ssn: `123-45-6789` menjadi ssn:.

```
{
"Operation": {
    "Deidentify": {
        "RedactConfig": {}
    }
}
```

Pada pesan masuk, data sensitif tidak diidentifikasi setelah operasi audit, dan `SNS:Publish` API pemanggil menerima kesalahan parameter tidak valid berikut ketika seluruh pesan sensitif.

```
Error code: AuthorizationError ...
```

Tolak operasi

Operasi Tolak mengganggu `Publish` API permintaan atau pengiriman pesan jika pesan berisi data sensitif. Objek operasi Deny kosong, karena tidak memerlukan konfigurasi tambahan.

```
"Operation": {
    "Deny": {}
}
```

Pada pesan masuk, `SNS:Publish` API penelepon menerima kesalahan otorisasi.

```
Error code: AuthorizationError ...
```

Pada pesan keluar, SNS topik Amazon tidak mengirimkan pesan ke langganan. Untuk melacak pengiriman yang tidak sah, aktifkan pencatatan status pengiriman topik. Berikut ini adalah contoh log status pengiriman:

```
{
    "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"
}
```

Contoh kebijakan perlindungan SNS data Amazon

Contoh berikut adalah kebijakan perlindungan data yang dapat Anda gunakan untuk mengaudit dan menolak data sensitif. Untuk tutorial lengkap yang menyertakan aplikasi contoh, lihat Memperkenalkan perlindungan data pesan untuk posting SNS blog Amazon.

Topik

- Contoh kebijakan untuk audit
- Contoh kebijakan dengan pernyataan topeng de-identifikasi inbound
- Contoh kebijakan dengan pernyataan redact de-identifikasi-inbound
- Contoh kebijakan dengan pernyataan topeng de-identifikasi keluar
- Contoh kebijakan dengan pernyataan redact de-identifikasi keluar
- Contoh kebijakan dengan pernyataan penolakan masuk
- Contoh kebijakan dengan pernyataan penolakan keluar

Contoh kebijakan untuk audit

Kebijakan audit memungkinkan Anda mengaudit hingga 99% pesan masuk dan mengirim temuan ke Amazon CloudWatch, Amazon Data Firehose, dan Amazon S3.

Misalnya, Anda dapat membuat kebijakan audit untuk mengevaluasi apakah ada sistem Anda yang secara tidak sengaja mengirim atau menerima data sensitif. Jika hasil audit Anda menunjukkan bahwa sistem mengirimkan informasi kartu kredit ke sistem yang tidak memerlukannya, Anda dapat menerapkan kebijakan perlindungan data untuk memblokir pengiriman data.

Contoh berikut mengaudit 99% pesan yang mengalir melalui topik dengan mencari nomor kartu kredit dan mengirimkan temuan ke CloudWatch Log, Firehose, dan Amazon S3.

Kebijakan perlindungan data:

```
{
  "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>"
            }
          }
        }
      }
    }
  ]
}
```

Contoh format hasil audit:

```
{
    "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"
}
```

Contoh kebijakan dengan pernyataan topeng de-identifikasi inbound

Contoh berikut mencegah pengguna memublikasikan pesan ke topik `CreditCardNumber` dengan menyembunyikan data sensitif dari konten pesan.

```
{
  "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": "#"
          }
        }
      }
    }
  ]
}
```

Contoh hasil topeng de-identifikasi masuk:

```
// original message
My credit card number is 4539894458086459
```

```
// delivered message
My credit card number is ################
```

Contoh kebijakan dengan pernyataan redact de-identifikasi-inbound

Contoh berikut mencegah pengguna memublikasikan pesan ke topik `CreditCardNumber` dengan menyunting data sensitif dari konten pesan.

```
{
  "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": {}
        }
      }
    }
  ]
}
```

Contoh hasil penyuntingan de-identifikasi masuk:

```
// original message
My credit card number is 4539894458086459

// delivered message
My credit card number is
```

Contoh kebijakan dengan pernyataan topeng de-identifikasi keluar

Contoh berikut mencegah pengguna menerima pesan `CreditCardNumber` dengan menyembunyikan data sensitif dari konten pesan.

```
{
  "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": "-"
          }
        }
      }
    }
  ]
}
```

Contoh hasil topeng de-identifikasi keluar:

```
// original message
My credit card number is 4539894458086459

// delivered message
My credit card number is ----------------
```

Contoh kebijakan dengan pernyataan redact de-identifikasi keluar

Contoh berikut mencegah pengguna menerima pesan `CreditCardNumber` dengan menyunting data sensitif dari konten pesan.

```
{
  "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": {}
        }
      }
    }
  ]
}
```

Contoh hasil penyuntingan de-identifikasi keluar:

```
// original message
My credit card number is 4539894458086459

// delivered message
My credit card number is
```

Contoh kebijakan dengan pernyataan penolakan masuk

Contoh berikut memblokir pengguna untuk memublikasikan pesan ke topik `CreditCardNumber` dengan konten pesan. Muatan yang ditolak dalam API respons memiliki kode status”403 AuthorizationError".

```
{
  "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": {}
      }
    }
  ]
}
```

Contoh kebijakan dengan pernyataan penolakan keluar

Contoh berikut memblokir AWS akun agar tidak menerima pesan yang berisi`CreditCardNumber`.

```
{
  "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": {}
      }
    }
  ]
}
```

Contoh hasil penolakan keluar, masuk ke Amazon 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"
}
```

Membuat kebijakan perlindungan data di Amazon SNS

Kebijakan perlindungan data membantu Anda melindungi data yang dipublikasikan ke SNS topik Amazon Anda dengan mengaudit, menghapus identifikasi (menutupi atau menyunting), dan menolak (memblokir) informasi sensitif yang berpindah antar aplikasi atau. Layanan AWS Anda dapat menggunakan AWS API, AWS CLI, AWS CloudFormation, atau AWS Management Console membuat kebijakan perlindungan data di AmazonSNS. Hanya satu kebijakan yang dapat ditentukan per SNS topik Amazon. Setiap kebijakan perlindungan data dapat memiliki satu atau lebih pernyataan de-identifikasi dan penolakan, tetapi hanya satu pernyataan audit.

Topik

- Membuat kebijakan perlindungan data di Amazon SNS menggunakan API
- Membuat kebijakan perlindungan data di Amazon SNS menggunakan CLI
- Membuat kebijakan perlindungan data di Amazon SNS menggunakan CloudFormation
- Membuat kebijakan perlindungan data di Amazon SNS menggunakan konsol
- Membuat kebijakan perlindungan SNS data Amazon untuk mengamankan data pesan menggunakan SDK

Membuat kebijakan perlindungan data di Amazon SNS menggunakan API

Jumlah dan ukuran sumber SNS daya Amazon dalam suatu AWS akun terbatas. Untuk informasi selengkapnya, lihat titik akhir dan kuota Amazon Simple Notification Service.

Membuat kebijakan perlindungan data menggunakan API

Anda dapat membuat kebijakan perlindungan SNS data Amazon menggunakan AWS API.

Untuk membuat kebijakan perlindungan data bersama dengan SNS topik Amazon (AWS API)

Gunakan `DataProtectionPolicy` properti SNS topik Amazon standar:

- `CreateTopic`

Untuk mengambil atau membuat kebijakan perlindungan data untuk SNS topik Amazon yang ada ()AWS API

Panggil salah satu operasi berikut ini:

- GetDataProtectionPolicy
- PutDataProtectionPolicy

Membuat kebijakan perlindungan data di Amazon SNS menggunakan CLI

Jumlah dan ukuran sumber SNS daya Amazon dalam suatu AWS akun terbatas. Untuk informasi selengkapnya, lihat titik akhir dan kuota Amazon Simple Notification Service.

Membuat kebijakan perlindungan data menggunakan AWS CLI

Anda dapat membuat kebijakan perlindungan SNS data Amazon menggunakan AWS Command Line Interface.

Untuk membuat kebijakan perlindungan data bersama dengan SNS topik Amazon (AWS CLI)

Gunakan opsi ini untuk membuat kebijakan perlindungan data baru bersama dengan SNS topik Amazon standar:

- buat-topik

Untuk membuat atau mengambil kebijakan perlindungan data untuk SNS topik Amazon yang ada ()AWS CLI

Panggil salah satu operasi berikut ini:

- get-data-protection-policy
- put-data-protection-policy

Membuat kebijakan perlindungan data di Amazon SNS menggunakan CloudFormation

Jumlah dan ukuran sumber SNS daya Amazon dalam AWS akun terbatas. Untuk informasi selengkapnya, lihat titik akhir dan kuota Amazon Simple Notification Service.

Membuat kebijakan perlindungan data (CloudFormation)

Anda dapat membuat kebijakan perlindungan SNS data Amazon menggunakan AWS CloudFormation.

Untuk membuat kebijakan perlindungan data bersama dengan SNS topik Amazon (CloudFormation)

Gunakan opsi ini untuk membuat kebijakan perlindungan data baru bersama dengan SNS topik Amazon standar:

- AWS::SNS: :Topik

Membuat kebijakan perlindungan data di Amazon SNS menggunakan konsol

Jumlah dan ukuran sumber SNS daya Amazon dalam AWS akun terbatas. Untuk informasi selengkapnya, lihat titik akhir dan kuota Amazon Simple Notification Service.

Untuk membuat kebijakan perlindungan data bersama dengan SNS topik Amazon (Konsol)

Gunakan opsi ini untuk membuat kebijakan perlindungan data baru bersama dengan SNS topik Amazon standar.

1. Masuk ke SNSkonsol Amazon.
2. Pilih topik atau buat yang baru. Untuk detail selengkapnya tentang membuat topik, lihat Membuat SNS topik Amazon.
3. Pada halaman Buat topik, di bagian Detail, pilih Standar.
   a. Masukkan Nama untuk topik.
   b. (Opsional) Masukkan Nama tampilan untuk topik.
4. Perluas kebijakan perlindungan data.
5. Pilih mode Konfigurasi:
   - Dasar — Tentukan kebijakan perlindungan data menggunakan menu sederhana.

- Advanced - Tentukan kebijakan perlindungan data kustom menggunakanJSON.

6. (Opsional) Untuk membuat pengidentifikasi data kustom Anda sendiri, perluas bagian Konfigurasi pengenal data kustom lakukan hal berikut:

   a. Masukkan nama unik untuk pengenal data kustom. Nama pengidentifikasi data kustom mendukung karakter alfanumerik, garis bawah (_), dan tanda hubung (-). Hingga 128 karakter didukung. Nama ini tidak dapat berbagi nama yang sama dengan pengenal data terkelola. Untuk daftar lengkap batasan pengenal data kustom, lihatKendala pengenal data kustom.

   b. Masukkan ekspresi reguler (RegEx) untuk pengidentifikasi data kustom. RegExmendukung karakter alfanumerik, karakter yang RegEx dicadangkan, dan simbol. RegEx memiliki panjang maksimum 200 karakter. Jika RegEx terlalu rumit, Amazon SNS akan gagal API menelepon. Untuk daftar lengkap RegEx batasan, lihatKendala pengenal data kustom.

   c. (Opsional) Pilih Tambahkan pengenal data khusus untuk menambahkan pengidentifikasi data tambahan sesuai kebutuhan. Maksimal 10 pengidentifikasi data kustom didukung untuk setiap kebijakan perlindungan data.

7. Pilih pernyataan yang ingin Anda tambahkan ke kebijakan perlindungan data Anda. Anda dapat menambahkan jenis pernyataan audit, de-identifikasi (menutupi atau menyunting), dan menolak (memblokir) ke kebijakan perlindungan data yang sama.

   a. Tambahkan pernyataan audit — Konfigurasikan data sensitif mana yang akan diaudit, berapa persentase pesan yang ingin Anda audit untuk data tersebut, dan ke mana harus mengirim log audit.

      > **Note**
      >
      > Hanya satu pernyataan audit yang diizinkan per kebijakan atau topik perlindungan data.

      i. Pilih pengidentifikasi data untuk menentukan data sensitif yang ingin Anda audit.

      ii. Untuk tingkat sampel Audit, masukkan persentase pesan yang akan diaudit untuk informasi sensitif, hingga maksimum 99%.

iii. Untuk tujuan Audit, pilih yang Layanan AWS akan mengirim hasil pencarian audit, dan masukkan nama tujuan untuk setiap Layanan AWS yang Anda gunakan. Anda dapat memilih dari Amazon Web Services berikut:

- Amazon CloudWatch - CloudWatch Log adalah solusi logging AWS standar. Menggunakan CloudWatch Log, Anda dapat melakukan analisis log menggunakan Wawasan Log (lihat contoh di sini) dan membuat metrik dan alarm. CloudWatch Log adalah tempat banyak layanan menerbitkan log, yang membuatnya lebih mudah untuk menggabungkan semua log menggunakan satu solusi. Untuk informasi tentang Amazon CloudWatch, lihat Panduan CloudWatch Pengguna Amazon.
- Amazon Data Firehose — Firehose memenuhi permintaan streaming real-time ke Splunk,, dan OpenSearch Amazon Redshift untuk analisis log lebih lanjut. Untuk informasi tentang Amazon Data Firehose, lihat Panduan Pengguna Amazon Data Firehose.
- Amazon Simple Storage Service - Amazon S3 adalah tujuan log ekonomis untuk tujuan arsip. Anda mungkin diminta untuk menyimpan log untuk jangka waktu bertahun-tahun. Dalam hal ini, Anda dapat memasukkan log ke Amazon S3 untuk menghemat biaya. Untuk informasi tentang Amazon Simple Storage Service, lihat Panduan Pengguna Amazon Simple Storage Service.

b. Tambahkan pernyataan de-identifikasi — Konfigurasikan data sensitif yang ingin Anda de-identifikasi dalam pesan, apakah Anda ingin menutupi atau menyunting data tersebut, dan akun untuk menghentikan pengiriman data tersebut.

i. Untuk pengidentifikasi Data, pilih data sensitif yang ingin Anda de-identifikasi.

ii. Untuk Tentukan pernyataan de-identifikasi ini, pilih AWS akun atau IAM prinsip tempat pernyataan de-identifikasi ini berlaku. Anda dapat menerapkannya ke semua AWS akun, atau ke AWS akun atau IAMentitas tertentu (akar akun, peran, atau pengguna) yang menggunakan akun IDs atau IAM entitasARNs. Pisahkan beberapa IDs atau ARNs menggunakan koma (,).

IAMPrinsipal berikut didukung:

- IAMprinsipal akun — Misalnya,. `arn:aws:iam::AWS-account-ID:root`
- IAMprinsip peran — Misalnya,. `arn:aws:iam::AWS-account-ID:role/role-name`

- IAMprinsip pengguna — Misalnya,. `arn:aws:iam::AWS-account-ID:user/user-name`

iii. Untuk De-identify Option, pilih bagaimana Anda ingin menghapus identifikasi data sensitif. Opsi berikut didukung:

- Redact - Menghapus data sepenuhnya. Misalnya, email: `classified@amazon.com` menjadi email: .
- Mask — Mengganti data dengan karakter tunggal. Misalnya, email: `classified@amazon.com` menjadi email:`**********`.

iv. (Opsional) Lanjutkan untuk menambahkan pernyataan de-identifikasi sesuai kebutuhan.

c. Tambahkan pernyataan penolakan — Konfigurasikan data sensitif mana yang mencegah agar tidak bergerak melalui topik Anda, dan prinsip mana yang harus mencegah pengiriman data tersebut.

i. Untuk arah data, pilih arah pesan untuk pernyataan penolakan:

- Pesan masuk — Terapkan pernyataan penolakan ini ke pesan yang dikirim ke topik.
- Pesan keluar — Terapkan pernyataan penolakan ini ke pesan yang disampaikan topik ke titik akhir langganan.

ii. Pilih pengidentifikasi data untuk menentukan data sensitif yang ingin Anda tolak.

iii. Pilih IAMprinsip yang berlaku untuk pernyataan penolakan ini. Anda dapat menerapkannya ke semua AWS akun, untuk spesifik Akun AWS, atau IAMentitas (misalnya, akar akun, peran, atau pengguna) yang menggunakan akun IDs atau IAM entitasARNs. Pisahkan beberapa IDs atau ARNs menggunakan koma (,). IAMPrinsipal berikut didukung:

- IAMprinsipal akun — Misalnya,. `arn:aws:iam::AWS-account-ID:root`
- IAMprinsip peran — Misalnya,. `arn:aws:iam::AWS-account-ID:role/role-name`
- IAMprinsip pengguna — Misalnya,. `arn:aws:iam::AWS-account-ID:user/user-name`

iv. (Opsional) Terus tambahkan pernyataan penolakan sesuai kebutuhan.

Membuat kebijakan perlindungan SNS data Amazon untuk mengamankan data pesan menggunakan SDK

Jumlah dan ukuran sumber SNS daya Amazon dalam AWS akun terbatas. Untuk informasi selengkapnya, lihat titik akhir dan kuota Amazon Simple Notification Service.

Membuat kebijakan perlindungan data menggunakan AWS SDK

Anda dapat membuat kebijakan perlindungan SNS data Amazon menggunakan AWS SDK.

Untuk membuat kebijakan perlindungan data bersama dengan SNS topik Amazon (AWS SDK)

Gunakan opsi berikut untuk membuat kebijakan perlindungan data baru bersama dengan SNS topik Amazon standar:

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();
```

Untuk membuat atau mengambil kebijakan perlindungan data untuk SNS topik Amazon yang ada ()AWS SDK

Gunakan opsi berikut untuk membuat atau mengambil kebijakan perlindungan data baru bersama dengan SNS topik Amazon standar:

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();
```

Menghapus kebijakan perlindungan data di Amazon SNS

Anda dapat menghapus kebijakan perlindungan SNS data Amazon menggunakan AWS API, AWS CLI, AWS CloudFormation, atau AWS Management Console.

Untuk informasi umum tentang kebijakan perlindungan SNS data Amazon, lihatMemahami kebijakan perlindungan SNS data Amazon.

Jumlah dan ukuran sumber daya kebijakan perlindungan SNS data Amazon dalam AWS akun terbatas. Untuk informasi selengkapnya, lihat Amazon SNS API melambat. Referensi Umum AWS

Topik

- Menghapus kebijakan perlindungan data menggunakan konsol
- Menghapus kebijakan perlindungan data menggunakan string kosong JSON
- Menghapus kebijakan perlindungan data menggunakan AWS CLI

Menghapus kebijakan perlindungan data menggunakan konsol

Untuk menghapus kebijakan perlindungan data terkelola menggunakan konsol

1. Masuk ke SNSkonsol Amazon.
2. Pilih topik yang berisi kebijakan perlindungan data yang ingin Anda hapus.
3. Pilih Edit.
4. Perluas bagian Kebijakan Perlindungan Data.
5. Pilih Hapus di samping pernyataan kebijakan perlindungan data yang ingin Anda hapus.
6. Pilih Simpan perubahan.

Menghapus kebijakan perlindungan data menggunakan string kosong JSON

Anda dapat menghapus kebijakan perlindungan data dengan memperbaruinya ke JSON string kosong.

Menghapus kebijakan perlindungan data menggunakan AWS CLI

Anda dapat menghapus kebijakan perlindungan data menggunakan AWS CLI.

```
//aws sns put-data-protection-policy --resource-arn topic-arn --data-
protection-policy ""
```

Pengidentifikasi SNS data Amazon

Amazon SNS menggunakan kombinasi kriteria dan teknik, termasuk pembelajaran mesin dan pencocokan pola, untuk mendeteksi data sensitif. Kriteria dan teknik ini, secara kolektif disebut sebagai pengidentifikasi data, dapat mendeteksi daftar tipe data sensitif yang besar dan terus bertambah untuk banyak negara dan wilayah. Pengidentifikasi data SNS terkelola Amazon menawarkan tipe data yang telah dikonfigurasi sebelumnya untuk melindungi data keuangan, informasi kesehatan pribadi (PHI), dan informasi identitas pribadi (). PII Anda juga dapat menggunakan pengidentifikasi data khusus untuk membuat pengidentifikasi data Anda sendiri yang disesuaikan dengan kasus penggunaan spesifik Anda.

Topik

- Menggunakan pengidentifikasi data terkelola di Amazon SNS

- Menggunakan pengidentifikasi data khusus di Amazon SNS

Menggunakan pengidentifikasi data terkelola di Amazon SNS

Topik

- Apa itu pengidentifikasi data terkelola?
- Jenis data SNS sensitif Amazon: Kredensyal
- Jenis data SNS sensitif Amazon: Perangkat
- Jenis data SNS sensitif Amazon: Keuangan
- Jenis data SNS sensitif Amazon: Informasi kesehatan yang dilindungi (PHI)
- Jenis data SNS sensitif Amazon: Informasi yang dapat diidentifikasi secara pribadi () PII

Apa itu pengidentifikasi data terkelola?

Pengidentifikasi data SNS terkelola Amazon dirancang untuk mendeteksi jenis data sensitif tertentu, seperti nomor kartu kredit, kunci akses AWS rahasia, atau nomor paspor untuk negara atau wilayah tertentu. Saat membuat kebijakan perlindungan data, Anda dapat mengonfigurasi Amazon SNS untuk menggunakan pengidentifikasi ini untuk menganalisis pesan yang membahas topik, dan mengambil tindakan saat terdeteksi.

Amazon SNS dapat mendeteksi kategori data sensitif berikut dengan menggunakan pengidentifikasi data terkelola:

- Kredensyal, seperti kunci pribadi atau kunci akses AWS rahasia
- Pengidentifikasi perangkat, seperti alamat IP atau MAC alamat
- Informasi keuangan, seperti nomor kartu kredit
- Informasi Kesehatan, PHI seperti asuransi kesehatan atau nomor identifikasi medis
- Informasi pribadi, PII seperti SIM atau nomor jaminan sosial

Dalam setiap kategori, Amazon SNS dapat mendeteksi beberapa jenis data sensitif. Topik di bagian ini mencantumkan dan menjelaskan setiap jenis dan persyaratan yang relevan untuk mendeteksinya. Untuk setiap jenis, mereka juga menunjukkan pengenal unik (ID) untuk pengidentifikasi data terkelola yang dirancang untuk mendeteksi data. Saat membuat kebijakan perlindungan data, Anda dapat menggunakan ID ini untuk menyertakan pengenal data terkelola untuk mendeteksi perlindungan data pesan.

Persyaratan kata kunci

Untuk mendeteksi jenis data sensitif tertentu, Amazon SNS memindai kata kunci di dekat data. Jika kasus ini adalah untuk tipe data tertentu, topik berikutnya di bagian ini menunjukkan persyaratan kata kunci tertentu untuk data tersebut.

Kata kunci tidak sensitif terhadap kasus. Selain itu, jika kata kunci berisi spasi, Amazon SNS secara otomatis mencocokkan variasi kata kunci yang tidak berisi spasi, atau berisi garis bawah (_) atau tanda hubung (-) alih-alih spasi. Dalam kasus tertentu, Amazon SNS juga memperluas atau menyingkat kata kunci untuk mengatasi variasi umum kata kunci.

Amazon SNS mengelola pengidentifikasi data untuk tipe data sensitif

Tabel berikut mencantumkan dan menjelaskan jenis informasi kredensi, perangkat, keuangan, medis, dan kesehatan pribadi (PHI) yang SNS dapat dideteksi Amazon menggunakan pengidentifikasi data terkelola. Ini adalah tambahan untuk jenis data tertentu yang mungkin juga memenuhi syarat sebagai informasi identitas pribadi (). PII

Pengidentifikasi data yang bergantung pada wilayah memerlukan nama pengidentifikasi dengan tanda hubung, dan kode dua huruf (ISO3166-1 alfa-2). Misalnya, DriversLicense -US.

Pengidentifikasi	Kategori	Negara/Bahasa
BankAccountNumber	Keuangan	DE, ES, FR, GB, ITU
CepCode	Pribadi	BR
Cnpj	Pribadi	BR
CpfCode	Pribadi	BR
DriversLicense	Pribadi	DI, AU, BE, BG, CA, CY, CZ, DE, DK, E, ES, FI, FR, GB, GR, HR, HU, YAITU, ITU, LT, LU, LV, MT, NL, PL, PT, RO, SE, SI, SK, US
DrugEnforcementAgencyNumber	Kondisi	AS

Pengidentifikasi	Kategori	Negara/Bahasa
ElectoralRollNumber	Pribadi	GB
HealthInsuranceCardNumber	Kondisi	EU
HealthInsuranceClaimNumber	Kondisi	AS
HealthInsuranceNumber	Kondisi	FR
HealthcareProcedureCode	Kondisi	AS
IndividualTaxIdentification Number	Pribadi	AS
InseeCode	Pribadi	FR
MedicareBeneficiaryNumber	Kondisi	AS
NationalDrugCode	Kondisi	AS
NationalIdentificationNumber	Pribadi	DE, ES, ITU
NationalInsuranceNumber	Pribadi	GB
NationalProviderId	Kondisi	AS
NhsNumber	Kondisi	GB
NieNumber	Pribadi	ES
NifNumber	Pribadi	ES
PassportNumber	Pribadi	CA, DE, ES, FR, GB, ITU, KAMI
PermanentResidenceNumber	Pribadi	CA
PersonalHealthNumber	Kondisi	CA
PhoneNumber	Pribadi	BR, DE, ES, FR, GB, ITU, KAMI

Pengidentifikasi	Kategori	Negara/Bahasa
PostalCode	Pribadi	CA
RgNumber	Pribadi	BR
SocialInsuranceNumber	Pribadi	CA
Ssn	Pribadi	ES, KITA
TaxId	Pribadi	DE, ES, FR, GB
ZipCode	Pribadi	AS

Pengenal yang Didukung yang independen bahasa/wilayah

Pengidentifikasi	Kategori
Alamat	Pribadi
AwsSecretKey	Kredensial
CreditCardExpiration	Keuangan
CreditCardNumber	Keuangan
CreditCardSecurityCode	Keuangan
EmailAddress	Pribadi
IpAddress	Pribadi
LatLong	Pribadi
Nama	Pribadi
OpenSshPrivateKey	Kredensial
PgpPrivateKey	Kredensial

Pengidentifikasi	Kategori
PkcsPrivateKey	Kredensial
PuttyPrivateKey	Kredensial
VehicleIdentificationNumber	Pribadi

Jenis data SNS sensitif Amazon: Kredensyal

Tabel berikut mencantumkan dan menjelaskan jenis kredensil yang SNS dapat dideteksi Amazon menggunakan pengidentifikasi data terkelola.

Tipe Deteksi	ID pengenal data terkelola	Diperlukan kata kunci	Negara dan wilayah
AWS kunci akses rahasia	AwsSecretKey	aws_secret_access_key, credentials, secret access key, secret key, set-awscredential	Setiap
Buka kunci SSH pribadi	OpenSshPrivateKey	Tidak	Setiap
PGPkunci pribadi	PgpPrivateKey	Tidak	Setiap
Kunci pribadi Standar Kriptografi Kunci Publik () PKCS	PkcsPrivateKey	Tidak	Setiap
Kunci TTY pribadi Pu	PuttyPrivateKey	Tidak	Setiap

Pengidentifikasi data ARNs untuk tipe data kredensyal

Berikut ini mencantumkan Nama Sumber Daya Amazon (ARNs) untuk pengidentifikasi data yang dapat Anda tambahkan ke kebijakan perlindungan data Anda.

Pengenal data kredensi ARNs
arn:aws:dataprotection: :aws:data-identifier/ AwsSecretKey
arn:aws:dataprotection: :aws:data-identifier/ OpenSshPrivateKey
arn:aws:dataprotection: :aws:data-identifier/ PgpPrivateKey
arn:aws:dataprotection: :aws:data-identifier/ PkcsPrivateKey
arn:aws:dataprotection: :aws:data-identifier/ PuttyPrivateKey

Jenis data SNS sensitif Amazon: Perangkat

Tabel berikut mencantumkan dan menjelaskan jenis pengidentifikasi perangkat yang SNS dapat dideteksi Amazon menggunakan pengidentifikasi data terkelola.

Tipe Deteksi	ID pengenal data terkelola	Diperlukan kata kunci	Negara dan wilayah
Alamat IP	IpAddress	Tidak	Setiap

Pengidentifikasi data ARNs untuk tipe data perangkat

Berikut ini mencantumkan Nama Sumber Daya Amazon (ARNs) untuk pengidentifikasi data yang dapat Anda tambahkan ke kebijakan perlindungan data Anda.

Pengidentifikasi data perangkat ARN
arn:aws:dataprotection: :aws:data-identifier/ IpAddress

Jenis data SNS sensitif Amazon: Keuangan

Tabel berikut mencantumkan dan menjelaskan jenis informasi keuangan yang SNS dapat dideteksi Amazon menggunakan pengidentifikasi data terkelola.

Tipe Deteksi	ID pengenal data terkelola	Diperlukan kata kunci	Informasi tambahan	Negara dan wilayah
Nomor rekening bank	BankAccountNumber BankAccountNumber-US	Ya, lihatKata kunci untuk nomor rekening bank.	Ini termasuk: Nomor Rekening Bank Internasional (IBANs) yang terdiri dari hingga 34 karakter alfanumerik, termasuk elemen seperti kode negara.	Prancis, Jerman, Italia, Spanyol, Inggris
Tanggal kedaluwarsa kartu kredit	CreditCardExpiration	exp d, exp m, exp y, kedaluwarsa, kedaluwarsa	–	Setiap
Data strip magnetik kartu kredit	CreditCardMagneticStripe	Ya, termasuk: data kartu, iso7813, mag, magstripe, stripe, swipe.	Hal ini mencakup lintasan 1 dan 2.	Setiap
Nomor kartu kredit	CreditCardNumber	nomor rekening, american express, amex, kartu bank, kartu, nomor kartu, nomor kartu, cc #, ccn, kartu cek, kredit, kartu kredit #, dankort, debit, kartu debit, klub pengunjun	Deteksi mengharuskan data menjadi urutan 13-19 digit yang mematuhi rumus pemeriksaan Luhn, dan menggunakan awalan nomor kartu standar untuk salah	Setiap

Tipe Deteksi	ID pengenal data terkelola	Diperlukan kata kunci	Informasi tambahan	Negara dan wilayah
		g, temukan, elektron, kode verifikasi elo, biro kartu jepang, jcb, mastercar d, mc, pan, nomor rekening pembayaran, nomor kartu pembayaran, pcn, pembayara n serikat, visa	satu jenis kartu kredit berikut: American Express, Dankort, Diner's Club, Discover, Electron, Japanese Card Bureau (JCB), Mastercard,, dan Visa (tautan superskrip di bawah 1). UnionPay	
Kode verifikasi kartu kredit	CreditCar dSecurityCode	id kartu, kode identifikasi kartu, nomor identifik asi kartu, kode keamanan kartu, kode validasi kartu, nomor validasi kartu, data verifikas i kartu, nilai verifikasi kartu, cvc, cvc2, cvv, cvv2, kode verifikasi elo	–	Setiap

1. Amazon SNS tidak melaporkan kejadian urutan berikut, yang telah disediakan oleh penerbit kartu kredit untuk pengujian publik:

122000000000003, 2222405343248877, 2222990905257051, 2223007648726984, 2223577120017656, 3056930909025904, 343434343434343434, 3528000700000000, 3530111333300000, 3566002020360505, 3614898900647913, 36700102000000, 371449635398431, 378282246310005, 378734493671000, 38520000023237, 4012888888881881, 4111111111111111, 4222222222222, 444433332222111111, 4462030000000000, 42222221111 484070000000000, 4911830000000, 4917300800000000, 4917610000000000, 49176100000000003, 5019717010103742, 5105105105105105100, 5111010030175156, 5185540810000019, 520082828282828210, 520423000000 80000017, 5204740009900014, 5420923878724339, 54545454545454545454, 5455330760000018, 5506900490000436, 5506900490000444, 5506900510000234, 5506920809243667, 5506922400634930, 55069692727427317625, 5553042241984105, 5555553753048194, 555555555555554444, 5610591081018250, 6011000990139424, 6011000400000000, 601111111111111117, 630490017740292441, 630495060000000000, 630495060000000000, 630490017740292441, 630495060000000000, 630495060000000000, 630490017740292441 331101999990016, 6759649826438453, 6799990100000000019, dan 76009244561.

Kata kunci untuk nomor rekening bank

Gunakan kata kunci berikut untuk mendeteksi Nomor Rekening Bank Internasional (IBANs) yang terdiri dari hingga 34 karakter alfanumerik, termasuk elemen seperti kode negara.

Negara atau wilayah	Kata kunci			
France	account code, account number, accountno #, accountnu mber#, bban, code bancaire, compte bancaire, customer account id, customer account number, customer bank			

Negara atau wilayah	Kata kunci			
	account id, iban, numéro de compte			
Germany	account code, account number, accountno #, accountnu mber#, bankleitz ahl, bban, customer account id, customer account number, customer bank account id, geheimzahl, iban, kartennum mer, kontonumm er, kreditkar tennummer, sepa			

Negara atau wilayah	Kata kunci			
Italy	account code, account number, accountno #, accountnu mber#, bban, codice bancario, conto bancario, customer account id, customer account number, customer bank account id, iban, numero di conto			

Negara atau wilayah	Kata kunci			
Spain	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			
UK	account code, account number, accountno #, accountnu mber#, bban, customer account id, customer account number, customer bank account id, iban, sepa			

Negara atau wilayah	Kata kunci			
US	rekening bank, rekening bank, rekening giro, cek acct, rekening deposito, setoran, rekening tabungan, rekening tabungan, rekening cek, cek acct			

Pengidentifikasi data ARNs untuk tipe data keuangan

Berikut ini mencantumkan Nama Sumber Daya Amazon (ARNs) untuk pengidentifikasi data yang dapat Anda tambahkan ke kebijakan perlindungan data Anda.

Pengidentifikasi data keuangan ARNs
arn:aws:dataprotection: :aws:data-identifier/ -DE BankAccountNumber
arn:aws:dataprotection: :aws:data-identifier/ -ES BankAccountNumber
arn:aws:dataprotection: :aws:data-identifier/ -FR BankAccountNumber
arn:aws:dataprotection: :aws:data-identifier/ -GB BankAccountNumber
arn:aws:dataprotection: :aws:data-identifier/ -IT BankAccountNumber
arn:aws:dataprotection: :aws:data-identifier/ -US BankAccountNumber
arn:aws:dataprotection: :aws:data-identifier/ CreditCardExpiration

Pengidentifikasi data keuangan ARNs
arn:aws:dataprotection: :aws:data-identifier/ CreditCardNumber
arn:aws:dataprotection: :aws:data-identifier/ CreditCardSecurityCode

Jenis data SNS sensitif Amazon: Informasi kesehatan yang dilindungi (PHI)

Tabel berikut mencantumkan dan menjelaskan jenis informasi kesehatan yang dilindungi (PHI) yang SNS dapat dideteksi Amazon menggunakan pengidentifikasi data terkelola.

Tipe Deteksi	ID pengenal data terkelola	Diperlukan kata kunci	Negara dan wilayah
Nomor Registrasi Badan Penegakan Narkoba (DEA)	DrugEnforcementAgencyNumber	dea number, dea registration	AS
Nomor Kartu Asuransi Kesehatan (EHIC)	HealthInsuranceCardNumber	nomor bantuan kebersihan, carta assicurazione number, carte d'assurance maladie, carte européenne d'assurance maladie, ceam, ehic, ehic#, finlandehicnumber #, kartu kesehatan , kartu kesehatan, kartu kesehatan, kartu asuransi kesehatan , nomor asuransi kesehatan, nomor asuransi kesehatan, nomor kartu asuransi, nomor kartu asuransi, kartu asuransi kartu	EU

Tipe Deteksi	ID pengenal data terkelola	Diperlukan kata kunci	Negara dan wilayah
		kredit, nomor layanan kesehatan, nomor rekening medis, nomor 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, sairausvakuutuskortti, nomor sairausvakuutusnumero, nomor sjukförsärsärsskort, suomi ehic-numero, tarjeta de salud, terveyskortti, tessera sanitaria assicurazione number, versicherer nomor	
Nomor Klaim Asuransi Kesehatan (HICN)	HealthInsuranceClaimNumber	health insurance claim number, hic no, hic no., hic number, hic#, hicn, hicn#., hicno#	AS
Nomor asuransi kesehatan atau nomor identifikasi medis	HealthInsuranceNumber	carte d'assuré social, carte vitale, insurance card	FR

Tipe Deteksi	ID pengenal data terkelola	Diperlukan kata kunci	Negara dan wilayah
Kode Sistem Pengkodean Prosedur Umum Kesehatan (HCPCS)	HealthcareProcedureCode	current procedural terminology, hcpcs, healthcare common procedure coding system	AS
Nomor Penerima Medicare () MBN	MedicareBeneficiaryNumber	mbi, medicare beneficiary	AS
Kode Obat Nasional (NDC)	NationalDrugCode	national drug code, ndc	AS
Pengidentifikasi Penyedia Nasional () NPI	NationalProviderId	hipaa, n.p.i, national provider, npi	AS
Nomor Pelayanan Kesehatan Nasional (NHS)	NhsNumber	national health service, NHS	GB
Nomor Kesehatan Pribadi (PHN)	PersonalHealthNumber	canada healthcare number, msp number, personal healthcare number, phn, soins de santé	CA

Kata kunci untuk nomor asuransi kesehatan dan nomor identifikasi medis

Untuk mendeteksi berbagai jenis asuransi kesehatan dan nomor identifikasi medis, Amazon SNS membutuhkan kata kunci untuk berada di dekat angka-angka tersebut. Ini termasuk nomor Kartu Asuransi Kesehatan Eropa (UE, Finlandia), nomor asuransi kesehatan (Prancis), Pengidentifikasi Penerima Medicare (AS), nomor Asuransi Nasional (Inggris), NHS nomor (Inggris), dan Nomor Kesehatan Pribadi (Kanada).

Tabel berikut mencantumkan kata kunci yang SNS diakui Amazon untuk negara dan wilayah tertentu.

Negara atau wilayah	Kata kunci
Canada	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 kuutuskortti, sairausvakuutusnumero, sjukförsäkring nummer, sjukförsäkringskort, suomi ehic-numero, tarjeta de salud, terveyskortti, tessera sanitaria assicurazione numero, versicherungsnummer
Finland	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

Negara atau wilayah	Kata kunci
France	carte d'assuré social, carte vitale, insurance card
UK	layanan kesehatan nasional, NHS
AS	mbi, medicare beneficiary

Pengidentifikasi data ARNs untuk tipe data informasi kesehatan yang dilindungi () PHI

Berikut ini mencantumkan pengenal data Amazon Resource Names (ARNs) yang dapat digunakan dalam kebijakan perlindungan PHI data.

PHIpengenal data ARNs
arn:aws:dataprotection: :aws:data-identifier/ -US DrugEnforcementAgencyNumber
arn:aws:dataprotection: :aws:data-identifier/ -US HealthcareProcedureCode
arn:aws:dataprotection: :aws:data-identifier/ -EU HealthInsuranceCardNumber
arn:aws:dataprotection: :aws:data-identifier/ -US HealthInsuranceClaimNumber
arn:aws:dataprotection: :aws:data-identifier/ -FR HealthInsuranceNumber
arn:aws:dataprotection: :aws:data-identifier/ -US MedicareBeneficiaryNumber
arn:aws:dataprotection: :aws:data-identifier/ -US NationalDrugCode
arn:aws:dataprotection: :aws:data-identifier/ -GB NationalInsuranceNumber
arn:aws:dataprotection: :aws:data-identifier/ -US NationalProviderId
arn:aws:dataprotection: :aws:data-identifier/ -GB NhsNumber
arn:aws:dataprotection: :aws:data-identifier/ -CA PersonalHealthNumber

Jenis data SNS sensitif Amazon: Informasi yang dapat diidentifikasi secara pribadi () PII

Tabel berikut mencantumkan dan menjelaskan jenis informasi yang dapat diidentifikasi secara pribadi (PII) yang dapat dideteksi Amazon SNS menggunakan pengidentifikasi data terkelola.

Tipe Deteksi	ID pengenal data terkelola	Diperlukan kata kunci	Informasi tambahan	Negara dan wilayah
Tanggal lahir	DateOfBirth	dob, date of birth, birthdate , birth date, birthday, b-day, bday	Support mencakup sebagian besar format tanggal, seperti semua digit dan kombinasi digit dan nama bulan. Komponen tanggal dapat dipisahkan oleh spasi, garis miring (/), atau tanda hubung (-).	Setiap
Kode Pos Endereçamento () CEP	CepCode	cep, código de endereçamento postal, codigo de endereçamento postal	–	Brazil
Kadastro Nacional da Pessoa Jurídica () CNPJ	Cnpj	cadastro nacional da pessoa jurídica, cadastro nacional da pessoa juridica, cnpj	–	Brazil

Tipe Deteksi	ID pengenal data terkelola	Diperlukan kata kunci	Informasi tambahan	Negara dan wilayah
Kadaster de Pessoas Físicas () CPF	CpfCode	Cadastro de pessoas fisicas, cadastro de pessoas físicas, cadastro de pessoa física, cadastro de pessoa fisica, cpf	–	Brazil
Nomor identifikasi lisensi	DriversLicense	Ya, lihatKata kunci untuk nomor identifikasi surat izin mengemudi.	–	Australia, Austria, Belgium, Bulgaria, Canada, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, UK, US

Tipe Deteksi	ID pengenal data terkelola	Diperlukan kata kunci	Informasi tambahan	Negara dan wilayah
Nomor Electoral roll	Electoral RollNumber	electoral#, electoral #, electoralnumber, electoral number, electoralroll#, electoral roll#, electoral roll #, electoral roll no., electoral roll number, electoralrollno	–	UK
Identifikasi wajib pajak peroranga n	Individua lTaxIdentification Number	Ya, lihatKata kunci untuk nomor pokok wajib pajak.	–	AS
Institut Nasional untuk Statistik dan Studi Ekonomi (INSEE)	InseeCode	Ya, lihatKata kunci untuk nomor induk kependudukan.	–	France

Tipe Deteksi	ID pengenal data terkelola	Diperlukan kata kunci	Informasi tambahan	Negara dan wilayah
Nomor identifikasi nasional	NationalIdentificationNumber	Ya, lihatKata kunci untuk nomor induk kependudukan.	Ini termasuk pengidentifikasi Documento Nacional de Identidad (SpanyolDNI), kode fiskal Codice (Italia), dan nomor Kartu Identitas Nasional (Jerman).	Jerman, Italia, Spanyol
Nomor Asuransi Nasional (NINO)	NationalInsuranceNumber	insurance no., insurance number, insurance #, national insurance number, nationalinsurance#, nationalinsurancenumber, nin, nino	–	UK
Nama Identidad de extranjero () NIE	NieNumber	Ya, lihatKata kunci untuk nomor pokok wajib pajak.	–	Spain
Nomor Identifikasi Fiskal () NIF	NifNumber	Ya, lihatKata kunci untuk nomor pokok wajib pajak.	–	Spain

Tipe Deteksi	ID pengenal data terkelola	Diperlukan kata kunci	Informasi tambahan	Negara dan wilayah
Nomor paspor	PassportNumber	Ya, lihatKata kunci untuk nomor paspor.	–	Canada, France, Germany, Italy, Spain, UK, US
Nomor tempat tinggal permanen	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	–	Kanada

Tipe Deteksi	ID pengenal data terkelola	Diperlukan kata kunci	Informasi tambahan	Negara dan wilayah
Nomor telepon	PhoneNumber	Brasil: kata kunci juga meliputi: cel, celular, fone, móvel, número residencial, numero residencial, telefone Lainnya: sel, kontak, faks, nomor faks, ponsel, telepon, nomor telepon, telp, telepon, nomor telepon	Ini termasuk nomor bebas pulsa di US dan nomor fax. Jika kata kunci berada di dekat data, nomor tersebut tidak harus menyertakan kode negara. Jika kata kunci tidak dekat dengan data, nomor tersebut harus menyertakan kode negara.	Brazil, Canada, France, Germany, Italy, Spain, UK, US
Kode Pos	PostalCode	No	–	Kanada
Registrasi Geral (RG)	RgNumber	Ya, lihatKata kunci untuk nomor induk kependudukan.	–	Brazil
Nomor Asuransi Sosial (SIN)	SocialInsuranceNumber	canadian id, numéro d'assurance sociale, social insurance number, sin	–	Kanada

Tipe Deteksi	ID pengenal data terkelola	Diperlukan kata kunci	Informasi tambahan	Negara dan wilayah
Nomor Jaminan Sosial (SSN)	Ssn	Spanyol - número de la security sosial, jaminan sosial no., jaminan sosial no. número de la security sosial, nomor jaminan sosial, socialsec urityno #, ssn, ssn # AS - jaminan sosial, ss#, ssn	–	Spain, US
Nomor pokok wajib pajak	TaxId	Ya, lihatKata kunci untuk nomor pokok wajib pajak.	Ini termasuk TIN (Prancis); Steueridentifikati onsnummer (Jerman); (Spanyol); dan, CIF (Inggris). TRN UTR	Prancis, Jerman, Spanyol, Inggris
Kode pos US	ZipCode	zip code, zip+4	–	AS

Tipe Deteksi	ID pengenal data terkelola	Diperlukan kata kunci	Informasi tambahan	Negara dan wilayah
Alamat surat-menyurat	Address	No	Meskipun kata kunci tidak diperlukan, deteksi memerlukan alamat untuk menyertakan nama kota atau tempat dan ZIP atau Kode Pos.	Australia, Canada, France, Germany, Italy, Spain, UK, US
Alamat surat elektronik	EmailAddress	email, alamat email, email, alamat email	–	Setiap

Tipe Deteksi	ID pengenal data terkelola	Diperlukan kata kunci	Informasi tambahan	Negara dan wilayah
Koordinat Sistem Pemosisian Global (GPS)	LatLong	coordinate, coordinates, lat long, latitude longitude, location, position	Amazon SNS dapat mendeteksi GPS koordinat jika koordinat lintang dan bujur disimpan sebagai pasangan dan mereka dalam format Derajat Desimal (DD), misalnya, 41.948614, -87.655311. Support tidak menyertakan koordinat dalam format Derajat Desimal Menit (DDM), misalnya format 41° 56.9168'N 87° 39.3187'W, atau Derajat, Menit, Detik () format, misalnya 41° 56'55.0104 “N 87° 39'19.1196" W. DMS	Setiap

Tipe Deteksi	ID pengenal data terkelola	Diperlukan kata kunci	Informasi tambahan	Negara dan wilayah
Nama lengkap	Name	No	Amazon hanya SNS dapat mendeteksi nama lengkap. Dukungan terbatas pada set karakter Latin.	Setiap
Nomor identifikasi kendaraan (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 SNS dapat mendeteksi VINs yang terdiri dari urutan 17 karakter dan mematuhi standar ISO 3779 dan 3780. Standar ini dirancang untuk penggunaan di seluruh dunia.	Setiap

Kata kunci untuk nomor identifikasi surat izin mengemudi

Untuk mendeteksi berbagai jenis nomor identifikasi SIM, Amazon SNS memerlukan kata kunci untuk berada di dekat nomor. Tabel berikut mencantumkan kata kunci yang SNS diakui Amazon untuk negara dan wilayah tertentu.

Negara atau wilayah	Kata kunci
Australia	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
Austria	führerschein, fuhrerschein, führerschein republik österreich, fuhrerschein republik osterreich
Belgium	fuehrerschein, fuehrerschein- nr, fuehrersc heinnummer, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrersch einnummer, führerscheinnummer, numéro permis conduire, permis de conduire, rijbewijs, rijbewijsnummer
Bulgaria	превозно средство, свидетелство за управление на моторно, свидетелство за управление на мпс, сумпс, шофьорска книжка
Canada	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
Croatia	vozačka dozvola
Cyprus	άδεια οδήγησης

Negara atau wilayah	Kata kunci
Czech Republic	číslo licence, číslo licence řidiče, číslo řidičskéh o průkazu, ovladače lic., povolení k jízdě, povolení řidiče, řidiči povolení, řidičský prúkaz, řidičský průkaz
Denmark	kørekort, kørekortnummer
Estonia	juhi litsentsi number, juhiloa number, juhiluba, juhiluba number
Finland	ajokortin numero, ajokortti, förare lic., körkort, körkort nummer, kuljettaja lic., permis de conduire
France	permis de conduire
Germany	fuehrerschein, fuehrerschein- nr, fuehrersc heinnummer, fuhrerschein, führerschein, fuhrerschein- nr, führerschein- nr, fuhrersch einnummer, führerscheinnummer
Greece	δεια οδήγησης, adeia odigisis
Hungary	illesztőprogramok lic, jogosítvány, jogsi, licencszám, vezető engedély, vezetői engedély
Ireland	ceadúnas tiomána
Italy	patente di guida, patente di guida numero, patente guida, patente guida numero
Latvia	autovadītāja apliecība, licences numurs, vadītāja apliecība, vadītāja apliecības numurs, vadītāja atļauja, vadītāja licences numurs, vadītāji lic.
Lithuania	vairuotojo pažymėjimas

Negara atau wilayah	Kata kunci
Luxembourg	fahrerlaubnis, führerschäin
Malta	liċenzja tas-sewqan
Netherlands	permis de conduire, rijbewijs, rijbewijsnummer
Poland	numer licencyjny, prawo jazdy, zezwolenie na prowadzenie
Portugal	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
Romania	numărul permisului de conducere, permis de conducere
Slovakia	číslo licencie, číslo vodičského preukazu, ovládače lic., povolenia vodičov, povolenie jazdu, povolenie na jazdu, povolenie vodiča, vodičský preukaz
Slovenia	vozniško dovoljenje
Spain	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
Sweden	ajokortin numero, dlno# ajokortti, drivere lic., förare lic., körkort, körkort nummer, körkortsn ummer, kuljettajat lic.

Negara atau wilayah	Kata kunci
UK	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
US	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

Kata kunci untuk nomor induk kependudukan

Untuk mendeteksi berbagai jenis nomor identifikasi nasional, Amazon SNS membutuhkan kata kunci untuk berada di dekat angka-angka tersebut. Ini termasuk kode Documento Nacional de Identidad (DNI) (Spanyol), kode Institut Nasional Prancis untuk Statistik dan Studi Ekonomi (INSEE), nomor Kartu Identitas Nasional Jerman, dan nomor Registro Geral (RG) (Brasil).

Tabel berikut mencantumkan kata kunci yang SNS diakui Amazon untuk negara dan wilayah tertentu.

Negara atau wilayah	Kata kunci
Brazil	registro geral, rg
France	assurance sociale, carte nationale d’identit é, cni, code sécurité sociale, French social security number, fssn#, insee, insurance number, national id number, nationalid#,

Negara atau wilayah	Kata kunci
	numéro d'assurance, sécurité sociale, sécurité sociale non., sécurité sociale numéro, social, social security, social security number, socialsecuritynumber, ss#, ssn, ssn#
Germany	ausweisnummer, id number, identification number, identity number, insurance number, personal id, personalausweis
Italy	codice fiscal, dati anagrafici, ehic, health card, health insurance card, p. iva, partita i.v.a., personal data, tax code, tessera sanitaria
Spain	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#

Kata kunci untuk nomor paspor

Untuk mendeteksi berbagai jenis nomor paspor, Amazon SNS membutuhkan kata kunci untuk berada di dekat angka-angka tersebut. Tabel berikut mencantumkan kata kunci yang SNS diakui Amazon untuk negara dan wilayah tertentu.

Negara atau wilayah	Kata kunci
Canada	passeport, passeport#, passport, passport#, passportno, passportno#
France	numéro de passeport, passeport, passeport #, passeport #, passeportn °, passeport n °, passeportNon, passeport non

Negara atau wilayah	Kata kunci
Germany	ausstellungsdatum, ausstellungsort, geburtsda tum, passport, passports, reisepass, reisepass– nr, reisepassnummer
Italy	italian passport number, numéro passeport , numéro passeport italien, passaporto, passaporto italiana, passaporto numero, passport number, repubblica italiana passaport o
Spain	españa pasaporte, libreta pasaporte, número pasaporte, pasaporte, passport, passport book, passport no, passport number, spain passport
UK	passeport #, passeport n °, passeportNon, passeport non, passeportn °, passport #, passport no, passport number, passport#, passportid
US	passport, travel document

Kata kunci untuk nomor pokok wajib pajak

Untuk mendeteksi berbagai jenis identifikasi wajib pajak dan nomor referensi, Amazon SNS membutuhkan kata kunci untuk berada di dekat angka-angka tersebut. Tabel berikut mencantumkan kata kunci yang SNS diakui Amazon untuk negara dan wilayah tertentu.

Negara atau wilayah	Kata kunci
Brazil	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

Negara atau wilayah	Kata kunci
France	numéro d'identification fiscale, tax id, tax identification number, tax number, tin, tin#
Germany	identifikationsnummer, steuer id, steueride ntifikationsnummer, steuernummer, tax id, tax identification number, tax number
Spain	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#
UK	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
US	nomor identifikasi wajib pajak individu, itin, i.t.i.n.

Pengidentifikasi data ARNs untuk informasi yang dapat diidentifikasi secara pribadi () PII

Tabel berikut mencantumkan Nama Sumber Daya Amazon (ARNs) untuk pengidentifikasi data yang dapat ditambahkan ke kebijakan perlindungan data Anda.

PIIpengidentifikasi data ARNs
arn:aws:dataprotection: :aws:Pengenal data/alamat
arn:aws:dataprotection: :aws:data-identifier/ -BR CepCode
arn:aws:dataprotection: :aws:Pengenal data/CNPJ-BR

PIIpengidentifikasi data ARNs
arn:aws:dataprotection: :aws:data-identifier/ -BR CpfCode
arn:aws:dataprotection: :aws:data-identifier/ DateOfBirth
arn:aws:dataprotection: :aws:data-identifier/ -AT DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -AU DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -BE DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -BG DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -CA DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -CY DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -CZ DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -DE DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -DK DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -EE DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -ES DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -FI DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -FR DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -GB DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -GR DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -HR DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -HU DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -IE DriversLicense

PIIpengidentifikasi data ARNs
arn:aws:dataprotection: :aws:data-identifier/ -IT DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -LT DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -LU DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -LV DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -MT DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -NL DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -PL DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -PT DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -RO DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -SE DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -SI DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -SK DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -US DriversLicense
arn:aws:dataprotection: :aws:data-identifier/ -GB ElectoralRollNumber
arn:aws:dataprotection: :aws:data-identifier/ EmailAddress
arn:aws:dataprotection: :aws:data-identifier/ -US IndividualTaxIdentificationNumber
arn:aws:dataprotection: :aws:data-identifier/ -FR InseeCode
arn:aws:dataprotection: :aws:data-identifier/ LatLong
arn:aws:dataprotection: :aws:data-identifier/nama
arn:aws:dataprotection: :aws:data-identifier/ -DE NationalIdentificationNumber

PIIpengidentifikasi data ARNs
arn:aws:dataprotection: :aws:data-identifier/ -ES NationalIdentificationNumber
arn:aws:dataprotection: :aws:data-identifier/ -IT NationalIdentificationNumber
arn:aws:dataprotection: :aws:data-identifier/ -ES NieNumber
arn:aws:dataprotection: :aws:data-identifier/ -ES NifNumber
arn:aws:dataprotection: :aws:data-identifier/ -CA PassportNumber
arn:aws:dataprotection: :aws:data-identifier/ -DE PassportNumber
arn:aws:dataprotection: :aws:data-identifier/ -ES PassportNumber
arn:aws:dataprotection: :aws:data-identifier/ -FR PassportNumber
arn:aws:dataprotection: :aws:data-identifier/ -GB PassportNumber
arn:aws:dataprotection: :aws:data-identifier/ -IT PassportNumber
arn:aws:dataprotection: :aws:data-identifier/ -US PassportNumber
arn:aws:dataprotection: :aws:data-identifier/ -CA PermanentResidenceNumber
arn:aws:dataprotection: :aws:data-identifier/ -BR PhoneNumber
arn:aws:dataprotection: :aws:data-identifier/ -DE PhoneNumber
arn:aws:dataprotection: :aws:data-identifier/ -ES PhoneNumber
arn:aws:dataprotection: :aws:data-identifier/ -FR PhoneNumber
arn:aws:dataprotection: :aws:data-identifier/ -GB PhoneNumber
arn:aws:dataprotection: :aws:data-identifier/ -IT PhoneNumber
arn:aws:dataprotection: :aws:data-identifier/ -US PhoneNumber
arn:aws:dataprotection: :aws:data-identifier/ -CA PostalCode

PIIpengidentifikasi data ARNs
arn:aws:dataprotection: :aws:data-identifier/ -BR RgNumber
arn:aws:dataprotection: :aws:data-identifier/ -CA SocialInsuranceNumber
arn:aws:dataprotection: :aws:pengenal data/ssn-es
arn:aws:dataprotection: :aws:Pengenal data/ssn-US
arn:aws:dataprotection: :aws:data-identifier/ -DE TaxId
arn:aws:dataprotection: :aws:data-identifier/ -ES TaxId
arn:aws:dataprotection: :aws:data-identifier/ -FR TaxId
arn:aws:dataprotection: :aws:data-identifier/ -GB TaxId
arn:aws:dataprotection: :aws:data-identifier/ VehicleIdentificationNumber
arn:aws:dataprotection: :aws:data-identifier/ -US ZipCode

Menggunakan pengidentifikasi data khusus di Amazon SNS

Pengidentifikasi data kustom (CDIs) memungkinkan Anda menentukan ekspresi reguler kustom Anda sendiri yang dapat digunakan dalam kebijakan perlindungan data Anda. Dengan menggunakan pengidentifikasi data khusus, Anda dapat menargetkan kasus penggunaan informasi identitas pribadi (PII) khusus bisnis yang tidak dapat diberikan oleh pengidentifikasi data terkelola. Misalnya, Anda dapat menggunakan pengenal data khusus untuk mencari karyawan khusus perusahaan. IDs Pengidentifikasi data khusus dapat digunakan bersama dengan pengidentifikasi data terkelola.

Topik

- Apa itu pengidentifikasi data khusus?
- Menggunakan pengidentifikasi data khusus dalam kebijakan perlindungan data Anda
- Kendala pengenal data kustom

Apa itu pengidentifikasi data khusus?

Pengidentifikasi data kustom (CDIs) memungkinkan Anda menentukan ekspresi reguler kustom Anda sendiri yang dapat digunakan dalam kebijakan perlindungan data Anda. Dengan menggunakan pengidentifikasi data khusus, Anda dapat menargetkan kasus penggunaan informasi identitas pribadi (PII) khusus bisnis yang tidak dapat diberikan oleh pengidentifikasi data terkelola. Misalnya, Anda dapat menggunakan pengenal data khusus untuk mencari karyawan khusus perusahaan. IDs Pengidentifikasi data khusus dapat digunakan bersama dengan pengidentifikasi data terkelola.

Menggunakan pengidentifikasi data khusus dalam kebijakan perlindungan data Anda

Kebijakan perlindungan data berikut menginstruksikan SNS topik Amazon untuk mendeteksi muatan yang membawa karyawan khusus perusahaanIDs, lalu menutupinya IDs menggunakan simbol hash (#).

1. Buat `Configuration` blok dalam kebijakan perlindungan data Anda.
2. Masukkan a `Name` untuk pengenal data kustom Anda. Misalnya, **`EmployeeId`**.
3. Masukkan a `Regex` untuk pengenal data kustom Anda. Misalnya, **`EID-\d{9}-US`**.
4. Lihat pengenal data kustom berikut dalam pernyataan kebijakan.

```
{
 "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. (Opsional) Lanjutkan untuk menambahkan pengidentifikasi data kustom tambahan ke `Configuration` blok sesuai kebutuhan. Kebijakan perlindungan data saat ini mendukung maksimal 10 pengidentifikasi data kustom.

Kendala pengenal data kustom

Pengidentifikasi data SNS khusus Amazon memiliki batasan berikut:

- Maksimal 10 pengidentifikasi data kustom didukung untuk setiap kebijakan perlindungan data.
- Nama pengidentifikasi data kustom memiliki panjang maksimum 128 karakter. Karakter berikut didukung:
  - Alfanumerik: (A-za-Z0-9)
  - Simbol: ('_' | '-')
- RegEx memiliki panjang maksimum 200 karakter. Karakter berikut didukung:
  - Alfanumerik: (A-za-Z0-9)
  - Simbol: ('_' | '#' | '=' | '@' |'/' | ';' | ',' | '-' | '')
  - RegEx karakter yang dipesan: ('^' | '$' | '?' | '[' | ']' | '{' | '}' | '|'\ '|' * '|' + '|'. ')
- Pengidentifikasi data kustom tidak dapat berbagi nama yang sama dengan pengenal data terkelola.
- Pengidentifikasi data khusus harus ditentukan dalam setiap kebijakan perlindungan data untuk setiap SNS topik Amazon.

Pengiriman SNS pesan Amazon

Topik ini menjelaskan bagaimana Amazon SNS menangani pengiriman pesan di berbagai skenario. Anda akan belajar tentang pengiriman pesan mentah, di mana Amazon SNS mengirimkan pesan dalam format aslinya yang tidak dimodifikasi ke titik akhir. Anda juga akan menemukan cara mengirim pesan dari SNS topik Amazon ke SQS antrian Amazon yang berbeda Akun AWS, memberikan wawasan tentang pesan lintas akun.

Topik ini memberikan informasi tentang pengiriman SNS pesan Amazon ke SQS antrian Amazon atau fungsi Lambda yang Wilayah AWS berbeda, cara kerja pengiriman lintas wilayah, dan pertimbangan yang terlibat.

Selain itu, Anda akan mempelajari cara memantau dan menafsirkan status pengiriman pesan, yang memberikan informasi penting tentang apakah pesan berhasil dikirim atau mengalami masalah. Jika pengiriman pesan gagal, Anda akan memahami proses percobaan ulang pengiriman pesan, termasuk cara Amazon SNS secara otomatis mencoba mengirim ulang pesan untuk memastikan mereka mencapai tujuan yang diinginkan. Topik ini juga membahas penggunaan antrian surat mati untuk menangkap pesan yang tidak dapat dikirimkan setelah beberapa upaya, memungkinkan Anda untuk menganalisis dan memecahkan masalah kegagalan ini secara efektif.

Topik

Pengiriman pesan SNS mentah Amazon

Untuk menghindari Amazon Data FirehoseSQS, Amazon, dan titik akhir HTTP/S memproses JSON pemformatan pesan, Amazon SNS mengizinkan pengiriman pesan mentah:

- Saat Anda mengaktifkan pengiriman pesan mentah untuk Amazon Data Firehose atau SQS titik akhir Amazon, SNS metadata Amazon apa pun akan dilucuti dari pesan yang dipublikasikan dan pesan dikirim apa adanya.
- Saat Anda mengaktifkan pengiriman pesan mentah untuk titik akhir HTTP /S, HTTP header `x-amz-sns-rawdelivery` dengan nilai yang disetel ke `true` ditambahkan ke pesan, yang menunjukkan bahwa pesan telah diterbitkan tanpa JSON memformat.
- Saat Anda mengaktifkan pengiriman pesan mentah untuk titik akhir HTTP /S, isi pesan, IP klien, dan header yang diperlukan akan dikirimkan. Ketika Anda menentukan atribut pesan, itu tidak akan dikirim.
- Saat Anda mengaktifkan pengiriman pesan mentah untuk endpoint Firehose, isi pesan akan terkirim. Ketika Anda menentukan atribut pesan, itu tidak akan dikirim.

Untuk mengaktifkan pengiriman pesan mentah menggunakan AWS SDK, Anda harus menggunakan `SetSubscriptionAttribute` API tindakan dan menetapkan nilai `RawMessageDelivery` atribut ke`true`.

Mengaktifkan pengiriman pesan mentah menggunakan AWS Management Console

1. Masuk ke SNSkonsol Amazon.
2. Di panel navigasi, pilih Topik.
3. Pada halaman Topik, pilih topik yang berlangganan Firehose, SQS Amazon, HTTP atau /S endpoint.
4. Pada ***`MyTopic`*** halaman, di bagian Berlangganan, pilih langganan dan pilih Edit.
5. Pada Edit ***`EXAMPLE1-23bc-4567-d890-ef12g3hij456`***halaman, di bagian Detail, pilih Aktifkan pengiriman pesan mentah.
6. Pilih Simpan perubahan.

Contoh format pesan

Dalam contoh berikut, pesan yang sama dikirim ke SQS antrian Amazon yang sama dua kali. Satu-satunya perbedaan adalah pengiriman pesan mentah dinonaktifkan untuk pesan pertama, dan diaktifkan untuk pesan kedua.

- Pengiriman pesan mentah dinonaktifkan

```
{
  "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"
}
```

- Pengiriman pesan mentah diaktifkan

```
This is a test message.
```

Atribut pesan dan pengiriman pesan mentah untuk SQS langganan Amazon

Amazon SNS mendukung pengiriman atribut pesan, yang memungkinkan Anda menyediakan item metadata terstruktur, seperti stempel waktu, data geospasial, tanda tangan, dan pengidentifikasi, tentang pesan tersebut. Untuk SQS langganan Amazon dengan Pengiriman Pesan Mentah diaktifkan, maksimal 10 atribut pesan dapat dikirim. Untuk mengirim lebih dari 10 atribut pesan, Anda harus menonaktifkan Pengiriman Pesan Mentah. Namun, Amazon SNS membuang pesan dengan lebih dari 10 atribut pesan yang diarahkan ke SQS langganan Amazon dengan Pengiriman Pesan Mentah diaktifkan, memperlakukannya sebagai kesalahan sisi klien.

Mengirim SNS pesan Amazon ke SQS antrian Amazon di akun yang berbeda

Dokumen ini menjelaskan cara mempublikasikan pemberitahuan ke SNS topik Amazon dengan satu atau beberapa langganan ke SQS antrian Amazon di akun lain. Atur topik dan antrean dengan cara yang sama jika mereka berada di akun yang sama (lihat SNSPemberitahuan Fanout Amazon ke SQS antrian Amazon untuk pemrosesan asinkron). Perbedaan utamanya adalah cara Anda menangani konfirmasi berlangganan, dan itu tergantung pada cara Anda berlangganan antrean ke topik.

Ini adalah praktik terbaik untuk mengikuti langkah-langkah yang direferensikan di bagian Queue owner create subscription bila memungkinkan, karena konfirmasi otomatis ketika pemilik antrian membuat langganan.

> **Note**
>
> Jika SQS antrian Amazon memiliki volume pesan yang tinggi, sebaiknya pemilik antrian membuat langganan.

Topik

- Pemilik antrean membuat langganan
- Pengguna yang tidak memiliki antrian membuat langganan
- Bagaimana cara memaksa langganan untuk meminta otentikasi pada permintaan berhenti berlangganan?

Pemilik antrean membuat langganan

Akun yang membuat SQS antrian Amazon adalah pemilik antrian. Saat pemilik antrean membuat langganan, maka memerlukan konfirmasi. Antrean mulai menerima notifikasi dari topik segera setelah tindakan `Subscribe` selesai. Agar pemilik antrean berlangganan pada pemilik topik, pemilik topik harus memberikan izin akun pemilik antrean untuk memanggil tindakan `Subscribe` pada topik.

Langkah 1: Untuk menetapkan kebijakan topik menggunakan AWS Management Console

1. Masuk ke SNSkonsol Amazon.

2. Di panel navigasi, pilih Topics (Topik).
3. Pilih topik, kemudian pilih Edit (Edit).
4. Pada Edit ***MyTopic***halaman, perluas bagian Kebijakan akses.
5. Masukkan kebijakan berikut:

```
{
   "Statement": [
      {
         "Effect": "Allow",
         "Principal": {
            "AWS": "111122223333"
         },
         "Action": "sns:Subscribe",
         "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic"
      }
   ]
}
```

   Kebijakan ini memberikan `111122223333` izin akun untuk menelepon `sns:Subscribe` `MyTopic` di akun`123456789012`.

   Seorang pengguna dengan kredensi untuk akun `111122223333` dapat berlangganan. `MyTopic` Izin ini memungkinkan ID akun untuk mendelegasikan izin kepada IAM pengguna/peran mereka. Hanya akun root atau pengguna administrator yang diizinkan untuk meneleponsns:Subscribe. IAMPengguna/peran juga harus mengizinkan antrian mereka `sns:subscribe` untuk berlangganan.
6. Pilih Simpan perubahan.

   Seorang pengguna dengan kredensi untuk akun `111122223333` dapat berlangganan. MyTopic

Langkah 2: Untuk menambahkan langganan SQS antrian Amazon ke topik lain Akun AWS menggunakan AWS Management Console

Sebelum Anda mulai, pastikan Anda memiliki topik dan antrian Anda, dan bahwa Anda telah memberikan izin untuk topik untuk mengirim pesan ke antrian. ARNs

1. Masuk ke SQSkonsol Amazon.
2. Pada panel navigasi, pilih Antrian.

3. Dari daftar antrian, pilih antrian untuk berlangganan topik Amazon. SNS
4. Pilih Berlangganan ke SNS topik Amazon.
5. Dari Tentukan SNS topik Amazon yang tersedia untuk menu antrian ini, pilih SNStopik Amazon untuk antrian Anda.
6. Pilih Masukkan SNS topik Amazon ARN dan kemudian masukkan topik Nama Sumber Daya Amazon (ARN).
7. Pilih Simpan.

> **Note**
>
> - Untuk dapat berkomunikasi dengan layanan, antrian harus memiliki izin untuk Amazon. SNS
> - Karena Anda adalah pemilik antrean, Anda tidak perlu mengonfirmasi langganan.

Pengguna yang tidak memiliki antrian membuat langganan

Setiap pengguna yang membuat langganan tetapi bukan pemilik antrean harus mengonfirmasi langganan.

Saat Anda menggunakan `Subscribe` tindakan, Amazon SNS mengirimkan konfirmasi berlangganan ke antrian. Langganan ditampilkan di SNS konsol Amazon, dengan ID langganannya disetel ke Konfirmasi Tertunda.

Untuk mengonfirmasi langganan, pengguna dengan izin untuk membaca pesan dari antrian harus mengambil konfirmasi langgananURL, dan pemilik langganan harus mengonfirmasi langganan menggunakan konfirmasi langganan. URL Tidak ada notifikasi yang dipublikasikan ke topik yang dikirim ke antrean sampai langganan dikonfirmasi. Untuk mengonfirmasi langganan, Anda dapat menggunakan SQS konsol Amazon atau `ReceiveMessage` tindakan.

> **Note**
>
> Sebelum Anda berlangganan endpoint ke topik, pastikan antrean dapat menerima olahpesan dari topik dengan menetapkan izin `sqs:SendMessage` untuk antrean. Untuk informasi selengkapnya, lihat Langkah 2: Berikan izin ke SNS topik Amazon untuk mengirim pesan ke SQS antrian Amazon.

Langkah 1: Untuk menambahkan langganan SQS antrian Amazon ke topik lain Akun AWS menggunakan AWS Management Console

Sebelum Anda mulai, pastikan Anda memiliki topik dan antrian Anda, dan bahwa Anda telah memberikan izin untuk topik untuk mengirim pesan ke antrian. ARNs

1. Masuk ke SNSkonsol Amazon.
2. Di panel navigasi, pilih Subscriptions (Langganan).
3. Di halaman Subscriptions (Langganan), pilih Create subscription (Buat langganan).
4. Di halaman Create subscription (Buat langganan), di bagian Details (Detail), lakukan:
   a. Untuk Topik ARN, ARN masukkan topik.
   b. Untuk Protokol, pilih Amazon SQS.
   c. Untuk Endpoint, masukkan ARN antrian.
   d. Pilih Buat langganan.

> **Note**
>
> - Untuk dapat berkomunikasi dengan layanan, antrian harus memiliki izin untuk Amazon. SNS

Berikut ini adalah contoh pernyataan kebijakan yang memungkinkan SNS topik Amazon mengirim pesan ke SQS antrian Amazon.

```
{
   "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"
      }
   }
}
```

Langkah 2: Untuk mengonfirmasi langganan menggunakan AWS Management Console

1. Masuk ke SQSkonsol Amazon.
2. Pilih antrean yang memiliki langganan tertunda untuk topik.
3. Pilih Kirim dan terima pesan, lalu pilih Poll untuk pesan.

   Olahpesan dengan konfirmasi berlangganan diterima dalam antrean.
4. Di kolom Body (Isi), lakukan:

   a. Pilih More Details (Detail selengkapnya).

   b. Dalam kotak dialog Detail Pesan, temukan dan catat URL nilai Berlangganan. Ini adalah tautan langganan Anda (contoh di bawah). Untuk detail tambahan tentang validasi API token, lihat `ConfirmSubscription`di SNS API Referensi Amazon.

   ```
   https://sns.us-west-2.amazonaws.com/?
   Action=ConfirmSubscription&TopicArn=arn:aws:sns:us-
   east-2:123456789012:MyTopic&Token=2336412f37fb...
   ```

   c. Catat tautan konfirmasi berlangganan. URLHarus diteruskan dari pemilik antrian ke pemilik langganan. Pemilik langganan harus memasukkan URL ke dalam SNSkonsol Amazon.
5. Masuk sebagai pemilik langganan ke SNSkonsol Amazon Pemilik langganan melakukan konfirmasi.
6. Pilih topik yang relevan.
7. Pilih langganan yang relevan di tabel daftar langganan topik. Ini diberi label sebagai “Konfirmasi tertunda”.
8. Pilih Konfirmasi langganan.
9. Modal muncul yang meminta tautan konfirmasi berlangganan. Rekatkan tautan konfirmasi berlangganan.
10. Pilih Konfirmasi langganan di modal.

    XMLRespons ditampilkan, misalnya:

    ```
    <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>
```

Antrean berlangganan siap menerima olahpesan dari topik.

11. (Opsional) Jika Anda melihat langganan topik di SNS konsol Amazon, Anda dapat melihat bahwa pesan Konfirmasi Tertunda telah diganti dengan langganan ARN di kolom ID Langganan.

Bagaimana cara memaksa langganan untuk meminta otentikasi pada permintaan berhenti berlangganan?

Pemilik langganan harus menyetel `AuthenticateOnUnsubscribe` bendera ke true pada konfirmasi langganan.

- `AuthenticateOnUnsubscribe`secara otomatis diatur ke true ketika pemilik antrian membuat langganan.
- `AuthenticateOnUnsubscribe`tidak dapat disetel ke true saat tautan konfirmasi langganan dinavigasi tanpa autentikasi.

Mengirim SNS pesan Amazon ke SQS antrian atau AWS Lambda fungsi Amazon di Wilayah yang berbeda

Amazon SNS mendukung pengiriman lintas wilayah, baik untuk Wilayah yang diaktifkan secara default maupun untuk Wilayah keikutsertaan. Untuk daftar Wilayah saat ini yang SNS didukung Amazon, termasuk AWS Wilayah keikutsertaan, lihat titik akhir dan kuota Layanan Pemberitahuan Sederhana Amazon di.Referensi Umum Amazon Web Services

Amazon SNS mendukung pengiriman notifikasi lintas wilayah ke SQS antrian Amazon dan ke fungsi. AWS Lambda Jika salah satu Wilayah adalah Wilayah keikutsertaan, Anda harus menentukan prinsip SNS layanan Amazon yang berbeda dalam kebijakan sumber daya berlangganan.

Perintah SNS berlangganan Amazon harus dijalankan di wilayah tempat Amazon SNS di-host, di wilayah yang sesuai. Misalnya, jika Amazon SNS berada di akun “A” di wilayah us-east-1, dan fungsi Lambda ada di akun “B” di wilayah us-east-2, CLI perintah berlangganan harus dijalankan di akun “A” di wilayah us-east-1.

Wilayah Keikutsertaan

Amazon SNS mendukung Wilayah keikutsertaan berikut:

Nama Wilayah	Wilayah
Wilayah Afrika (Cape Town)	af-south-1
Wilayah Asia Pasifik (Hong Kong)	ap-east-1
Wilayah Asia Pasifik (Hyderabad)	ap-south-2
Wilayah Asia Pasifik (Jakarta)	ap-southeast-3
Wilayah Asia Pasifik (Melbourne)	ap-southeast-4
Wilayah Asia Pasifik (Osaka)	ap-northeast-3
Wilayah Eropa (Milan)	eu-south-1
Wilayah Eropa (Spanyol)	eu-south-2
Wilayah Eropa (Zürich)	eu-central-2
Wilayah Israel (Tel Aviv)	il-central-1
Wilayah Timur Tengah (Bahrain)	me-south-1
Timur Tengah (UAE) Wilayah	me-central-1

Untuk informasi tentang mengaktifkan Wilayah keikutsertaan, lihat Mengelola AWS Wilayah di.Referensi Umum Amazon Web Services

Saat Anda menggunakan Amazon SNS untuk mengirimkan pesan dari Wilayah keikutsertaan ke Wilayah yang diaktifkan secara default, Anda harus mengubah kebijakan sumber daya yang dibuat untuk antrean. Mengganti utama `sns.amazonaws.com` dengan `sns.<opt-in-region>.amazonaws.com`. Sebagai contoh:

- Untuk berlangganan SQS antrian Amazon di US East (Virginia N.) ke SNS topik Amazon di Asia Pasifik (Hong Kong), ubah prinsipal dalam kebijakan antrian menjadi. `sns.ap-`

`east-1.amazonaws.com` Wilayah opt-in mencakup setiap wilayah yang diluncurkan setelah 20 Maret 2019, yang meliputi Asia Pasifik (Hong Kong), Asia Pasifik (Jakarta), Timur Tengah (Bahrain), Eropa (Milan), dan Afrika (Cape Town). Wilayah yang diluncurkan sebelum 20 Maret 2019 diaktifkan secara default.

Dukungan pengiriman lintas wilayah ke Amazon SQS

Jenis pengiriman lintas wilayah	Didukung/Tidak didukung	
Wilayah berkemampuan default untuk ikut serta Wilayah	Didukung menggunakan `sns.<opt-in-region>.amazonaws.com` dalam prinsip layanan untuk antrian	
Keikutsertaan Wilayah ke Wilayah yang diaktifkan default	Didukung menggunakan `sns.<opt-in-region>.amazonaws.com` dalam prinsip layanan untuk antrian	
Keikutsertaan Wilayah untuk ikut serta Wilayah	Tidak didukung	

Berikut ini adalah contoh pernyataan kebijakan akses yang memungkinkan SNS topik Amazon di Wilayah keikutsertaan (af-south-1) untuk dikirimkan ke antrian SQS Amazon di Wilayah (us-east-1). enabled-by-default Ini berisi konfigurasi utama layanan regional yang diperlukan di bawah jalur`Statement`//`Principal`. `Service`

```
{
  "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"
        }
      }
    },
    ...
  ]
}
```

- Untuk berlangganan AWS Lambda fungsi di AS Timur (Virginia Utara) ke SNS topik Amazon di Asia Pasifik (Hong Kong), ubah prinsip kebijakan AWS Lambda fungsi menjadi`sns.ap-east-1.amazonaws.com`. Wilayah opt-in mencakup setiap wilayah yang diluncurkan setelah 20 Maret 2019, yang meliputi Asia Pasifik (Hong Kong), Asia Pasifik (Jakarta), Timur Tengah (Bahrain), Eropa (Milan), dan Afrika (Cape Town). Wilayah yang diluncurkan sebelum 20 Maret 2019 diaktifkan secara default.

  Dukungan pengiriman lintas wilayah ke AWS Lambda

Jenis pengiriman lintas wilayah	Didukung/Tidak didukung	
Wilayah berkemampuan default untuk ikut serta Wilayah	Tidak didukung	
Keikutsertaan Wilayah ke Wilayah yang diaktifkan default	Didukung menggunakan `sns.<opt-in-region>.amazonaws.com` dalam prinsip layanan untuk fungsi Lambda	
Keikutsertaan Wilayah untuk ikut serta Wilayah	Tidak didukung	

Status pengiriman SNS pesan Amazon

Amazon SNS menyediakan dukungan untuk mencatat status pengiriman pesan notifikasi yang dikirim ke topik dengan SNS titik akhir Amazon berikut:

- HTTP
- Amazon Data Firehose
- AWS Lambda
- Titik akhir aplikasi platform
- Amazon Simple Queue Service

Setelah Anda mengonfigurasi atribut status pengiriman pesan, entri log dikirim ke CloudWatch Log untuk pesan yang dikirim ke pelanggan topik. Mencatat status pengiriman pesan membantu memberikan wawasan operasional yang lebih baik, seperti berikut ini:

- Mengetahui apakah pesan dikirim ke SNS titik akhir Amazon.
- Mengidentifikasi respons yang dikirim dari SNS titik akhir Amazon ke AmazonSNS.
- Menentukan waktu tinggal pesan (waktu antara stempel waktu publikasi dan sebelum menyerahkan ke titik akhir Amazon). SNS

Untuk mengonfigurasi atribut topik untuk status pengiriman pesan, Anda dapat menggunakan AWS Management Console, kit pengembangan AWS perangkat lunak (SDKs), kueriAPI, atau AWS CloudFormation.

Topik

Mengkonfigurasi pencatatan status pengiriman menggunakan AWS Management Console

1. Masuk ke SNSkonsol Amazon.

2. Di panel navigasi, pilih Topics (Topik).
3. Di halaman Topics (Topik), pilih topik dan kemudian pilih Edit (Edit).
4. Pada Edit ***MyTopic***halaman, perluas bagian Pencatatan status pengiriman.
5. Pilih protokol yang ingin Anda catat status pengiriman; misalnya AWS Lambda.
6. Masukkan sample rate Sukses (persentase pesan sukses yang ingin Anda terima CloudWatch Log).
7. Di bagian IAMperan, lakukan salah satu hal berikut:

   - Untuk memilih peran layanan yang ada dari akun Anda, pilih Gunakan peran layanan yang ada, lalu tentukan IAM peran untuk pengiriman yang berhasil dan gagal.

   - Untuk membuat peran layanan baru di akun Anda, pilih Buat peran layanan baru, pilih Buat peran baru untuk menentukan IAM peran untuk pengiriman yang berhasil dan gagal di IAM konsol.

     Untuk memberi Amazon akses SNS tulis untuk menggunakan CloudWatch Log atas nama Anda, pilih Izinkan.
8. Pilih Simpan perubahan.

   Sekarang Anda dapat melihat dan mengurai CloudWatch Log yang berisi status pengiriman pesan. Untuk informasi selengkapnya tentang penggunaan CloudWatch, lihat CloudWatchDokumentasi.

Mengkonfigurasi pencatatan status pengiriman menggunakan AWS SDKs

AWS SDKsMenyediakan APIs dalam beberapa bahasa untuk menggunakan atribut status pengiriman pesan dengan AmazonSNS.

Atribut topik

Anda dapat menggunakan nilai nama atribut topik berikut untuk status pengiriman pesan:

HTTP

- `HTTPSuccessFeedbackRoleArn`— Menunjukkan status pengiriman pesan yang berhasil untuk SNS topik Amazon yang berlangganan titik HTTP akhir.

- `HTTPSuccessFeedbackSampleRate`— Menunjukkan persentase pesan yang berhasil untuk sampel untuk SNS topik Amazon yang berlangganan titik HTTP akhir.
- `HTTPFailureFeedbackRoleArn`— Menunjukkan status pengiriman pesan gagal untuk SNS topik Amazon yang berlangganan titik HTTP akhir.

Amazon Data Firehose

- `FirehoseSuccessFeedbackRoleArn`— Menunjukkan status pengiriman pesan yang berhasil untuk SNS topik Amazon yang berlangganan titik akhir Amazon Kinesis Data Firehose.
- `FirehoseSuccessFeedbackSampleRate`— Menunjukkan persentase pesan yang berhasil untuk diambil sampel untuk SNS topik Amazon yang berlangganan titik akhir Amazon Kinesis Data Firehose.
- `FirehoseFailureFeedbackRoleArn`— Menunjukkan status pengiriman pesan yang gagal untuk SNS topik Amazon yang berlangganan titik akhir Amazon Kinesis Data Firehose.

AWS Lambda

- `LambdaSuccessFeedbackRoleArn`— Menunjukkan status pengiriman pesan yang berhasil untuk SNS topik Amazon yang berlangganan titik akhir Lambda.
- `LambdaSuccessFeedbackSampleRate`— Menunjukkan persentase pesan yang berhasil untuk sampel untuk SNS topik Amazon yang berlangganan titik akhir Lambda.
- `LambdaFailureFeedbackRoleArn`— Menunjukkan status pengiriman pesan gagal untuk SNS topik Amazon yang berlangganan titik akhir Lambda.

Titik akhir aplikasi platform

- `ApplicationSuccessFeedbackRoleArn`— Menunjukkan status pengiriman pesan yang berhasil untuk SNS topik Amazon yang berlangganan titik akhir AWS aplikasi.
- `ApplicationSuccessFeedbackSampleRate`— Menunjukkan persentase pesan yang berhasil untuk sampel untuk SNS topik Amazon yang berlangganan titik akhir AWS aplikasi.
- `ApplicationFailureFeedbackRoleArn`— Menunjukkan status pengiriman pesan gagal untuk SNS topik Amazon yang berlangganan titik akhir AWS aplikasi.

> **Note**
>
> Selain dapat mengonfigurasi atribut topik untuk status pengiriman pesan pesan notifikasi yang dikirim ke titik akhir SNS aplikasi Amazon, Anda juga dapat mengonfigurasi atribut aplikasi untuk status pengiriman pesan pemberitahuan push yang dikirim ke layanan pemberitahuan push. Untuk informasi selengkapnya, lihat Menggunakan Atribut SNS Aplikasi Amazon untuk Status Pengiriman Pesan.

Amazon SQS

- `SQSSuccessFeedbackRoleArn`— Menunjukkan status pengiriman pesan yang berhasil untuk SNS topik Amazon yang berlangganan titik SQS akhir Amazon.
- `SQSSuccessFeedbackSampleRate`— Menunjukkan persentase pesan yang berhasil untuk sampel untuk SNS topik Amazon yang berlangganan titik SQS akhir Amazon.
- `SQSFailureFeedbackRoleArn`— Menunjukkan status pengiriman pesan yang gagal untuk SNS topik Amazon yang berlangganan titik SQS akhir Amazon.

> **Note**
>
> `<ENDPOINT>FailureFeedbackRoleArn`Atribut `<ENDPOINT>SuccessFeedbackRoleArn` dan digunakan untuk memberi Amazon akses SNS tulis untuk menggunakan CloudWatch Log atas nama Anda. Atribut `<ENDPOINT>SuccessFeedbackSampleRate` adalah untuk menentukan persentase tingkat sampel (0-100) dari pesan yang berhasil terkirim. Setelah Anda mengonfigurasi `<ENDPOINT>FailureFeedbackRoleArn` atribut, maka semua pengiriman pesan yang gagal menghasilkan CloudWatch Log.

AWS SDKcontoh untuk mengkonfigurasi atribut topik

Contoh kode berikut menunjukkan cara menggunakan`SetTopicAttributes`.

CLI

AWS CLI

Untuk menetapkan atribut untuk topik

`set-topic-attributes`Contoh berikut menetapkan `DisplayName` atribut untuk topik yang ditentukan.

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name DisplayName \
    --attribute-value MyTopicDisplayName
```

Perintah ini tidak menghasilkan output.

- Untuk API detailnya, lihat SetTopicAttributesdi Referensi AWS CLI Perintah.

Java

SDKuntuk Java 2.x

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

```
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);
        }
    }
}
```

- Untuk API detailnya, lihat SetTopicAttributesdi AWS SDK for Java 2.x APIReferensi.

JavaScript

SDKuntuk JavaScript (v3)

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

Buat klien dalam modul terpisah dan ekspor klien tersebut.

```
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({});
```

Impor modul SDK dan klien dan panggil fileAPI.

```
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;
};
```

- Untuk informasi selengkapnya, lihat AWS SDK for JavaScript Panduan Developer.
- Untuk API detailnya, lihat SetTopicAttributesdi AWS SDK for JavaScript APIReferensi.

Kotlin

SDKuntuk Kotlin

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

```
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.")
    }
}
```

- Untuk API detailnya, lihat SetTopicAttributes AWSSDKAPIreferensi Kotlin.

PHP

SDKuntuk PHP

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

```
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());
}
```

- Untuk API detailnya, lihat SetTopicAttributesdi AWS SDK for PHP APIReferensi.

Ruby

SDKuntuk Ruby

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

```
# 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
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for Ruby.
- Untuk API detailnya, lihat SetTopicAttributesdi AWS SDK for Ruby APIReferensi.

SAP ABAP

SDKuntuk SAP ABAP

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

```
    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.
```

- Untuk API detailnya, lihat SetTopicAttributes AWSSDKuntuk SAP ABAP API referensi.

Mengkonfigurasi pencatatan status pengiriman menggunakan AWS CloudFormation

Untuk mengonfigurasi `DeliveryStatusLogging` penggunaan AWS CloudFormation, gunakan YAML templat JSON atau untuk membuat AWS CloudFormation tumpukan. Untuk informasi selengkapnya, lihat `DeliveryStatusLogging` properti `AWS::SNS::Topic` sumber daya di Panduan AWS CloudFormation Pengguna. Di bawah ini adalah contoh AWS CloudFormation template dalam JSON dan YAML untuk membuat topik baru atau memperbarui topik yang ada dengan semua `DeliveryStatusLogging` atribut untuk SQS protokol Amazon.

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
```

Mencoba lagi pengiriman SNS pesan Amazon

Amazon SNS mendefinisikan kebijakan pengiriman untuk setiap protokol pengiriman. Kebijakan pengiriman menentukan cara Amazon SNS mencoba ulang pengiriman pesan saat terjadi kesalahan sisi server (saat sistem yang meng-host titik akhir berlangganan menjadi tidak tersedia). Ketika kebijakan pengiriman habis, Amazon SNS berhenti mencoba kembali pengiriman dan membuang pesan—kecuali antrian surat mati dilampirkan ke langganan. Untuk informasi selengkapnya, lihat Antrian SNS surat mati Amazon.

Topik

- Protokol dan kebijakan pengiriman
- Tahap kebijakan pengiriman
- Membuat kebijakan pengiriman HTTP /S

Protokol dan kebijakan pengiriman

> **Note**
>
> - Dengan pengecualian HTTP/S, you can't change Amazon SNS-defined delivery policies. Only HTTP/S mendukung kebijakan khusus. Lihat Membuat kebijakan pengiriman HTTP / S.
> - Amazon SNS menerapkan jittering pada percobaan ulang pengiriman. Untuk informasi selengkapnya, lihat posting Exponential Backoff and Jitter (Backoff Eksponensial dan Jitter) di AWS Architecture Blog (Blog Arsitektur).
> - Total waktu coba ulang kebijakan untuk titik akhir HTTP /S tidak boleh lebih dari 3.600 detik. Ini adalah batas yang sulit dan tidak dapat ditingkatkan.

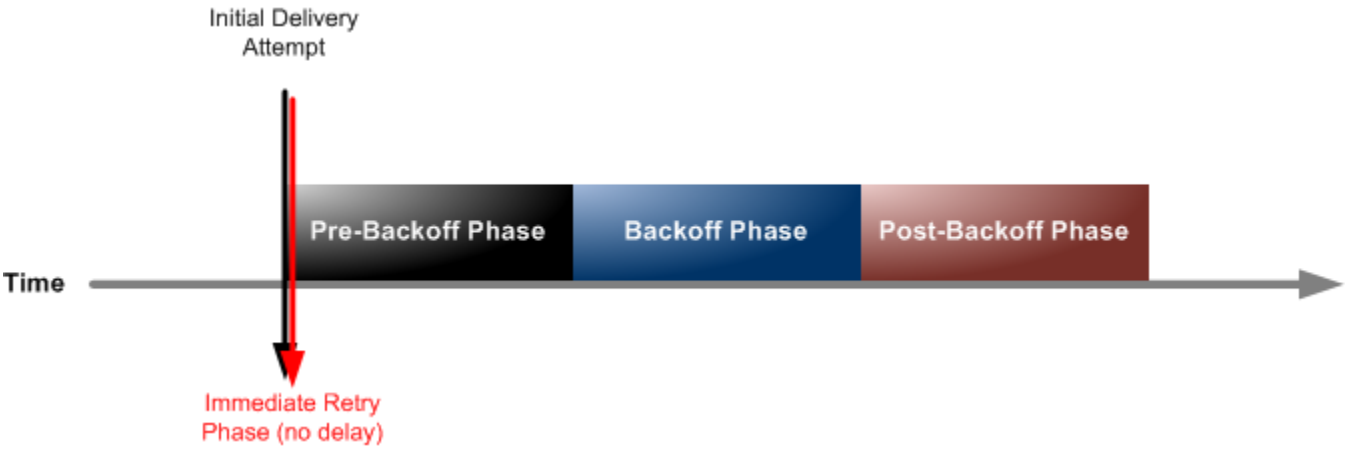
Jenis endpoint	Protokol pengiriman	Fase coba ulang segera (tanpa penundaan)	Fase pra-backoff	Fase backoff	Fase pasca-backoff	Jumlah percobaan
AWS titik akhir terkelola	Hos Pemadam Kebakaran Data Amazon[1]	3 kali, tanpa penundaan	2 kali, 1 detik terpisah	10 kali, dengan backoff eksponensial, dari 1 detik sampai 20 detik	100.000 kali, terpisah 20 detik	100,015 kali, selama 23 hari
	AWS Lambda					
	Amazon SQS					
Endpoint yang	SMTP	0 kali, tanpa penundaan	2 kali, 10 detik terpisah	10 kali, dengan backoff	38 kali, 600 detik	50 percobaan
	SMS					

Jenis endpoint	Protokol pengiriman	Fase coba ulang segera (tanpa penundaan)	Fase pra-backoff	Fase backoff	Fase pasca-backoff	Jumlah percobaan
dikelola pelanggan	Push seluler			eksponensial, dari 10 detik hingga 600 detik (10 menit)	(10 menit) terpisah	, lebih dari 6 jam

[1] Untuk kesalahan pembatasan dengan protokol Firehose, Amazon SNS menggunakan kebijakan pengiriman yang sama seperti untuk titik akhir yang dikelola pelanggan.

Tahap kebijakan pengiriman

Diagram berikut menunjukkan fase kebijakan pengiriman.



Setiap kebijakan pengiriman terdiri dari empat fase.

1. Immediate Retry Phase (No Delay) (Fase Coba Ulang Segera) (Tanpa Penundaan) – Fase ini terjadi segera setelah upaya pengiriman awal. Tidak ada penundaan antara pengiriman ulang pesan dalam fase ini.

2. Pre-Backoff Phase (Fase Pra-Backoff) – Fase ini mengikuti Immediate Retry Phase (Fase Coba Ulang Segera). Amazon SNS menggunakan fase ini untuk mencoba serangkaian percobaan ulang sebelum menerapkan fungsi backoff. Fase ini menentukan jumlah pengiriman ulang pesan dan jumlah penundaan antara mereka.
3. Backoff Phase (Fase Backoff) - Fase ini mengontrol penundaan antara pengiriman ulang pesan dengan menggunakan fungsi pengiriman retry-backoff. Fase ini menetapkan penundaan minimum, penundaan maksimum, dan fungsi retry-backoff yang menentukan seberapa cepat penundaan meningkat dari penundaan minimum ke maksimum. Fungsi backoff berupa aritmatika, eksponensial, geometris, atau linier.
4. Post-Backoff Phase (Fase Pasca Backoff) - Fase ini mengikuti fase backoff. Fase ini menentukan sejumlah pengiriman ulang pesan dan jumlah penundaan di antara mereka. Ini adalah fase terakhir.

Membuat kebijakan pengiriman HTTP /S

Anda dapat menggunakan kebijakan pengiriman dan empat tahapannya untuk menentukan cara Amazon SNS mencoba ulang pengiriman pesan ke titik akhir HTTP /S. Amazon SNS memungkinkan Anda mengganti kebijakan coba ulang default untuk HTTP titik akhir ketika Anda mungkin, misalnya, ingin menyesuaikan kebijakan berdasarkan kapasitas HTTP server Anda.

Anda dapat mengatur HTTP/S delivery policy as a JSON object at the subscription or topic level. When you define the policy at the topic level, it applies to all HTTP/S langganan yang terkait dengan topik tersebut. Untuk menetapkan kebijakan pengiriman di tingkat langganan, Anda dapat menggunakan `SetSubscriptionAttributes`APItindakan `Subscribe`atau tindakan. Untuk menetapkan kebijakan pengiriman di tingkat topik, Anda dapat menggunakan `SetTopicAttributes`APItindakan `CreateTopic`atau tindakan. Atau, Anda juga dapat menggunakan sumber daya AWS::SNS: :Subscription di AWS CloudFormation template Anda.

Anda harus menyesuaikan kebijakan pengiriman sesuai dengan HTTP/S server's capacity. You can set the policy as a topic attribute or a subscription attribute. If all HTTP/S subscriptions in your topic target the same HTTP/S server, we recommend that you set the delivery policy as a topic attribute, so that it remains valid for all HTTP/S subscriptions in the topic. Otherwise, you must compose a delivery policy for each HTTP/S subscription in your topic, according the capacity of the HTTP/S server yang ditargetkan oleh kebijakan tersebut.

Anda juga dapat mengatur header Content-Type dalam kebijakan permintaan untuk menentukan jenis media notifikasi. Secara default, Amazon SNS mengirimkan semua notifikasi ke titik akhir

HTTP /S dengan jenis konten yang disetel `ketext/plain; charset=UTF-8`. Amazon SNS memungkinkan Anda mengganti kebijakan permintaan default. Lihat tabel di bawah ini untuk dukungan `headerContentType`dan pengekangan.

JSONObjek berikut menunjukkan kebijakan pengiriman yang menginstruksikan Amazon SNS untuk mencoba kembali upaya pengiriman HTTP /S yang gagal, sebagai berikut:

1. 3 kali segera dalam fase tanpa penundaan
2. 2 kali (1 detik terpisah) dalam fase pra-backoff
3. 10 kali (dengan backoff eksponensial dari 1 detik hingga 60 detik)
4. 35 kali (60 detik terpisah) dalam fase pasca-backoff

Dalam kebijakan pengiriman sampel ini, Amazon SNS melakukan total 50 upaya sebelum membuang pesan. Untuk menyimpan pesan setelah percobaan ulang yang ditentukan dalam kebijakan pengiriman habis, konfigurasikan langganan Anda untuk memindahkan pesan yang tidak terkirim ke antrean huruf mati (). DLQ Untuk informasi selengkapnya, lihat Antrian SNS surat mati Amazon.

> **Note**
>
> Kebijakan pengiriman ini juga menginstruksikan Amazon SNS untuk membatasi pengiriman hingga tidak lebih dari 10 per detik, menggunakan properti. `maxReceivesPerSecond` Tingkat self-throttling ini dapat menghasilkan lebih banyak pesan yang dipublikasikan (lalu lintas masuk) daripada yang dikirim (lalu lintas keluar). Jika lalu lintas masuk lebih banyak daripada lalu lintas keluar, langganan Anda dapat mengakumulasi simpanan pesan yang besar, yang dapat menyebabkan latensi pengiriman pesan menjadi tinggi. Dalam kebijakan pengiriman Anda, pastikan untuk menentukan nilai untuk `maxReceivesPerSecond` yang tidak berdampak buruk pada beban kerja Anda.

> **Note**
>
> Kebijakan pengiriman ini mengesampingkan tipe konten default untuk notifikasi HTTP /S. `application/json`

```
{
```

```
    "healthyRetryPolicy": {
        "minDelayTarget": 1,
        "maxDelayTarget": 60,
        "numRetries": 50,
        "numNoDelayRetries": 3,
        "numMinDelayRetries": 2,
        "numMaxDelayRetries": 35,
        "backoffFunction": "exponential"
    },
    "throttlePolicy": {
        "maxReceivesPerSecond": 10
    },
    "requestPolicy": {
        "headerContentType": "application/json"
    }
}
```

Kebijakan pengiriman terdiri dari kebijakan coba lagi, kebijakan throttle, dan kebijakan permintaan. Secara total, ada 9 atribut dalam kebijakan pengiriman.

Kebijakan	Deskripsi	Kendala
`minDelayTarget`	Penundaan minimum untuk pengiriman ulang. Unit: Detik	1 hingga penundaan maksimum Default: 20
`maxDelayTarget`	Penundaan maksimum untuk pengiriman ulang. Unit: Detik	Minimal delay ke 3.600 Default: 20
`numRetries`	Jumlah total pengiriman ulang, termasuk pengiriman ulang langsung, pra-backoff, backoff, dan pasca-backoff.	0 hingga 100 Default: 3
`numNoDelayRetries`	Jumlah pengiriman ulang yang harus dilakukan segera, tanpa penundaan di antara mereka.	0 atau lebih Default: 0

Kebijakan	Deskripsi	Kendala
`numMinDelayRetries`	Jumlah pengiriman ulang dalam fase pra-backoff, dengan penundaan minimum yang ditentukan di antara keduanya.	0 atau lebih Default: 0
`numMaxDelayRetries`	Jumlah pengiriman ulang dalam fase pasca-bac koff, dengan penundaan maksimum di antara keduanya.	0 atau lebih Default: 0
`backoffFunction`	Model untuk backoff di antara pengiriman ulang.	Salah satu dari empat pilihan: • aritmatika • eksponensial • geometris • linier Default: linier
`maxReceivesPerSecond`	Jumlah maksimum pengiriman per detik, per langganan.	1 atau lebih Default: Tidak ada throttling

Kebijakan	Deskripsi	Kendala
`headerContentType`	Jenis konten notifikasi yang dikirim ke titik akhir HTTP /S.	Jika kebijakan permintaan tidak ditentukan, jenis konten akan menjadi default. `text/plain; charset=UTF-8` Saat pengiriman pesan mentah dinonaktifkan untuk langganan (default), atau saat kebijakan pengirima n ditentukan pada tingkat topik, jenis konten header yang didukung adalah dan. `application/json text/plain` Saat pengiriman pesan mentah diaktifkan untuk langganan, jenis konten berikut didukung: • teks/css • teks/csv • teks/html • teks/polos • teks/xml • aplikasi/atom+xml+ • aplikasi/json • aplikasi/oktet-aliran • aplikasi/sabun+xml+ • aplikasi/ x-www-form-urlencoded • aplikasi/xhtml+xml+ • aplikasi/xml

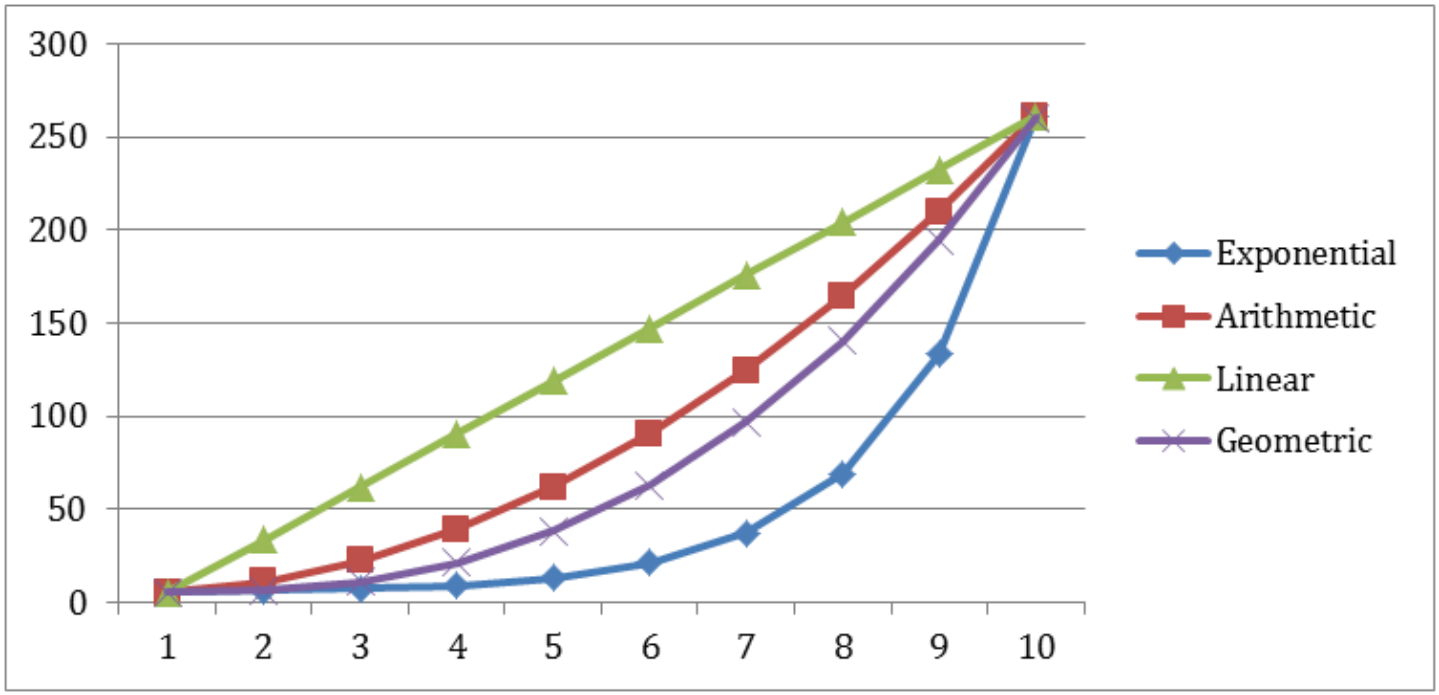
Amazon SNS menggunakan rumus berikut untuk menghitung jumlah percobaan ulang dalam fase backoff:

```
numRetries - numNoDelayRetries - numMinDelayRetries - numMaxDelayRetries
```

Anda dapat menggunakan tiga parameter untuk mengontrol frekuensi pengiriman ulang dalam fase backoff.

- `minDelayTarget` — Menetapkan penundaan terkait dengan pengiriman ulang dalam fase backoff.
- `maxDelayTarget` — Menetapkan penundaan terkait dengan pengiriman ulang terakhir dalam fase backoff.
- `backoffFunction`— Mendefinisikan algoritme yang SNS digunakan Amazon untuk menghitung penundaan yang terkait dengan semua upaya coba lagi antara percobaan ulang pertama dan terakhir di fase backoff. Anda dapat menggunakan salah satu dari empat fungsi retry-backoff.

Diagram berikut menunjukkan bagaimana setiap fungsi backoff pengiriman ulang mempengaruhi penundaan terkait dengan pengiriman ulang selama fase backoff: Kebijakan pengiriman dengan jumlah total pengiriman ulang diatur ke 10, penundaan minimum diatur ke 5 detik, dan penundaan maksimum diatur ke 260 detik. Sumbu vertikal mewakili penundaan dalam detik yang terkait dengan masing-masing 10 kali pengiriman ulang. Sumbu horizontal mewakili jumlah pengiriman ulang, dari pengiriman ulang pertama hingga kesepuluh.



Antrian SNS surat mati Amazon

Antrian surat mati adalah SQS antrian Amazon yang dapat ditargetkan oleh SNS langganan Amazon untuk pesan yang tidak dapat dikirim ke pelanggan dengan sukses. Pesan yang tidak dapat dikirim karena kesalahan klien atau kesalahan server disimpan dalam antrean surat mati untuk analisis lebih lanjut atau pemrosesan ulang. Untuk informasi selengkapnya, silakan lihat Mengonfigurasi antrian SNS surat mati Amazon untuk berlangganan dan Mencoba lagi pengiriman SNS pesan Amazon.

Note

- SNSLangganan Amazon dan SQS antrian Amazon harus berada di bawah AWS akun dan Wilayah yang sama.
- Untuk FIFOtopik, Anda dapat menggunakan SQS antrian Amazon sebagai antrian surat mati untuk langganan Amazon. SNS FIFOlangganan topik menggunakan FIFO antrian, dan langganan topik standar menggunakan antrian standar.
- Untuk menggunakan antrean Amazon terenkripsi sebagai SQS antrian huruf mati, Anda harus menggunakan kustom KMS dengan kebijakan kunci yang memberikan akses utama layanan Amazon ke tindakan. SNS AWS KMS API Untuk informasi selengkapnya, lihat Mengamankan SNS data Amazon dengan enkripsi sisi server di panduan ini dan

Melindungi SQS Data Amazon Menggunakan Enkripsi Sisi Server (SSE) dan AWS KMS di Panduan Pengembang Layanan Antrian Sederhana Amazon.

Topik

Mengapa pengiriman pesan gagal?

Secara umum, pengiriman pesan gagal saat Amazon tidak SNS dapat mengakses titik akhir berlangganan karena kesalahan sisi klien atau sisi server. Saat Amazon SNS menerima kesalahan sisi klien, atau terus menerima kesalahan sisi server untuk pesan di luar jumlah percobaan ulang yang ditentukan oleh kebijakan percobaan ulang yang sesuai, Amazon akan SNS membuang pesan tersebut—kecuali antrian surat mati dilampirkan ke langganan. Pengiriman yang gagal tidak mengubah status langganan Anda. Untuk informasi selengkapnya, lihat Mencoba lagi pengiriman SNS pesan Amazon.

Kesalahan sisi klien

Kesalahan sisi klien dapat terjadi ketika Amazon SNS memiliki metadata langganan basi. Kesalahan ini biasanya terjadi ketika pemilik menghapus titik akhir (misalnya, fungsi Lambda yang berlangganan topik SNS Amazon) atau saat pemilik mengubah kebijakan yang dilampirkan ke titik akhir berlangganan dengan cara yang mencegah SNS Amazon mengirimkan pesan ke titik akhir. Amazon SNS tidak mencoba lagi pengiriman pesan yang gagal sebagai akibat dari kesalahan sisi klien.

Kesalahan sisi server

Kesalahan sisi server dapat terjadi ketika sistem yang bertanggung jawab atas titik akhir berlangganan menjadi tidak tersedia atau mengembalikan pengecualian yang menunjukkan bahwa sistem tidak dapat memproses permintaan yang valid dari Amazon. SNS Ketika kesalahan sisi server terjadi, Amazon SNS mencoba kembali pengiriman yang gagal menggunakan fungsi backoff linier atau eksponensial. Untuk kesalahan sisi server yang disebabkan oleh titik akhir AWS terkelola yang

didukung oleh Amazon atauSQS, AWS Lambda Amazon SNS mencoba ulang pengiriman hingga 100.015 kali, selama 23 hari.

Titik akhir yang dikelola pelanggan (sepertiHTTP,, SMTPSMS, atau push seluler) juga dapat menyebabkan kesalahan sisi server. Amazon SNS mencoba kembali pengiriman ke jenis titik akhir ini juga. Meskipun HTTP titik akhir mendukung kebijakan percobaan ulang yang ditentukan pelanggan, Amazon SNS menetapkan kebijakan coba ulang pengiriman internal hingga 50 kali selama 6 jam, for SMTPSMS, dan titik akhir push seluler.

Bagaimana cara kerja antrean surat mati?

Antrian surat mati dilampirkan ke SNS langganan Amazon (bukan topik) karena pengiriman pesan terjadi pada tingkat langganan. Hal ini memungkinkan Anda mengidentifikasi titik akhir target asli untuk setiap pesan dengan lebih mudah.

Antrian surat mati yang terkait dengan SNS langganan Amazon adalah antrian Amazon biasa. SQS Untuk informasi selengkapnya tentang periode retensi pesan, lihat Kuota Terkait Pesan di Panduan Developer Amazon Simple Queue Service. Anda dapat mengubah periode penyimpanan pesan menggunakan SQS `SetQueueAttributes` API tindakan Amazon. Agar aplikasi Anda lebih tangguh, sebaiknya pengaturan periode penyimpanan maksimum untuk antrean surat mati hingga 14 hari.

Bagaimana pesan dipindahkan ke antrean surat mati?

Pesan Anda dipindahkan ke antrean surat mati menggunakan kebijakan penggerak ulang. Kebijakan redrive adalah JSON objek yang mengacu pada ARN antrian huruf mati. `deadLetterTargetArn`Atribut menentukan. ARN ARNHarus menunjuk ke SQS antrian Amazon yang sama Akun AWS dan Wilayah sebagai SNS langganan Amazon Anda. Untuk informasi selengkapnya, lihat Mengonfigurasi antrian SNS surat mati Amazon untuk berlangganan.

JSONObjek berikut adalah contoh kebijakan redrive, yang dilampirkan ke SNS langganan.

```
{
  "deadLetterTargetArn": "arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue"
}
```

Bagaimana cara memindahkan pesan dari antrean surat mati?

Anda dapat memindahkan pesan dari antrean surat mati dalam dua cara:

- Hindari menulis logika SQS konsumen Amazon - Tetapkan antrian surat mati Anda sebagai sumber peristiwa ke fungsi Lambda untuk menguras antrian huruf mati Anda.
- Tulis logika SQS konsumen Amazon — Gunakan Amazon SQSAPI, AWS SDK, atau AWS CLI untuk menulis logika konsumen khusus untuk polling, pemrosesan, dan penghapusan pesan dalam antrean huruf mati.

Bagaimana saya bisa memantau dan mencatat antrean surat mati?

Anda dapat menggunakan CloudWatch metrik Amazon untuk memantau antrian surat mati yang terkait dengan langganan Amazon Anda. SNS Semua SQS antrian Amazon memancarkan CloudWatch metrik pada interval satu menit. Untuk informasi selengkapnya, lihat CloudWatch Metrik yang tersedia untuk Amazon SQS di Panduan Pengembang Layanan Antrian Sederhana Amazon. Semua SNS langganan Amazon dengan antrian surat mati juga memancarkan metrik. CloudWatch Untuk informasi selengkapnya, lihat Memantau SNS topik Amazon menggunakan CloudWatch.

Untuk mengetahui aktivitas dalam antrian surat mati, Anda dapat menggunakan metrik dan alarm. CloudWatch Menyiapkan alarm untuk `NumberOfMessagesSent` metrik tidak cocok karena metrik ini tidak menangkap pesan yang dikirim ke a DLQ sebagai akibat dari upaya pemrosesan yang gagal. Sebagai gantinya, gunakan `ApproximateNumberOfMessagesVisible` metrik, yang menangkap semua pesan yang saat ini tersedia diDLQ, termasuk yang dipindahkan karena kegagalan pemrosesan.

Contoh pengaturan CloudWatch alarm

1. Buat CloudWatchalarm untuk **`ApproximateNumberOfMessagesVisible`** metrik.
2. Atur ambang alarm ke 1 (atau nilai lain yang sesuai berdasarkan harapan dan DLQ lalu lintas Anda).
3. Tentukan SNS topik Amazon yang akan diberi tahu saat alarm berbunyi. SNSTopik Amazon ini dapat mengirimkan pemberitahuan alarm Anda ke jenis titik akhir apa pun (seperti alamat email, nomor telepon, atau aplikasi pager seluler).

Anda dapat menggunakan CloudWatch Log untuk menyelidiki pengecualian yang menyebabkan SNS pengiriman Amazon gagal dan pesan dikirim ke antrian surat mati. Amazon SNS dapat mencatat pengiriman yang berhasil dan gagal. CloudWatch Untuk informasi selengkapnya, lihat Atribut aplikasi SNS seluler Amazon.

Mengonfigurasi antrian SNS surat mati Amazon untuk berlangganan

Antrian surat mati adalah SQS antrian Amazon yang dapat ditargetkan oleh SNS langganan Amazon untuk pesan yang tidak dapat dikirim ke pelanggan dengan sukses. Pesan yang tidak dapat dikirim karena kesalahan klien atau kesalahan server disimpan dalam antrean surat mati untuk analisis lebih lanjut atau pemrosesan ulang. Untuk informasi selengkapnya, silakan lihat Antrian SNS surat mati Amazon dan Mencoba lagi pengiriman SNS pesan Amazon.

Halaman ini menunjukkan bagaimana Anda dapat menggunakan AWS Management Console, dan AWS SDK, yang AWS CLI, dan AWS CloudFormation untuk mengonfigurasi antrian huruf mati untuk langganan Amazon. SNS

> **Note**
>
> Untuk FIFOtopik, Anda dapat menggunakan SQS antrian Amazon sebagai antrian surat mati untuk langganan Amazon. SNS FIFOlangganan topik menggunakan FIFO antrian, dan langganan topik standar menggunakan antrian standar.

Prasyarat

Sebelum Anda mengonfigurasi antrean surat mati, selesaikan prasyarat berikut:

1. Buat SNS topik Amazon bernama`MyTopic`.
2. Buat SQS antrian Amazon bernama`MyEndpoint`, untuk digunakan sebagai titik akhir langganan AmazonSNS.
3. (Lewati untuk AWS CloudFormation) Berlangganan antrian ke topik.
4. Buat SQS antrian Amazon lain bernama`MyDeadLetterQueue`, untuk digunakan sebagai antrian huruf mati untuk langganan Amazon. SNS
5. Untuk memberikan akses SNS utama Amazon ke SQS API tindakan Amazon, tetapkan kebijakan antrean berikut untuk`MyDeadLetterQueue`.

```
{
  "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"
      }
    }
  }]
}
```

Topik

- Untuk mengonfigurasi antrian surat mati untuk langganan Amazon SNS menggunakan AWS Management Console
- Untuk mengonfigurasi antrian surat mati untuk langganan Amazon SNS menggunakan AWS SDK
- Untuk mengonfigurasi antrian surat mati untuk langganan Amazon SNS menggunakan AWS CLI
- Untuk mengonfigurasi antrian surat mati untuk langganan Amazon menggunakan SNS AWS CloudFormation

Untuk mengonfigurasi antrian surat mati untuk langganan Amazon SNS menggunakan AWS Management Console

Sebelum memulai tutorial ini, pastikan Anda menyelesaikan prasyarat.

1. Masuk ke SQSkonsol Amazon.
2. Buat SQS antrian Amazon atau gunakan antrean yang ada dan catat antrian pada tab Detail antrian, misalnya: ARN

   ```
   arn:aws:sqs:us-east-2:123456789012:MyDeadLetterQueue
   ```

3. Masuk ke SNSkonsol Amazon.
4. Di panel navigasi, pilih Berlangganan.
5. Pada halaman Berlangganan, pilih langganan yang ada, lalu pilih Edit.
6. Pada Edit ***1234a567-bc89-012d-3e45-6fg7h890123i***halaman, perluas bagian kebijakan Remrive (antrian huruf mati), lalu lakukan hal berikut:

   a. Pilih Diaktifkan.

b. Tentukan ARN SQS antrian Amazon.

7. Pilih Simpan perubahan.

Langganan Anda dikonfigurasi untuk menggunakan antrean surat mati.

Untuk mengonfigurasi antrian surat mati untuk langganan Amazon SNS menggunakan AWS SDK

Sebelum Anda menjalankan contoh ini, pastikan Anda menyelesaikan prasyarat.

Untuk menggunakan AWS SDK, Anda harus mengkonfigurasinya dengan kredensi Anda. Untuk informasi selengkapnya, lihat File konfigurasi dan kredensial bersama di Panduan Referensi Alat AWS SDKs dan Alat.

Contoh kode berikut menunjukkan cara menggunakan`SetSubscriptionAttributesRedrivePolicy`.

Java

SDKuntuk Java 1.x

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

```
// 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);
```

Untuk mengonfigurasi antrian surat mati untuk langganan Amazon SNS menggunakan AWS CLI

Sebelum memulai tutorial ini, pastikan Anda menyelesaikan prasyarat.

1. Pasang dan konfigurasikan AWS CLI. Untuk informasi selengkapnya, lihat Panduan Pengguna AWS Command Line Interface .
2. Gunakan perintah berikut ini.

```
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\"}"
```

Untuk mengonfigurasi antrian surat mati untuk langganan Amazon menggunakan SNS AWS CloudFormation

Sebelum memulai tutorial ini, pastikan Anda menyelesaikan prasyarat.

1. Salin JSON kode berikut ke file bernama`MyDeadLetterQueue.json`.

```
{
  "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. Masuk ke konsol AWS CloudFormation tersebut.
3. Pada halaman Pilih templat, pilih Unggah templat ke Amazon S3, pilih file `MyDeadLetterQueue.json` Anda, dan kemudian pilih Selanjutnya.
4. Pada halaman Tentukan Detail, masukkan `MyDeadLetterQueue` untuk Nama Tumpukan, lalu pilih Selanjutnya.
5. Pada halaman Opsi, pilih Selanjutnya.
6. Pada halaman Tinjau, pilih Buat.

   AWS CloudFormation mulai membuat `MyDeadLetterQueue` tumpukan dan menampilkan status CREATE_IN_ PROGRESS. Ketika proses selesai, AWS CloudFormation menampilkan COMPLETE status CREATE_.

Pengarsipan SNS pesan Amazon, pemutaran ulang, dan analitik

Topik SNS standar Amazon mendukung pengarsipan pesan melalui Amazon Data Firehose. Anda dapat menyebarkan notifikasi ke aliran pengiriman Firehose, yang memungkinkan Anda mengirim notifikasi ke tujuan penyimpanan dan analitik yang didukung Firehose, termasuk Amazon Simple Storage Service (Amazon S3), Amazon Redshift, dan banyak lagi.

SNSFIFOTopik Amazon mendukung arsip pesan di tempat, tanpa kode, yang memungkinkan pemilik topik menyimpan (atau mengarsipkan) pesan yang dipublikasikan ke topik hingga 365 hari. Untuk topik yang aktif`ArchivePolicy`, pelanggan kemudian dapat membuat `ReplayPolicy` untuk mengambil (atau memutar ulang) pesan yang diarsipkan kembali ke titik akhir berlangganan. Untuk mempelajari lebih lanjut tentang fitur ini, lihatPengarsipan dan pemutaran ulang SNS pesan Amazon untuk topik FIFO.

Fitur	Topik Standar	FIFOTopik
Pengarsipan pesan	Aliran pengiriman Fanout ke Firehose	Pengarsipan SNS pesan Amazon untuk pemilik FIFO topik
Putar ulang pesan	Putar ulang untuk topik standar bukanlah fitur bawaan. Banyak pelanggan membangun sendiri berdasarkan arsip pesan mereka.	Putar ulang SNS pesan Amazon untuk pelanggan FIFO topik

Manajemen dan pengoptimalan sumber daya di Amazon SNS

Topik ini memberikan panduan tentang cara memanfaatkan potensi penuh Amazon SNS dengan memastikan kinerja optimal, mengurangi biaya yang tidak perlu, dan mempertahankan sumber daya yang terorganisir dengan baik.

Topik

Penandaan SNS topik Amazon

Amazon SNS mendukung penandaan SNS topik Amazon. Ini dapat membantu Anda melacak dan mengelola biaya yang terkait dengan topik Anda, memberikan keamanan yang ditingkatkan dalam kebijakan AWS Identity and Access Management (IAM) Anda, dan memungkinkan Anda dengan mudah mencari atau memfilter ribuan topik. Penandaan memungkinkan Anda mengelola SNS topik Amazon menggunakan AWS Resource Groups. Untuk informasi selengkapnya tentang Resource Groups, lihat Panduan Pengguna AWS Resource Groups.

Topik

Penandaan untuk alokasi biaya

Untuk mengatur dan mengidentifikasi SNS topik Amazon Anda untuk alokasi biaya, Anda dapat menambahkan tag yang mengidentifikasi tujuan topik. Ini sangat berguna ketika Anda memiliki banyak topik. Anda dapat menggunakan tag alokasi biaya untuk mengatur AWS tagihan Anda untuk mencerminkan struktur biaya Anda sendiri. Untuk melakukan ini, daftar untuk mendapatkan tagihan AWS akun Anda untuk menyertakan kunci tag dan nilai. Untuk informasi selengkapnya, lihat Menyiapkan Laporan Alokasi Biaya Bulanan di Panduan Pengguna AWS Billing and Cost Management.

Misalnya, Anda dapat menambahkan tag yang mewakili pusat biaya dan tujuan SNS topik Amazon Anda, sebagai berikut:

Sumber Daya	Kunci	Nilai
Topik 1	Pusat Biaya	43289
	Aplikasi	Proses pemesanan
Topik 2	Pusat Biaya	43289
	Aplikasi	Pemrosesan pembayaran
Topik 3	Pusat Biaya	76585
	Aplikasi	Pengarsipan

Skema penandaan ini memungkinkan Anda mengelompokkan dua topik yang melakukan tugas terkait di pusat biaya yang sama, sambil menandai aktivitas yang tidak terkait dengan tag alokasi biaya yang berbeda.

Penandaan untuk kontrol akses

AWS Identity and Access Management mendukung pengendalian akses ke sumber daya berdasarkan tag. Setelah menandai sumber daya Anda, berikan informasi tentang tag sumber daya Anda di elemen kondisi IAM kebijakan untuk mengelola akses berbasis tag. Untuk informasi tentang cara menandai sumber daya Anda menggunakan SNSkonsol Amazon atau AWS SDK, lihat Mengonfigurasi tag.

Anda dapat membatasi akses untuk IAM identitas. Misalnya, Anda dapat membatasi `Publish` dan `PublishBatch` mengakses semua SNS topik Amazon yang menyertakan tag dengan kunci `environment` dan nilainya`production`, sambil mengizinkan akses ke semua SNS topik Amazon lainnya. Dalam contoh di bawah ini, kebijakan membatasi kemampuan untuk mempublikasikan pesan ke topik yang ditandai`production`, sambil mengizinkan pesan dipublikasikan ke topik yang ditandai. `development` Untuk informasi selengkapnya, lihat Mengontrol Akses Menggunakan Tag di Panduan IAM Pengguna.

> **Note**
>
> Mengatur IAM izin untuk `Publish` menetapkan izin untuk keduanya `Publish` dan`PublishBatch`.

```
{
  "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"
      }
    }
  }]
}
```

Penandaan untuk pencarian dan pemfilteran sumber daya

AWS Akun dapat memiliki puluhan ribu SNS topik Amazon (lihat SNSKuota Amazon untuk detailnya). Dengan menandai topik Anda, Anda dapat menyederhanakan proses mencari atau memfilter topik.

Misalnya, Anda mungkin memiliki ratusan topik yang terkait dengan lingkungan produksi Anda. Daripada harus mencari topik ini secara manual, Anda dapat menanyakan semua topik dengan tag yang diberikan:

```
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());
    }
}
```

Mengonfigurasi tag SNS topik Amazon

Halaman ini menunjukkan bagaimana Anda dapat menggunakan tag AWS SDK, dan AWS CLI untuk mengkonfigurasi tag untuk SNStopik Amazon. AWS Management Console

> **Important**
>
> Jangan menambahkan informasi identitas pribadi (PII) atau informasi rahasia atau sensitif lainnya dalam tag. Tag dapat diakses oleh Amazon Web Services lainnya, termasuk penagihan. Tag tidak dimaksudkan untuk digunakan dalam data sensitif atau privat.

Topik

- Membuat daftar, menambahkan, dan menghapus tag untuk SNS topik Amazon menggunakan AWS Management Console
- Menambahkan tag ke topik menggunakan AWS SDK
- Mengelola tag dengan SNS API tindakan Amazon
- APItindakan yang mendukung ABAC

Membuat daftar, menambahkan, dan menghapus tag untuk SNS topik Amazon menggunakan AWS Management Console

1. Masuk ke SNSkonsol Amazon.
2. Di panel navigasi, pilih Topics (Topik).
3. Di halaman Topics (Topik), pilih topik dan kemudian pilih Edit (Edit).
4. Perluas bagian Tags (Tag).

   Tag yang ditambahkan ke topik terdaftar.
5. Modifikasi tag topik:
   - Untuk menambahkan tag, pilih Tambah tag dan masukkan Kunci dan Nilai (opsional).
   - Untuk menghapus tag, pilih Remove tag (Hapus tag) di samping pasangan nilai kunci.
6. Pilih Simpan perubahan.

Menambahkan tag ke topik menggunakan AWS SDK

Untuk menggunakan AWS SDK, Anda harus mengkonfigurasinya dengan kredensional Anda. Untuk informasi selengkapnya, lihat File konfigurasi dan kredensial bersama di Panduan Referensi Alat AWS SDKs dan Alat.

Contoh kode berikut menunjukkan cara menggunakan`TagResource`.

CLI

AWS CLI

Untuk menambahkan tag ke topik

`tag-resource`Contoh berikut menambahkan tag metadata ke topik Amazon SNS yang ditentukan.

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

Perintah ini tidak menghasilkan output.

- Untuk API detailnya, lihat TagResourcedi Referensi AWS CLI Perintah.

Java

SDKuntuk Java 2.x

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

```
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);
        }
    }
}
```

- Untuk API detailnya, lihat TagResourcedi AWS SDK for Java 2.x APIReferensi.

Kotlin

SDKuntuk Kotlin

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

```
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")
    }
}
```

- Untuk API detailnya, lihat TagResource AWSSDKAPIreferensi Kotlin.

Mengelola tag dengan SNS API tindakan Amazon

Untuk mengelola tag menggunakan Amazon SNSAPI, gunakan API tindakan berikut:

- `ListTagsForResource`
- `TagResource`
- `UntagResource`

APItindakan yang mendukung ABAC

Berikut ini adalah daftar API tindakan yang mendukung kontrol akses berbasis atribut ()ABAC. Untuk detail selengkapnyaABAC, lihat ABACUntuk apa AWS? dalam IAMUser Guide.

- `AddPermission`
- `ConfirmSubscription`
- `DeleteTopic`
- `GetDataProtectionPolicy`
- `GetSubscriptionAttributes`
- `GetTopicAttributes`

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

Sumber dan tujuan SNS acara Amazon

Amazon SNS menghubungkan Layanan AWS dan sistem eksternal dengan merutekan notifikasi berbasis peristiwa. Amazon SNS menerima berbagai peristiwa Layanan AWS, seperti pembaruan saluran data, tindakan EC2 penskalaan Amazon, atau peringatan keamanan, dan menerbitkan peristiwa ini ke topik Amazon. SNS Topik-topik ini kemudian mengirim pemberitahuan ke tujuan yang ditentukan.

Amazon SNS mendukung dua jenis tujuan utama: Application-to-Application (A2A) dan Application-to-Person (A2P). Dalam pesan A2A, Amazon SNS dapat mengirim peristiwa ke Lambda untuk memicu logika bisnis khusus, ke Amazon untuk mengantri pesan, dan ke Amazon Kinesis Data Firehose SQS untuk streaming data ke layanan penyimpanan dan analitik. Untuk pesan A2P, Amazon SNS dapat mengirim notifikasi melaluiSMS, email, dan pemberitahuan push ke perangkat seluler, memastikan bahwa pengguna atau tim menerima peringatan tepat waktu.

Dengan bertindak sebagai hub pusat, Amazon SNS merutekan notifikasi ke tempat yang tepat, membantu Anda mengotomatiskan dan mengelola AWS infrastruktur dengan lebih efektif. Pengaturan ini memungkinkan integrasi yang mulus antara layanan dan komunikasi yang andal dengan pengguna dan sistem.

Topik

Sumber SNS acara Amazon

Amazon SNS terintegrasi dengan Layanan AWS berbagai macam kategori, memungkinkan layanan ini mempublikasikan acara ke SNS topik Amazon. Integrasi ini menyediakan pemberitahuan real-time dari peristiwa penting, seperti perubahan infrastruktur, kinerja aplikasi, dan manajemen biaya.

> **Note**
>
> Amazon SNS memperkenalkan FIFOtopik pada Oktober 2020. Saat ini, sebagian besar AWS layanan mendukung pengiriman acara ke topik standar saja.

Topik

Layanan analitik

Tabel berikut menjelaskan cara Amazon SNS berintegrasi dengan layanan AWS analitik seperti Athena AWS Data Pipeline, dan Amazon Redshift untuk memberikan notifikasi real-time untuk peristiwa penting, termasuk pelanggaran batas kontrol, pembaruan status pipeline, dan aktivitas gudang data.

Anda dapat memanfaatkan integrasi ini untuk mengotomatiskan respons dan mempertahankan pengawasan yang efektif atas operasi data Anda.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
Amazon Athena - Memungkinkan Anda menganalisis data di Amazon S3 menggunakan standar. SQL	Menerima notifikasi bila batas kontrol terlampaui. Untuk informasi selengkapnya, lihat Menetapkan batas kontrol penggunaan data di Panduan Pengguna Amazon Athena.
AWS Data Pipeline – Membantu mengotomatisasi pergerakan dan transformasi data.	Menerima notifikasi tentang status komponen alur. Untuk informasi selengkapnya, lihat SnsAlarmdi Panduan AWS Data Pipeline Pengembang.
Amazon Redshift – Mengelola semua pekerjaan pengaturan, pengoperasian, dan penskalaan gudang data.	Menerima notifikasi kejadian Amazon Redshift. Untuk informasi selengkapnya, lihat notifikasi peristiwa Amazon Redshift di Panduan Manajemen Pergeseran Merah Amazon.

Layanan integrasi aplikasi

Tabel berikut menjelaskan bagaimana Amazon SNS terintegrasi dengan layanan integrasi aplikasi seperti EventBridge dan AWS Step Functions, memungkinkan perutean data real-time dan pemberitahuan untuk aplikasi bisnis penting.

Anda dapat memanfaatkan integrasi ini untuk menerima peringatan dari EventBridge peristiwa dan mengatur alur kerja menggunakan Step Functions, meningkatkan otomatisasi dan daya tanggap aplikasi Anda.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
Amazon EventBridge — Mengirimkan aliran data real-time dari aplikasi Anda sendiri, aplikasi software-as-a-service (SaaS), AWS dan layanan serta rute data tersebut ke target, termasuk Amazon. SNS EventBridge Dulunya bernama CloudWatch Events.	Menerima pemberitahuan EventBridge acara. Untuk informasi selengkapnya, lihat EventBridgeTarget Amazon di Panduan EventBridge Pengguna Amazon.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
AWS Step Functions— Memungkinkan Anda menggabungkan AWS Lambda fungsi dan AWS layanan lain untuk membangun aplikasi bisnis yang penting.	Menerima notifikasi kejadian Step Functions. Untuk informasi selengkapnya, lihat Hubungi Amazon SNS dengan Step Functions di Panduan AWS Step Functions Pengembang.

Layanan manajemen penagihan & biaya

Tabel berikut menjelaskan bagaimana AWS Billing and Cost Management terintegrasi dengan Amazon SNS untuk memberikan pemberitahuan anggaran, perubahan harga, dan anomali biaya.

Anda dapat memanfaatkan integrasi ini untuk menyiapkan SNS topik Amazon untuk menerima peringatan waktu nyata tentang AWS pengeluaran Anda, membantu Anda memantau biaya dan menanggapi biaya tak terduga secara efisien.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
AWS Billing and Cost Management – Menyediakan fitur yang membantu Anda memantau biaya dan membayar tagihan Anda.	Menerima notifikasi anggaran, notifikasi perubahan harga, dan pemberitahuan anomali. Untuk informasi selengkapnya, lihat halaman berikut di Panduan Pengguna AWS Billing : • Membuat SNS topik Amazon untuk pemberitahuan anggaran • Menyiapkan notifikasi • Mendeteksi pengeluaran yang tidak biasa dengan AWS Cost Anomaly Detection

Layanan aplikasi bisnis

Tabel berikut menjelaskan bagaimana Amazon Chime terintegrasi dengan Amazon SNS untuk mengirim pemberitahuan untuk acara rapat penting, memungkinkan Anda untuk tetap mendapat informasi tentang komunikasi dan penjadwalan Anda.

Anda dapat memanfaatkan integrasi ini untuk memanfaatkan notifikasi acara Amazon SDK Chime untuk menyempurnakan alat kolaborasi Anda di dalam dan di luar organisasi Anda.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
Amazon Chime – Memungkinkan Anda untuk bertemu, mengobrol, dan melakukan panggilan bisnis di dalam dan di luar organisasi Anda.	Menerima notifikasi acara pertemuan penting. Untuk informasi selengkapnya, lihat notifikasi SDK peristiwa Amazon Chime di Panduan Pengembang Amazon Chime.

Layanan komputer

Tabel berikut menjelaskan cara Amazon SNS berintegrasi dengan berbagai layanan AWS komputasi, memungkinkan Anda menerima pemberitahuan untuk peristiwa penting seperti tindakan Auto Scaling, penyelesaian Image BuilderEC2, perubahan lingkungan Elastic Beanstalk, output fungsi Lambda, dan ambang metrik Lightsail.

Anda dapat memanfaatkan integrasi ini untuk mengelola aplikasi dan sumber daya Anda secara efisien dengan tetap mendapat informasi tentang pembaruan dan tindakan penting. Layanan AWS

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
Amazon EC2 Auto Scaling - Membantu Anda memiliki jumlah instans Amazon Elastic Compute Cloud (AmazonEC2) yang benar yang tersedia untuk menangani pemuatan aplikasi Anda.	Terima pemberitahuan saat Auto Scaling meluncurkan atau menghentikan EC2 instans Amazon di grup Auto Scaling Anda. Untuk informasi selengkapnya, lihat Mendapatkan SNS notifikasi Amazon saat grup Auto Scaling Anda menskalakan di Panduan Pengguna Amazon Auto EC2 Scaling.
EC2Image Builder - Membantu mengotoma tiskan pembuatan, pengelolaan, dan penyebara n gambar yang disesuaikan, aman, dan up-to-date server yang sudah diinstal sebelumny a dan dikonfigurasi sebelumnya dengan	Menerima notifikasi saat membangun selesai. Untuk informasi selengkapnya, lihat Melacak gambar server terbaru di pipeline EC2 Image Builder di AWS Compute Blog.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
perangkat lunak dan pengaturan untuk memenuhi standar TI tertentu.	
AWS Elastic Beanstalk – Menangani rincian penyediaan kapasitas, penyeimbangan beban, dan penskalaan untuk aplikasi Anda, dan menyediakan pemantauan kesehatan aplikasi.	Menerima notifikasi kejadian penting yang mempengaruhi aplikasi Anda. Untuk informasi selengkapnya, lihat pemberitahuan lingkungan Elastic Beanstalk SNS dengan Amazon AWS Elastic Beanstalk di Panduan Pengembang.
AWS Lambda – Memungkinkan Anda menjalankan kode tanpa penyediaan atau pengelolaan server.	Menerima data output fungsi dengan menetapkan SNS topik sebagai antrian huruf mati Lambda atau tujuan Lambda. Untuk informasi selengkapnya, lihat Invokasi asinkron di Panduan Developer AWS Lambda .
Amazon Lightsail — Membantu pengembang mulai AWS menggunakan untuk membangun situs web atau aplikasi web.	Menerima notifikasi ketika metrik untuk salah satu instans, basis data, atau penyeimbang beban Anda melebihi ambang batas yang telah ditentukan. Untuk informasi selengkapnya, lihat Menambahkan kontak notifikasi di Amazon Lightsail di Panduan Developer Amazon Lightsail.

Layanan kontainer

Tabel berikut menjelaskan bagaimana Amazon SNS terintegrasi dengan layanan AWS kontainer seperti Amazon EKS Distro dan AmazonECS, memungkinkan Anda melacak pembaruan dan tambalan keamanan untuk EKS kluster Amazon dan menerima pemberitahuan untuk rilis baru yang dioptimalkan. ECS AMI

Anda dapat memanfaatkan integrasi ini untuk menjaga keamanan dan efisiensi penerapan container Anda dengan tetap mendapat informasi tentang pembaruan dan perubahan penting.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
Amazon EKS Distro - Memungkinkan Anda membuat cluster yang andal dan aman di mana pun aplikasi Anda digunakan.	Lacak pembaruan dan tambalan keamanan untuk cluster yang dibuat dengan Amazon EKS Distro. Untuk informasi selengkapnya, lihat Memperkenalkan Amazon EKS Distro - distribusi Kubernetes open source yang digunakan oleh Amazon. EKS
Amazon Elastic Container Service (AmazonECS) — Memungkinkan Anda menjalankan, menghentikan, dan mengelola kontainer di klaster.	Terima pemberitahuan saat Amazon baru yang ECS dioptimalkan AMI tersedia. Untuk informasi selengkapnya, lihat Berlangganan notifikasi AMI pembaruan ECS yang dioptimalkan Amazon di Panduan Pengembang Layanan Kontainer Elastis Amazon.

Layanan keterlibatan pelanggan

Tabel berikut menjelaskan cara Amazon SNS meningkatkan layanan keterlibatan pelanggan dengan mengintegrasikan dengan Amazon Connect AWS Olah Pesan Pengguna Akhir SMS, dan Amazon Simple Email Service (SES), memungkinkan Anda menerima peringatan dan validasi, mengonfigurasi SMS pesan dua arah, dan memantau pemberitahuan email untuk pantulan, keluhan, dan pengiriman.

Integrasi ini membantu Anda mengelola komunikasi pelanggan di berbagai saluran.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
Amazon Connect – Memungkinkan Anda menyiapkan pusat kontak cloud omnichannel untuk berinteraksi dengan pelanggan Anda.	Menerima pemberitahuan dan validasi. Untuk informasi selengkapnya, lihat Kekuatan AWS Amazon Connect di Panduan Administrator Amazon Connect.
AWS Olah Pesan Pengguna Akhir SMS— Membantu Anda melibatkan pelanggan Anda dengan mengirimi mereka email, SMS dan pesan suara, dan pemberitahuan push.	Konfigurasikan dua arahSMS, yang memungkinkan Anda menerima pesan dari pelanggan Anda. Untuk informasi selengkapnya, lihat SMSPesan dua arah di Panduan AWS Olah Pesan Pengguna Akhir SMS Pengguna.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
Amazon Simple Email Service (AmazonSES) - Menyediakan cara hemat biaya bagi Anda untuk mengirim dan menerima email menggunakan alamat email dan domain Anda sendiri.	Menerima notifikasi dari pentalan, aduan, dan pengiriman. Untuk informasi selengkapnya, lihat Mengonfigurasi SNS notifikasi Amazon untuk Amazon SES di Panduan Pengembang Layanan Email Sederhana Amazon.

Layanan basis data

Tabel berikut menjelaskan cara Amazon SNS berintegrasi dengan layanan AWS database seperti AWS Database Migration Service (DMS), Amazon DynamoDB, Amazon ElastiCache, Amazon Neptune, Amazon Redshift, dan Amazon Relational Database RDS Service () untuk mengirim pemberitahuan tentang peristiwa penting seperti migrasi data, aktivitas pemeliharaan, pembaruan cache, dan perubahan database.

Integrasi ini membantu Anda memantau dan mengelola lingkungan database Anda secara lebih efektif dengan memberikan peringatan tepat waktu tentang peristiwa operasional utama.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
AWS Database Migration Service— Memigrasi data dari database lokal ke file. AWS Cloud	Menerima pemberitahuan ketika AWS DMS peristiwa terjadi; misalnya, ketika instance replikasi dibuat atau dihapus. Untuk informasi selengkapnya, lihat Bekerja dengan kejadian dan notifikasi di AWS Database Migration Service di Panduan Pengguna AWS Database Migration Service .
Amazon DynamoDB - Memberikan kinerja yang cepat dan dapat diprediksi dengan skalabilitas yang mulus dalam layanan basis data Tanpa terkelola sepenuhnya ini. SQL	Menerima notifikasi ketika kejadian pemelihar aan terjadi. Untuk informasi selengkapnya, lihat Menyesuaikan setelan DAX klaster di Panduan Pengembang Amazon DynamoDB.
Amazon ElastiCache - Menyediakan cache dalam memori berkinerja tinggi, dapat diubah ukurannya, dan hemat biaya, sekaligus	Menerima notifikasi ketika kejadian penting terjadi. Untuk informasi selengkapnya, lihat Pemberitahuan acara dan Amazon SNS di

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
menghilangkan kompleksitas yang terkait dengan penerapan dan pengelolaan lingkungan cache terdistribusi.	Panduan Pengguna Amazon ElastiCache (Memcached).
Amazon Neptune – Memungkinkan Anda untuk membangun dan menjalankan aplikasi yang berfugnsi dengan set data yang sangat terhubung.	Menerima notifikasi ketika kejadian Neptune terjadi. Untuk informasi selengkapnya, lihat Menggunakan notifikasi kejadian Neptune di Panduan Pengguna Neptune.
Amazon Redshift – Mengelola semua pekerjaan pengaturan, pengoperasian, dan penskalaan gudang data.	Menerima notifikasi kejadian Amazon Redshift. Untuk informasi selengkapnya, lihat notifikasi peristiwa Amazon Redshift di Panduan Manajemen Pergeseran Merah Amazon.
Amazon Relational Database Service — Memudahkan pengaturan, pengoperasian, dan skala database relasional di AWS Cloud.	Terima pemberitahuan RDS acara Amazon. Untuk informasi selengkapnya, lihat Menggunakan pemberitahuan RDS acara Amazon di Panduan RDS Pengguna Amazon.

Layanan alat developer

Tabel berikut menjelaskan cara Amazon SNS berintegrasi dengan layanan alat AWS pengembang, seperti AWS CodeBuild,, AWS CodeCommit, Amazon AWS CodeDeploy, dan CodeGuru AWS CodePipeline AWS CodeStar, untuk memberikan pemberitahuan tentang peristiwa penting seperti perubahan status build, pembaruan repositori, kemajuan penerapan, anomali kinerja, dan tindakan pipeline.

Integrasi ini membantu Anda memantau dan mengelola alur kerja pengembangan perangkat lunak secara efisien dengan menerima peringatan tepat waktu tentang peristiwa penting.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
AWS CodeBuild – Mengompilasi kode sumber Anda, menjalankan pengujian unit, dan menghasilkan artefak yang siap di-deploy.	Menerima notifikasi ketika membangun berhasil, gagal, atau beralih dari satu fase membangun ke fase yang lain. Untuk informasi selengkapnya, lihat Contoh pemberitahuan

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
	pembuatan untuk CodeBuild di Panduan AWS CodeBuild Pengguna.
AWS CodeCommit – Menyediakan kontrol versi untuk menyimpan dan mengelola aset secara privat di cloud.	Menerima pemberitahuan tentang CodeCommit acara repositori. Untuk informasi selengkapnya, lihat Contoh: Membuat AWS CodeCommit pemicu untuk SNS topik Amazon di Panduan AWS CodeCommit Pengguna.
AWS CodeDeploy— Mengotomatiskan penerapan aplikasi ke EC2 instans Amazon, instans lokal, fungsi Lambda tanpa server, atau layanan Amazon. ECS	Menerima pemberitahuan untuk CodeDeploy penyebaran atau kejadian instance. Untuk informasi selengkapnya, lihat Membuat pemicu untuk suatu CodeDeploy peristiwa di Panduan AWS CodeDeploy Pengguna.
Amazon CodeGuru — Mengumpulkan data kinerja runtime dari aplikasi live Anda, dan memberikan rekomendasi yang dapat membantu Anda menyempurnakan kinerja aplikasi.	Menerima notifikasi saat anomali terjadi. Untuk informasi selengkapnya, lihat Bekerja dengan anomali dan laporan rekomendasi di CodeGuru Panduan Pengguna Amazon.
AWS CodePipeline – Mengotomatiskan langkah-langkah yang diperlukan untuk merilis perubahan perangkat lunak secara berkelanjutan.	Menerima notifikasi tentang tindakan persetujuan. Untuk informasi selengkapnya, lihat Mengelola tindakan persetujuan CodePipeline di Panduan AWS CodePipeline Pengguna.
AWS CodeStar— Buat, kelola, dan bekerja dengan proyek pengembangan perangkat lunak di AWS.	Menerima notifikasi tentang kejadian yang terjadi di sumber daya yang Anda gunakan. Untuk informasi selengkapnya, lihat Mengonfigurasi SNS topik Amazon untuk pemberitahuan di Panduan Pengguna Konsol Alat Pengembang.

Layanan web & seluler front-end

Tabel berikut menjelaskan bagaimana Amazon SNS terintegrasi dengan AWS Olah Pesan Pengguna Akhir SMS untuk meningkatkan keterlibatan pelanggan dengan mengirim email, pesan suaraSMS, dan pemberitahuan push, termasuk kemampuan untuk mengonfigurasi dua arah SMS untuk menerima pesan pelanggan.

Integrasi ini memungkinkan Anda untuk berinteraksi lebih efektif dengan pelanggan Anda di berbagai saluran komunikasi.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
AWS Olah Pesan Pengguna Akhir SMS— Membantu Anda melibatkan pelanggan Anda dengan mengirimi mereka email, SMS dan pesan suara, dan pemberitahuan push.	Konfigurasikan dua arahSMS, yang memungkinkan Anda menerima pesan dari pelanggan Anda. Untuk informasi selengkapnya, lihat SMSPesan dua arah di Panduan AWS Olah Pesan Pengguna Akhir SMS Pengguna.

Layanan pengembangan permainan

Tabel berikut menjelaskan bagaimana Amazon SNS berintegrasi dengan Amazon GameLift untuk memberikan pemberitahuan untuk perjodohan dan peristiwa antrian di server game multipemain berbasis sesi.

Integrasi ini membantu pengembang game mengotomatiskan dan memantau penyebaran, pengoperasian, dan penskalaan server game mereka, memastikan pengalaman bermain game yang mulus.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
Amazon GameLift — Menyediakan solusi untuk hosting server game multipemain berbasis sesi di cloud, termasuk layanan yang dikelola sepenuhnya untuk menyebarkan, mengoperasikan, dan menskalakan server game.	Menerima notifikasi kejadian matchmaking dan antrian. Untuk informasi selengkapnya, lihat halaman berikut: • Untuk pemberitahuan perjodohan, lihat Mengatur pemberitahuan FlexMatch acara di

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
	Panduan GameLift FlexMatch Pengembang Amazon. • Untuk pemberitahuan antrian, lihat Mengatur notifikasi acara untuk penempatan sesi game di Panduan GameLift Pengembang Amazon.

Layanan Internet of Things

Tabel berikut menjelaskan bagaimana Amazon SNS terintegrasi dengan AWS IoT layanan, seperti,,, AWS IoT Core, dan AWS IoT Device Defender AWS IoT Events AWS IoT Greengrass, untuk memberikan pemberitahuan untuk peristiwa dan peringatan IoT.

Integrasi ini memungkinkan Anda memantau perilaku perangkat secara efektif, menerima peringatan untuk aktivitas abnormal, dan mengelola perangkat IoT dengan pembaruan dan tindakan waktu nyata.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
AWS IoT CoreMenyediakan layanan cloud yang menghubungkan perangkat IoT Anda ke perangkat dan AWS Cloud layanan lain.	Menerima pemberitahuan AWS IoT Core acara. Untuk informasi selengkapnya, lihat Membuat SNS aturan Amazon di Panduan AWS IoT Pengembang.
AWS IoT Device Defender – Memungkinkan Anda untuk mengaudit konfigurasi perangkat Anda, memantau perangkat yang terhubung untuk mendeteksi perilaku abnormal, dan mengurangi risiko keamanan.	Menerima alarm saat perangkat melanggar sebuah perilaku. Untuk informasi selengkap nya, lihat Cara menggunakan AWS IoT Device Defender deteksi di Panduan AWS IoT Pengembang.
AWS IoT Events – Memungkinkan Anda memantau peralatan atau armada perangkat Anda untuk kegagalan atau perubahan dalam operasi, dan memicu tindakan ketika kejadian tersebut terjadi.	Menerima pemberitahuan AWS IoT Events acara. Untuk informasi selengkapnya, lihat Amazon Simple Notification Service di Panduan Developer AWS IoT Events .

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
AWS IoT Greengrass— Memperluas AWS ke perangkat fisik sehingga mereka dapat bertindak secara lokal pada data yang mereka hasilkan, sambil tetap menggunakan cloud untuk manajemen, analitik, dan penyimpanan yang tahan lama.	Menerima pemberitahuan AWS IoT Greengras s acara. Untuk informasi selengkapnya, lihat SNSkonektor di Panduan AWS IoT Greengrass Version 1 Pengembang.

Layanan Machine Learning

Tabel berikut menjelaskan bagaimana Amazon SNS berintegrasi dengan layanan pembelajaran AWS mesin, seperti Amazon, Amazon DevOps Guru CodeGuru, Amazon Lookout for Metrics, Amazon Rekognition, dan Amazon, untuk memberikan pemberitahuan tentang anomali, wawasan operasional, dan SageMaker aktivitas pelabelan data.

Integrasi ini memungkinkan Anda memantau kinerja aplikasi, menerima peringatan untuk penyimpangan data, dan merampingkan penerapan model pembelajaran mesin dengan pembaruan waktu nyata.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
Amazon CodeGuru — Mengumpulkan data kinerja runtime dari aplikasi live Anda, dan memberikan rekomendasi yang dapat membantu Anda menyempurnakan kinerja aplikasi.	Menerima notifikasi saat anomali terjadi. Untuk informasi selengkapnya, lihat Bekerja dengan anomali dan laporan rekomendasi di CodeGuru Panduan Pengguna Amazon.
Amazon DevOps Guru — Menghasilkan wawasan operasional menggunakan pembelaja ran mesin untuk membantu Anda meningkat kan kinerja aplikasi operasional Anda.	Wawasan dan konfirmasi ke depan. Untuk informasi selengkapnya, lihat Memberika n wawasan operasional yang didukung MP ke tim panggilan Anda menggunakan PagerDuty Amazon DevOps Guru di Blog AWS Manajemen & Tata Kelola.
Amazon Lookout for Metrics – Menemukan anomali dalam data Anda, menentukan akar	Menerima notifikasi anomali. Untuk informasi selengkapnya, lihat Menggunakan Amazon

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
penyebabnya, dan memungkinkan Anda untuk segera mengambil tindakan.	SNS dengan Lookout for Metrics di Panduan Pengembang Amazon Lookout for Metrics.
Amazon Rekognition – Memungkinkan Anda menambahkan analisis citra dan video ke aplikasi Anda	Menerima notifikasi hasil permintaan. Untuk informasi selengkapnya, lihat Referensi: Notifikasi hasil analisis video di Panduan Developer Amazon Rekognition.
Amazon SageMaker - Memungkinkan ilmuwan data dan pengembang untuk membangun dan melatih model pembelajaran mesin, dan kemudian langsung menerapkannya ke lingkungan host yang siap produksi.	Menerima notifikasi ketika objek data diberi label. Untuk informasi selengkapnya, lihat Membuat pekerjaan pelabelan streaming di Panduan SageMaker Pengembang Amazon.

Layanan manajemen dan tata kelola

Tabel berikut menjelaskan cara Amazon SNS berintegrasi dengan layanan AWS manajemen dan tata kelola seperti AWS Chatbot,, AWS CloudFormation, CloudTrail, CloudWatch,, AWS Config, AWS Control Tower, AWS License Manager AWS Service Catalog, dan AWS Systems Manager, memberikan pemberitahuan untuk peristiwa penting seperti perubahan infrastruktur, peringatan kepatuhan, dan wawasan operasional.

Integrasi ini membantu Anda memantau dan mengelola AWS lingkungan Anda secara efisien dengan memberikan peringatan dan pembaruan tepat waktu ke tim dan sistem yang relevan.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
AWS Chatbot— Memungkinkan DevOps dan tim pengembangan perangkat lunak untuk menggunakan ruang obrolan Amazon Chime dan Slack untuk memantau dan menanggapi peristiwa operasional di. AWS Cloud	Mengirimkan notifikasi ke ruang obrolan. Untuk informasi selengkapnya, lihat Menyiapkan AWS Chatbot di Panduan Administrator AWS Chatbot .
AWS CloudFormation— Memungkinkan Anda untuk membuat dan menyediakan penyebara	Menerima notifikasi saat tumpukan dibuat dan diperbarui. Untuk informasi selengkapnya, lihat Pengaturan opsi tumpukan AWS CloudForm

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
n AWS infrastruktur yang dapat diprediksi dan berulang kali.	ation di Panduan Pengguna AWS CloudForm ation .
AWS CloudTrail – Menyediakan riwayat kejadian dari kegiatan Akun AWS Anda.	Terima pemberitahuan saat CloudTrail menerbitkan file log baru ke bucket Amazon S3 Anda. Untuk informasi selengkapnya, lihat Mengonfigurasi SNS notifikasi Amazon CloudTrail di Panduan AWS CloudTrail Pengguna.
Amazon CloudWatch — Memantau AWS sumber daya Anda dan aplikasi yang Anda jalankan AWS secara real time.	Menerima notifikasi ketika status alarm berubah. Untuk informasi selengkapnya, lihat Menggunakan CloudWatch alarm Amazon di Panduan CloudWatch Pengguna Amazon.
AWS Config— Memberikan tampilan rinci tentang konfigurasi sumber AWS daya di Anda Akun AWS.	Menerima notifikasi saat sumber daya diperbarui, atau saat AWS Config mengevaluasi aturan kustom atau terkelola dengan sumber daya Anda. Untuk informasi selengkapnya, lihat Pemberitahuan yang AWS Config mengirim ke SNS topik dan item konfigurasi Contoh mengubah notifikasi di Panduan AWS Config Pengembang.
AWS Control Tower— Memungkinkan Anda mengatur dan mengatur lingkungan multi-akun yang aman, patuh, dan AWS multi-akun.	Gunakan pemberitahuan untuk membantu Anda mencegah drift dalam zona landasan, dan menerima notifikasi kepatuhan. Untuk informasi selengkapnya, lihat Pelacakan pemberitahuan melalui Amazon Simple Notification Service di Panduan Pengguna AWS Control Tower .

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
AWS License Manager— Membantu Anda mengelola lisensi perangkat lunak dari vendor perangkat lunak secara terpusat di seluruh AWS dan lingkungan lokal Anda.	Menerima notifikasi dan pemberitahuan License Manager. Untuk informasi selengkap nya, lihat Pengaturan di License Manager di Panduan Pengguna License Manager dan Membuat ServiceNow insiden untuk AWS License Manager pemberitahuan di Blog AWS Manajemen & Tata Kelola.
AWS Service Catalog – Memungkinkan administrator IT untuk membuat, mengelola , dan mendistribusikan portofolio produk yang disetujui kepada pengguna akhir, yang kemudian dapat mengakses produk yang mereka butuhkan di portal yang dipersonalisasi.	Menerima notifikasi tentang kejadian tumpukan. Untuk informasi selengkapnya, lihat batasan AWS Service Catalog pemberitahuan di Panduan Administrator Service Catalog.
AWS Systems Manager— Memungkinkan Anda melihat dan mengontrol infrastruktur Anda AWS.	Menerima notifikasi tentang status perintah. Untuk informasi selengkapnya, lihat Perubahan status Manajer Sistem Pemantauan menggunakan SNS notifikasi Amazon di Panduan AWS Systems Manager Pengguna.

Layanan media

Tabel berikut menjelaskan bagaimana Amazon SNS berintegrasi dengan Amazon Elastic Transcoder untuk mengirim pemberitahuan saat pekerjaan transcoding media mengubah status, memungkinkan Anda memantau dan mengelola konversi file media yang disimpan di Amazon S3 secara efisien ke dalam format yang sesuai untuk perangkat pemutaran konsumen.

Integrasi ini membantu Anda merampingkan alur kerja pemrosesan media dengan memberikan peringatan waktu nyata tentang status pekerjaan.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
Amazon Elastic Transcoder – Memungkin kan Anda mengkonversi file media yang	Menerima notifikasi saat status tugas berubah. Untuk informasi selengkapnya, lihat Notifikas

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
Anda simpan di Amazon S3 ke file media dalam format yang dibutuhkan oleh perangkat pemutaran konsumen.	i status tugas di Panduan Developer Amazon Elastic Transcoder.

Lyanan migrasi & transfer

Tabel berikut menjelaskan cara Amazon SNS berintegrasi dengan layanan AWS migrasi dan transfer, seperti AWS Application Discovery Service, AWS Database Migration Service (DMS), dan AWS Snowball, untuk memberikan pemberitahuan untuk peristiwa seperti pengumpulan data server, aktivitas migrasi database, dan pekerjaan transfer data.

Integrasi ini membantu Anda mengelola dan memantau proses migrasi Cloud secara efektif dengan menawarkan peringatan dan pembaruan waktu nyata pada tugas migrasi penting.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
AWS Application Discovery Service— Membantu Anda merencanakan migrasi ke AWS Cloud dengan mengumpulkan data penggunaan dan konfigurasi tentang server lokal Anda.	Terima pemberitahuan acara melalui AWS CloudTrail. Untuk informasi selengkapnya, lihat Mencatat API panggilan Application Discovery Service AWS CloudTrail di Panduan Pengguna Application Discovery Service.
AWS Database Migration Service— Memigrasi data dari database lokal ke file. AWS Cloud	Menerima pemberitahuan ketika AWS DMS peristiwa terjadi; misalnya, ketika instance replikasi dibuat atau dihapus. Untuk informasi selengkapnya, lihat Bekerja dengan kejadian dan notifikasi di AWS Database Migration Service di Panduan Pengguna AWS Database Migration Service .
AWS Snowball— Menggunakan perangkat penyimpanan fisik untuk mentransfer sejumlah besar data antara Amazon S3 dan lokasi penyimpanan data di tempat Anda dengan kecepatan tinggi. faster-than-internet	Menerima notifikasi untuk tugas Snowball. Untuk informasi selengkapnya, lihat Pemberitahuan untuk perangkat Keluarga Salju di Panduan AWS Snowcone Pengguna.

Layanan jaringan dan pengiriman konten

Tabel berikut menjelaskan bagaimana Amazon SNS berintegrasi dengan AWS jaringan dan layanan pengiriman konten, seperti Amazon API Gateway, Amazon, CloudFront AWS Direct Connect, Elastic Load Balancing, Amazon Route 53, dan VPC Amazon, untuk mengirim pemberitahuan untuk peristiwa API seperti pesan CloudFront , alarm metrik, perubahan status koneksi, peristiwa penyeimbang beban, status pemeriksaan kesehatan, dan aktivitas titik akhir. VPC

Integrasi ini membantu Anda memantau dan mengelola operasi pengiriman jaringan dan konten Anda dengan memberikan peringatan dan pembaruan tepat waktu.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
Amazon API Gateway - Memungkinkan Anda untuk membuat dan menyebarkan milik Anda sendiri REST dan dalam WebSocket APIs skala apa pun.	Menerima pesan yang diposting ke titik akhir API Gateway. Untuk informasi selengkapnya, lihat Tutorial: Membangun API Gateway REST API dengan AWS integrasi dalam Panduan Pengembang API Gateway.
Amazon CloudFront - Mempercepat distribusi konten web statis dan dinamis Anda, seperti.h tml, .css, .php, gambar, dan file media.	Menerima pemberitahuan saat alarm berdasark an CloudFront metrik yang ditentukan terjadi. Untuk informasi selengkapnya, lihat Menyetel alarm untuk menerima notifikasi di Panduan CloudFront Pengembang Amazon.
AWS Direct Connect— Menghubungkan jaringan internal Anda ke AWS Direct Connect lokasi melalui kabel serat optik Ethernet standar.	Menerima notifikasi saat status alarm yang memantau status koneksi AWS Direct Connect berubah. Untuk informasi selengkapnya, lihat Membuat CloudWatch alarm untuk memantau AWS Direct Connect koneksi di Panduan AWS Direct Connect Pengguna.
Elastic Load Balancing — Secara otomatis mendistribusikan lalu lintas masuk Anda di beberapa target, seperti EC2 instans Amazon, container, dan alamat IP, di lebih banyak atau lebih Availability Zone.	Menerima notifikasi alarm yang telah Anda buat untuk kejadian penyeimbang beban. Untuk informasi selengkapnya, lihat Membuat CloudWatch alarm untuk penyeimbang beban di Panduan Pengguna untuk Penyeimbang Beban Klasik.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
Amazon Route 53 - Menyediakan pendaftar an domain, DNS perutean, dan pemeriksaan kesehatan.	Menerima notifikasi ketika status pemeriksaan kondisi tidak sehat. Untuk informasi selengkap nya, lihat Untuk menerima SNS pemberitahuan Amazon saat status pemeriksaan kesehatan tidak sehat (konsol) di Panduan Pengembang Amazon Route 53.
Amazon Virtual Private Cloud (AmazonVP C) - Memungkinkan Anda meluncurkan AWS sumber daya ke jaringan virtual yang telah Anda tentukan.	Menerima notifikasi untuk kejadian tertentu yang terjadi pada titik akhir antarmuka. Untuk informasi selengkapnya, lihat Membuat dan mengelola notifikasi untuk layanan titik akhir di Panduan VPC Pengguna Amazon.

Layanan keamanan, identitas, dan kepatuhan

Tabel berikut menjelaskan cara Amazon SNS berintegrasi dengan layanan AWS keamanan, identitas, dan kepatuhan, seperti Amazon AWS Directory Service, Amazon Inspector GuardDuty, AWS Security Hub dan, untuk memberikan pemberitahuan tentang perubahan status direktori, temuan keamanan, peristiwa Inspector, dan pengumuman hub keamanan.

Integrasi ini membantu Anda mempertahankan praktik keamanan yang kuat dengan menawarkan peringatan dan pembaruan tepat waktu tentang peristiwa keamanan dan kepatuhan.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
AWS Directory Service— Menyediakan beberapa cara untuk menggunakan Microsoft Active Directory (AD) dengan yang lain Layanan AWS.	Terima pesan email atau teks (SMS) saat status direktori Anda berubah. Untuk informasi selengkapnya, lihat Mengkonfigurasi notifikas i status direktori di Panduan Administrasi AWS Directory Service .
Amazon GuardDuty - Menyediakan pemantaua n keamanan berkelanjutan untuk membantu mengidentifikasi aktivitas yang tidak terduga	Menerima notifikasi tentang jenis temuan yang baru dirilis, pembaruan untuk jenis temuan yang ada, dan perubahan fungsiona litas lainnya. Untuk informasi selengkapnya,

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
dan berpotensi tidak sah atau berbahaya di AWS lingkungan Anda.	lihat Berlangganan SNS topik GuardDuty pengumuman di GuardDuty Panduan Pengguna Amazon.
Amazon Inspector - Menguji aksesibilitas jaringan EC2 instans Amazon Anda dan status keamanan aplikasi Anda yang berjalan pada instans tersebut.	Menerima notifikasi untuk kejadian Amazon Inspector. Untuk informasi selengkapnya, lihat Menyiapkan SNS topik untuk notifikasi Amazon Inspector di Panduan Pengguna Amazon Inspector.
AWS Security Hub— Mengotomatiskan pemeriksaan AWS keamanan dan memusatkan peringatan keamanan.	Menerima pemberitahuan tentang AWS Security Hub pengumuman, termasuk pemberitahuan tentang AWS Security Hub kontrol atau standar yang telah ditambahk an, diedit, atau dihentikan. Untuk informasi selengkapnya, lihat Berlangganan AWS Security Hub pengumuman dengan Amazon. SNS

Layanan nirserver

Tabel berikut menjelaskan bagaimana Amazon SNS terintegrasi dengan layanan seperti Amazon DynamoDB EventBridge, Amazon, dan Lambda untuk mengirim pemberitahuan untuk peristiwa penting seperti pembaruan pemeliharaan, aliran data waktu nyata, dan output fungsi Lambda.

Integrasi ini membantu Anda memantau dan mengelola aplikasi secara efisien dengan menerima peringatan tepat waktu tentang operasi penting dan mengotomatiskan respons terhadap peristiwa ini.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
Amazon DynamoDB - Memberikan kinerja yang cepat dan dapat diprediksi dengan skalabilitas yang mulus dalam layanan basis data Tanpa terkelola sepenuhnya ini. SQL	Menerima notifikasi ketika kejadian pemelihar aan terjadi. Untuk informasi selengkapnya, lihat Menyesuaikan setelan DAX klaster di Panduan Pengembang Amazon DynamoDB.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
Amazon EventBridge — Mengirimkan aliran data real-time dari aplikasi Anda sendiri, aplikasi software-as-a-service (SaaS), Layanan AWS dan dan merutekan data tersebut ke target, termasuk Amazon. SNS EventBridge Dulunya bernama CloudWatch Events.	Menerima pemberitahuan EventBridge acara. Untuk informasi selengkapnya, lihat EventBridgeTarget Amazon di Panduan EventBridge Pengguna Amazon.
AWS Lambda – Memungkinkan Anda menjalankan kode tanpa penyediaan atau pengelolaan server.	Menerima data output fungsi dengan menetapkan SNS topik sebagai antrian huruf mati Lambda atau tujuan Lambda. Untuk informasi selengkapnya, lihat Invokasi asinkron di Panduan Developer AWS Lambda .

Layanan penyimpanan

Tabel berikut menjelaskan bagaimana Amazon SNS berintegrasi dengan layanan AWS penyimpanan seperti AWS Backup, Amazon Elastic File System (EFS), Amazon S3 Glacier, Amazon S3 AWS Snowball , dan untuk memberikan pemberitahuan untuk berbagai peristiwa seperti aktivitas pencadangan, alarm sistem file, pekerjaan pengambilan data, perubahan bucket, dan operasi transfer data.

Integrasi ini membantu Anda memantau dan mengelola solusi penyimpanan secara efisien dengan menerima peringatan tepat waktu tentang peristiwa penyimpanan penting.

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
AWS Backup— Membantu Anda memusatkan dan mengotomatiskan pencadangan data Layanan AWS di Cloud dan di tempat	Menerima pemberitahuan AWS Backup acara. Untuk informasi selengkapnya, lihat Menggunakan Amazon SNS untuk melacak AWS Backup peristiwa di Panduan AWS Backup Pengembang.
Amazon Elastic File System - Menyediakan penyimpanan file untuk EC2 instans Amazon Anda.	Terima pemberitahuan alarm yang Anda buat untuk EFS acara Amazon. Untuk informasi selengkapnya, lihat Alat pemantauan otomatisa

Layanan AWS	Manfaat menggunakan dengan Amazon SNS
	si dalam Panduan Pengguna Amazon Elastic File System.
Amazon S3 Glacier – Menyediakan penyimpanan untuk data yang jarang digunakan.	Setel konfigurasi notifikasi pada vault sehingga saat pekerjaan selesai, pesan dikirim ke topik. SNS Untuk informasi selengkapnya, lihat Mengonfigurasi notifikasi vault di Amazon S3 Glacier di Panduan Developer Amazon S3.
Amazon Simple Storage Service (Amazon S3) – Menyediakan penyimpanan objek.	Menerima notifikasi saat perubahan terjadi pada bucket Amazon S3 atau kejadian langka saat objek tidak bereplikasi ke Wilayah tujuan mereka. Untuk informasi selengkapnya, lihat Panduan: Mengonfigurasi bucket untuk notifikasi (SNStopik atau SQS antrian) dan Memantau kemajuan dengan metrik replikasi dan notifikasi peristiwa Amazon S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.
AWS Snowball— Menggunakan perangkat penyimpanan fisik untuk mentransfer sejumlah besar data antara Amazon S3 dan lokasi penyimpanan data di tempat Anda dengan kecepatan tinggi. faster-than-internet	Menerima notifikasi untuk tugas Snowball. Untuk informasi selengkapnya, lihat Pemberitahuan untuk perangkat Keluarga Salju di Panduan AWS Snowcone Pengguna.

Sumber kejadian tambahan

Tabel berikut menjelaskan bagaimana Amazon SNS dapat digunakan untuk menerima pemberitahuan tepat waktu tentang pembaruan fitur AWS harian dan perubahan rentang alamat AWS IP, memastikan bahwa Anda mendapat informasi tentang rilis AWS layanan terbaru, jenis instans, VPC titik akhir, dan perubahan alamat IP publik.

Integrasi ini membantu Anda tetap up-to-date dengan perubahan AWS infrastruktur dan mengelola sumber daya Anda secara efektif.

Sumber	Manfaat menggunakan dengan Amazon SNS
AWS Pembaruan Fitur Harian	Terima informasi terperinci tepat waktu tentang rilis dan pembaruan AWS melalui SNS topik Amazon. Rilis ini mencakup Wilayah AWS, Layanan AWS, VPC titik akhir Amazon, Layanan AWS terintegrasi dengan AWS Service Quotas, jenis instans EC2 Amazon, jenis instans SageMaker Amazon, jenis instans Amazon Nimble Studio, versi mesin database Amazon, dan versi RDS Amazon Apache Kafka. MSK Untuk informasi selengkapnya, lihat Berlangganan Pembaruan Fitur AWS Harian melalui Amazon SNS di Blog AWS Berit a.
AWS Rentang alamat IP	Terima pemberitahuan perubahan rentang AWS IP melalui SNS topik Amazon. Untuk informasi selengkapnya, lihat pemberitahuan rentang alamat AWS IP di Referensi Umum Amazon Web Services, dan Berlangganan Perubahan Alamat IP AWS Publik melalui Amazon SNS di Blog AWS Berita.

Untuk informasi selengkapnya tentang komputasi berbasis kejadian, lihat sumber berikut:

- Apa itu Arsitektur Event-Driven?
- Event-Driven Computing dengan Amazon SNS dan AWS Compute, Storage, Database, dan Layanan Jaringan di Compute Blog AWS
- Memperkaya Arsitektur Event Driven dengan AWS Event Fork Pipelines di Blog Compute AWS

Tujuan SNS acara Amazon

Halaman ini mencantumkan semua tujuan yang dapat menerima informasi tentang acara, dikelompokkan berdasarkan pesan application-to-application (A2A) dan pemberitahuan application-to-person (A2P).

> **Note**
>
> Amazon SNS memperkenalkan FIFOtopik pada Oktober 2020. Saat ini, sebagian besar AWS layanan mendukung penerimaan acara dari topik SNS standar saja. Amazon SQS mendukung penerimaan acara dari SNS standar dan FIFO topik.

Topik

Tujuan A2A

Tabel berikut menjelaskan bagaimana Amazon SNS dapat mengirimkan peristiwa ke berbagai tujuan application-to-application (A2A) seperti Amazon Data Firehose, Lambda, Amazon, Event Fork Pipelines, SQS dan AWS titik akhir /S. HTTP

Integrasi ini memungkinkan Anda untuk mengarsipkan dan menganalisis data, memicu logika bisnis khusus, memfasilitasi integrasi aplikasi, dan merutekan peristiwa ke webhook eksternal, meningkatkan efisiensi dan fleksibilitas arsitektur berbasis peristiwa.

Tujuan peristiwa	Manfaat menggunakan dengan Amazon SNS
Amazon Data Firehose	Mengirimkan kejadian ke aliran pengiriman untuk tujuan pengarsipan dan analisis. Melalui streaming pengiriman, Anda dapat mengirimk an acara ke AWS tujuan seperti Amazon Simple Storage Service (Amazon S3), Amazon Redshift, dan OpenSearch Amazon Service OpenSearch (Service), atau ke tujuan pihak

Tujuan peristiwa	Manfaat menggunakan dengan Amazon SNS
	ketiga seperti Datadog, New Relic, MongoDB, dan Splunk. Untuk informasi selengkapnya, lihat Aliran pengiriman Fanout ke Firehose.
AWS Lambda	Mengirimkan kejadian ke fungsi untuk memicu eksekusi logika bisnis kustom. Untuk informasi selengkapnya, lihat SNSPemberitahuan Amazon Fanout ke fungsi Lambda untuk pemrosesan otomatis.
Amazon SQS	Mengirimkan kejadian ke antrean untuk tujuan integrasi aplikasi. Untuk informasi selengkap nya, lihat SNSPemberitahuan Fanout Amazon ke SQS antrian Amazon untuk pemrosesan asinkron.
AWS Pipa Garpu Acara	Mengirimkan kejadian ke pencadangan dan penyimpanan kejadian, pencarian dan analitik kejadian, atau alur putar ulang kejadian. Untuk informasi selengkapnya, lihat Acara Fanout Amazon ke SNS AWS Event Fork Pipelines.
HTTP/S	Mengirimkan kejadian ke webhooks eksternal . Untuk informasi selengkapnya, lihat SNSPemberitahuan Fanout Amazon ke titik akhir HTTPS.

Tujuan A2P

Tabel berikut menjelaskan cara Amazon SNS mengirimkan notifikasi application-to-person (A2P) ke berbagai tujuan, termasuk ponsel melalui SMS dan notifikasi push asli, kotak masuk email, ruang obrolan Amazon Chime, saluran Slack, dan wawasan operasional untuk tim panggilan melalui. PagerDuty

Integrasi ini meningkatkan efisiensi komunikasi dan operasional dengan memungkinkan peringatan dan pembaruan real-time di berbagai platform dan saluran komunikasi.

Tujuan peristiwa	Manfaat menggunakan dengan Amazon SNS
SMS	Mengirimkan kejadian ke ponsel sebagai pesan teks. Untuk informasi selengkapnya, lihat Pesan teks seluler dengan Amazon SNS.
Email	Mengirimkan kejadian ke kotak masuk sebagai pesan email. Untuk informasi selengkapnya, lihat Pengaturan dan manajemen langganan SNS email Amazon.
Titik akhir platform	Mengirimkan kejadian ke ponsel sebagai notifikasi push asli. Untuk informasi selengkap nya, lihat Mengirim notifikasi push seluler dengan Amazon SNS.
AWS Chatbot	Mengirimkan kejadian ke ruang obrolan Amazon Chime atau saluran Slack. Untuk informasi lebih lanjut, lihat halaman berikut di Panduan Administrator AWS Chatbot : • Menyiapkan AWS Chatbot dengan Amazon Chime • Menyiapkan AWS Chatbot dengan Slack • Menggunakan AWS Chatbot dengan AWS layanan lain
PagerDuty	Mengirimkan wawasan operasional ke tim on-call. Untuk informasi selengkapnya, lihat Memberikan wawasan operasional yang didukung MP ke tim panggilan Anda melalui PagerDuty Amazon DevOps Guru di Blog AWS Manajemen & Tata Kelola.

Note

Anda dapat mengirimkan AWS acara asli dan acara khusus ke aplikasi obrolan:

- AWS Acara asli - Anda dapat menggunakan AWS Chatbot untuk mengirim AWS acara asli, melalui SNS topik Amazon, ke Amazon Chime dan Slack. Kumpulan AWS acara asli yang didukung mencakup acara dari AWS Billing and Cost Management,, AWS Health, Amazon AWS CloudFormation CloudWatch, dan banyak lagi. Untuk informasi selengkapnya, lihat Menggunakan AWS Chatbot dengan layanan lain di Panduan AWS Chatbot Administrator.
- Acara khusus - Anda juga dapat mengirim acara khusus, melalui SNS topik Amazon, ke Amazon Chime, Slack, dan Microsoft Teams. Untuk melakukan ini, Anda mempublikasikan acara khusus ke suatu SNS topik, yang mengirimkan acara ke fungsi Lambda berlangganan. Fungsi Lambda kemudian menggunakan webhook aplikasi obrolan untuk mengirimkan kejadian ke penerima. Untuk informasi selengkapnya, lihat Bagaimana cara menggunakan webhook untuk mempublikasikan SNS pesan Amazon ke Amazon Chime, Slack, atau Microsoft Teams?

Menggunakan Amazon SNS untuk application-to-application pengiriman pesan

Amazon SNS menyederhanakan pengiriman pesan application-to-application (A2A) dengan memisahkan penerbit dari pelanggan, yang mendukung layanan mikro, sistem terdistribusi, dan aplikasi tanpa server. Pesan dikirim ke SNS topik Amazon, di mana mereka dapat difilter dan dikirim ke pelanggan seperti LambdaSQS, Amazon, atau HTTP titik akhir. Jika pengiriman gagal, pesan disimpan dalam antrian huruf mati untuk analisis atau pemrosesan ulang lebih lanjut.

Untuk informasi tentang menggunakan Amazon SNS untuk application-to-application mengirim pesan dengan pelanggan, lihat berikut ini:

Topik

- Aliran pengiriman Fanout ke Firehose
- SNSPemberitahuan Amazon Fanout ke fungsi Lambda untuk pemrosesan otomatis
- SNSPemberitahuan Fanout Amazon ke SQS antrian Amazon untuk pemrosesan asinkron
- SNSPemberitahuan Fanout Amazon ke titik akhir HTTPS
- Acara Fanout Amazon ke SNS AWS Event Fork Pipelines
- Menggunakan Amazon EventBridge Scheduler dengan Amazon SNS

Aliran pengiriman Fanout ke Firehose

Anda dapat berlangganan aliran pengiriman Amazon Data Firehose ke SNS topik Amazon, yang memungkinkan Anda mengirim pemberitahuan ke titik akhir penyimpanan dan analitik tambahan. Pesan yang dipublikasikan ke SNS topik Amazon dikirim ke aliran pengiriman Firehose berlangganan, dan dikirim ke tujuan seperti yang dikonfigurasi di Firehose. Pemilik langganan dapat berlangganan hingga lima aliran pengiriman Firehose ke topik Amazon. SNS Setiap aliran pengiriman Firehose memiliki kuota default untuk permintaan dan throughput per detik. Batas ini dapat menghasilkan lebih banyak pesan yang diterbitkan (lalu lintas masuk) daripada yang dikirim (lalu lintas keluar). Ketika ada lebih banyak lalu lintas masuk daripada keluar, langganan Anda dapat mengumpulkan backlog pesan yang besar, menyebabkan latensi pengiriman pesan tinggi. Anda dapat meminta kenaikan kuota berdasarkan tarif publikasi untuk menghindari dampak buruk pada beban kerja Anda.

Melalui aliran pengiriman Firehose, Anda dapat menyebarkan notifikasi Amazon SNS ke Amazon Simple Storage Service (Amazon S3), Amazon Redshift, OpenSearch Amazon Service (Layanan), dan ke penyedia OpenSearch layanan pihak ketiga seperti Datadog, New Relic, MongoDB, dan Splunk.

Sebagai contoh, Anda dapat menggunakan fungsionalitas ini untuk menyimpan pesan secara permanen yang dikirim ke topik dalam bucket Amazon S3 untuk kepatuhan, arsip, atau tujuan lainnya. Untuk melakukannya, buat aliran pengiriman Firehose dengan tujuan bucket S3, dan berlangganan aliran pengiriman tersebut ke topik Amazon. SNS Sebagai contoh lain, untuk melakukan analisis pada pesan yang dikirim ke SNS topik Amazon, buat aliran pengiriman dengan tujuan indeks OpenSearch Layanan. Anda kemudian dapat berlangganan aliran pengiriman Firehose ke topik AmazonSNS.

Amazon SNS juga mendukung pencatatan status pengiriman pesan untuk pemberitahuan yang dikirim ke titik akhir Firehose. Untuk informasi selengkapnya, lihat Status pengiriman SNS pesan Amazon.

Topik

- Prasyarat untuk berlangganan aliran pengiriman Firehose ke topik Amazon SNS
- Berlangganan aliran pengiriman Firehose ke topik Amazon SNS
- Mengelola SNS pesan Amazon di beberapa tujuan aliran pengiriman
- Pengarsipan dan analitik SNS pesan Amazon: Contoh kasus penggunaan untuk platform tiket pesawat

Prasyarat untuk berlangganan aliran pengiriman Firehose ke topik Amazon SNS

Untuk berlangganan aliran pengiriman Amazon Data Firehose ke suatu SNS topik, Anda Akun AWS harus memiliki:

- SNSTopik standar. Untuk informasi selengkapnya, lihat Membuat SNS topik Amazon.
- Aliran pengiriman Firehose. Untuk informasi selengkapnya, lihat Membuat Aliran Pengiriman Firehose Data Amazon dan Berikan Akses Aplikasi Anda ke Sumber Daya Firehose Anda di Panduan Pengembang Amazon Data Firehose.
- Peran AWS Identity and Access Management (IAM) yang mempercayai prinsipal SNS layanan Amazon dan memiliki izin untuk menulis ke aliran pengiriman. Anda akan memasukkan Amazon Resource Name (ARN) peran ini sebagai `SubscriptionRoleARN` saat Anda membuat langganan. Amazon SNS mengasumsikan peran ini, yang memungkinkan Amazon SNS untuk menempatkan catatan dalam aliran pengiriman Firehose.

  Kebijakan contoh berikut ini menunjukkan izin yang direkomendasikan:

```
{
  "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"
    }
  ]
}
```

  Untuk memberikan izin penuh untuk menggunakan Firehose, Anda juga dapat menggunakan kebijakan AWS terkelola. `AmazonKinesisFirehoseFullAccess` Atau, untuk memberikan izin

yang lebih ketat untuk menggunakan Firehose, Anda dapat membuat kebijakan sendiri. Minimal, kebijakan harus memberikan izin untuk menjalankan operasi `PutRecord` pada aliran pengiriman spesifik.

Dalam semua kasus, Anda juga harus mengedit hubungan kepercayaan untuk menyertakan prinsipal SNS layanan Amazon. Sebagai contoh:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "sns.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Untuk informasi selengkapnya tentang membuat peran, lihat Membuat peran untuk mendelegasikan izin ke AWS layanan di IAMPanduan Pengguna.

Setelah Anda menyelesaikan persyaratan ini, Anda dapat berlangganan aliran pengiriman ke SNS topik.

Berlangganan aliran pengiriman Firehose ke topik Amazon SNS

Untuk mengirimkan SNS notifikasi Amazon ke aliran pengiriman Amazon Data Firehose, pertama-tama pastikan bahwa Anda telah menangani semua prasyarat. Untuk daftar titik akhir yang didukung, lihat titik akhir Amazon Data Firehose dan kuota di. Referensi Umum Amazon Web Services

Untuk berlangganan aliran pengiriman Firehose ke suatu topik

1. Masuk ke SNSkonsol Amazon.
2. Di panel navigasi, pilih Subscriptions (Langganan).
3. Pada halaman Berlangganan, pilih Buat berlangganan.
4. Di halaman Create subscription (Buat langganan), di bagian Details (Detail), lakukan:

a. Untuk Topik ARN, pilih Amazon Resource Name (ARN) dari topik standar.

b. Untuk Protokol, pilih Firehose.

c. Untuk Endpoint, pilih aliran pengiriman Firehose yang dapat menerima notifikasi dari Amazon. ARN SNS

d. Untuk peran Langganan ARN, tentukan ARN peran AWS Identity and Access Management (IAM) yang Anda buat untuk menulis ke aliran pengiriman Firehose. Untuk informasi selengkapnya, lihat Prasyarat untuk berlangganan aliran pengiriman Firehose ke topik Amazon SNS.

e. (Opsional) Untuk menghapus SNS metadata Amazon dari pesan yang dipublikasikan, pilih Aktifkan pengiriman pesan mentah. Untuk informasi selengkapnya, lihat Pengiriman pesan SNS mentah Amazon.

5. (Opsional) Untuk mengkonfigurasi kebijakan filter, perluas bagian Subscription filter policy (Kebijakan filter langganan). Untuk informasi selengkapnya, lihat Kebijakan filter SNS langganan Amazon.
6. (Opsional) Untuk mengonfigurasi antrean surat mati untuk berlanggana, perluas bagian Redrive policy (dead-letter queue) (Kebijakan redrive (antrean surat mati)). Untuk informasi selengkapnya, lihat Antrian SNS surat mati Amazon.
7. Pilih Create subscription (Buat langganan).

Konsol tersebut membuat langganan dan membuka halaman Details (Detail) langganan.

Mengelola SNS pesan Amazon di beberapa tujuan aliran pengiriman

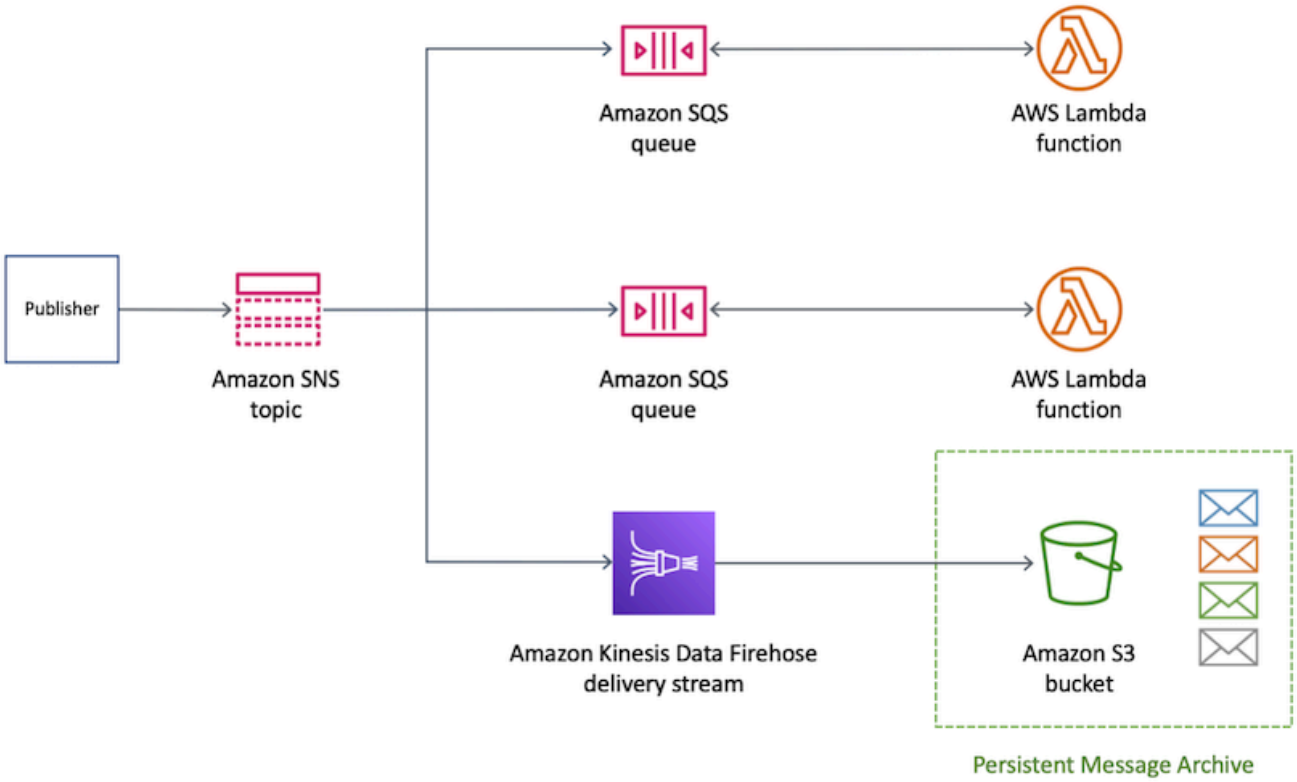
Melalui aliran pengiriman Amazon Data Firehose, Anda dapat mengirim pesan ke titik akhir tambahan. Bagian ini menjelaskan cara untuk bekerja dengan tujuan yang didukung.

Topik

- Menyimpan dan menganalisis SNS pesan Amazon di tujuan Amazon S3
- Mengintegrasikan SNS pesan Amazon dengan tujuan OpenSearch Layanan Amazon
- Mengonfigurasi pengiriman dan analisis SNS pesan Amazon di tujuan Amazon Redshift
- Mengonfigurasi pengiriman SNS pesan Amazon ke HTTP tujuan menggunakan Amazon Data Firehose

Menyimpan dan menganalisis SNS pesan Amazon di tujuan Amazon S3

Bagian ini memberikan informasi tentang aliran pengiriman Amazon Data Firehose yang mempublikasikan data ke Amazon Simple Storage Service (Amazon S3).



Topik

- Memformat SNS notifikasi Amazon untuk penyimpanan di tujuan Amazon S3
- Menganalisis SNS pesan Amazon yang disimpan di Amazon S3 menggunakan Athena

Memformat SNS notifikasi Amazon untuk penyimpanan di tujuan Amazon S3

Contoh berikut menunjukkan SNS notifikasi Amazon yang dikirim ke bucket Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3), menggunakan indentasi agar mudah dibaca.

> **Note**
>
> Dalam contoh ini, pengiriman pesan mentah dinonaktifkan untuk pesan yang diterbitkan. Saat pengiriman pesan mentah dinonaktifkan, Amazon SNS menambahkan JSON metadata ke pesan, termasuk properti ini:

- Type
- MessageId
- TopicArn
- Subject
- Timestamp
- UnsubscribeURL
- MessageAttributes

Untuk informasi selengkapnya tentang pengiriman mentah, lihat Pengiriman pesan SNS mentah Amazon.

```
{
    "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"
        }
    }
 }
```

Contoh berikut menunjukkan tiga SNS pesan yang dikirim melalui aliran pengiriman Amazon Data Firehose ke bucket Amazon S3 yang sama. Buffering diperhitungkan, dan jeda baris memisahkan pesan.

```
{"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"}
```


Menganalisis SNS pesan Amazon yang disimpan di Amazon S3 menggunakan Athena

Halaman ini menjelaskan cara menganalisis SNS pesan Amazon yang dikirim melalui aliran pengiriman Amazon Data Firehose ke tujuan Amazon Simple Storage Service (Amazon S3).

Untuk menganalisis SNS pesan yang dikirim melalui aliran pengiriman Firehose ke tujuan Amazon S3

1. Konfigurasi sumber daya Amazon S3 Anda. Untuk petunjuknya, lihat Membuat bucket di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon dan Bekerja dengan Bucket Amazon S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.
2. Konfigurasi aliran pengiriman Anda. Untuk petunjuknya, lihat Memilih Amazon S3 untuk Tujuan Anda di Panduan Pengembang Amazon Data Firehose.
3. Gunakan Amazon Athena untuk menanyakan objek Amazon S3 menggunakan standar. SQL Untuk informasi selanjutnya, lihat Memulai dalam Panduan Pengguna Amazon Athena.

Kueri contoh

Untuk kueri contoh ini, asumsikan berikut ini:

- Pesan disimpan dalam tabel `notifications` di skema `default`.

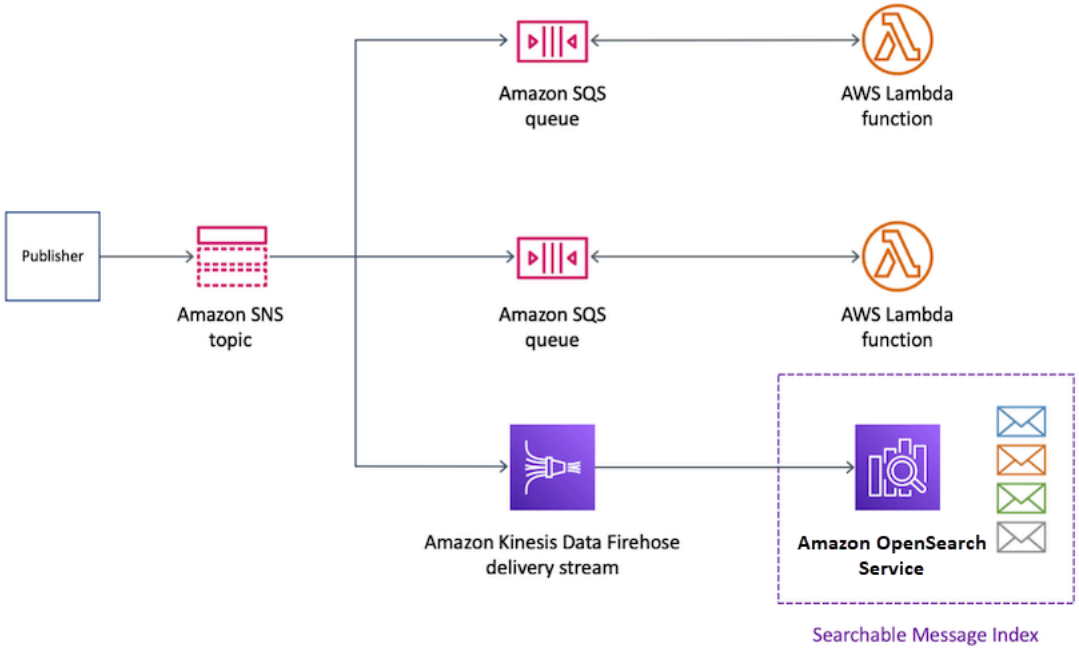
- Tabel `notifications` mencakup kolom `timestamp` dengan jenis `string`.

Kueri berikut mengembalikan semua SNS pesan yang diterima dalam rentang tanggal yang ditentukan:

```
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';
```

Mengintegrasikan SNS pesan Amazon dengan tujuan OpenSearch Layanan Amazon

Bagian ini memberikan informasi tentang aliran pengiriman Amazon Data Firehose yang mempublikasikan data ke OpenSearch Layanan Amazon (Layanan)OpenSearch .



Topik

- Menyimpan dan memformat SNS Pemberitahuan Amazon dalam indeks OpenSearch Layanan
- Menganalisis SNS pesan Amazon untuk tujuan OpenSearch Layanan

Menyimpan dan memformat SNS Pemberitahuan Amazon dalam indeks OpenSearch Layanan

Contoh berikut menunjukkan SNS notifikasi Amazon yang dikirim ke indeks OpenSearch Layanan Amazon (OpenSearch Layanan) bernama`my-index`. Indeks ini memiliki bidang filter waktu pada bidang `Timestamp`. SNSPemberitahuan ditempatkan di `_source` properti muatan.

> **Note**
>
> Dalam contoh ini, pengiriman pesan mentah dinonaktifkan untuk pesan yang diterbitkan. Saat pengiriman pesan mentah dinonaktifkan, Amazon SNS menambahkan JSON metadata ke pesan, termasuk properti ini:
>
> - `Type`
> - `MessageId`
> - `TopicArn`
> - `Subject`
> - `Timestamp`
> - `UnsubscribeURL`
> - `MessageAttributes`
>
> Untuk informasi selengkapnya tentang pengiriman mentah, lihat Pengiriman pesan SNS mentah Amazon.


```
{
  "_index": "my-index",
  "_type": "_doc",
  "_id": "49613100963111323203250405402193283794773886550985932802.0",
  "_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
  ]
}
```

Menganalisis SNS pesan Amazon untuk tujuan OpenSearch Layanan

Halaman ini menjelaskan cara menganalisis SNS pesan Amazon yang dikirim melalui aliran pengiriman Amazon Data Firehose ke tujuan OpenSearch Layanan Amazon (OpenSearch Layanan).

Untuk menganalisis SNS pesan yang dikirim melalui aliran pengiriman Firehose ke tujuan Layanan OpenSearch

1. Konfigurasikan sumber daya OpenSearch Layanan Anda. Untuk petunjuk, lihat Memulai OpenSearch Layanan Amazon di Panduan Pengembang OpenSearch Layanan Amazon.
2. Konfigurasi aliran pengiriman Anda. Untuk petunjuknya, lihat Memilih OpenSearch Layanan untuk Tujuan Anda di Panduan Pengembang Amazon Data Firehose.
3. Jalankan kueri menggunakan kueri OpenSearch Layanan dan Kibana. Untuk informasi selengkapnya, lihat Langkah 3: Cari Dokumen di Domain OpenSearch Layanan dan Kibana di Panduan Pengembang OpenSearch Layanan Amazon.

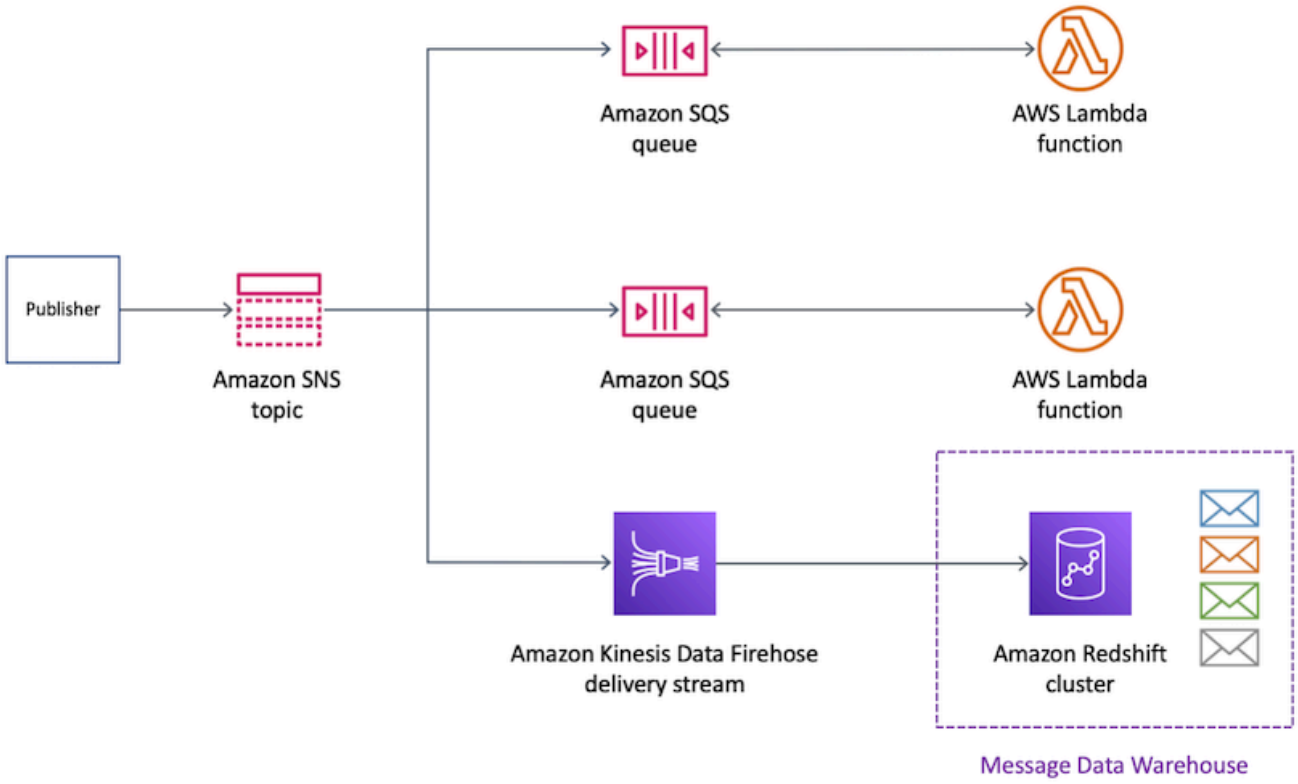
Kueri contoh

Contoh berikut menanyakan `my-index` indeks untuk semua SNS pesan yang diterima dalam rentang tanggal yang ditentukan:

```
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"
            }
          }
        }
      ]
    }
  }
}
```

Mengonfigurasi pengiriman dan analisis SNS pesan Amazon di tujuan Amazon Redshift

Bagian ini menjelaskan cara menyebarkan SNS notifikasi Amazon ke aliran pengiriman Amazon Data Firehose yang menerbitkan data ke Amazon Redshift. Dengan konfigurasi ini, Anda dapat terhubung ke database Amazon Redshift dan menggunakan alat SQL kueri untuk menanyakan database untuk SNS pesan Amazon yang memenuhi kriteria tertentu.



Topik

- Menata arsip SNS pesan Amazon di tabel Amazon Redshift
- Menganalisis SNS pesan Amazon yang disimpan di tujuan Amazon Redshift

Menata arsip SNS pesan Amazon di tabel Amazon Redshift

Untuk titik akhir Amazon Redshift, SNS pesan Amazon yang diterbitkan diarsipkan sebagai baris dalam tabel. Berikut adalah contohnya.

> **Note**
>
> Dalam contoh ini, pengiriman pesan mentah dinonaktifkan untuk pesan yang diterbitkan. Saat pengiriman pesan mentah dinonaktifkan, Amazon SNS menambahkan JSON metadata ke pesan, termasuk properti ini:
>
> - `Type`
> - `MessageId`
> - `TopicArn`

- `Subject`
- `Message`
- `Timestamp`
- `UnsubscribeURL`
- `MessageAttributes`

Untuk informasi selengkapnya tentang pengiriman mentah, lihat Pengiriman pesan SNS mentah Amazon.
Meskipun Amazon SNS menambahkan properti ke pesan menggunakan huruf besar yang ditampilkan dalam daftar ini, nama kolom di tabel Amazon Redshift muncul di semua karakter huruf kecil. Untuk mengubah JSON metadata untuk titik akhir Amazon Redshift, Anda dapat menggunakan perintah. SQL `COPY` Untuk informasi selengkapnya, lihat Menyalin dari JSON contoh dan Memuat dari JSON data menggunakan opsi 'auto ignorecase' di Panduan Pengembang Database Amazon Redshift.

jenis	messageid	topicarn	subjek	pesan	timestamp	unsubscribeurl	messageattributes
Notifikasi	ea544832-a0d8-581d-9275-108243c46103	arn:aws:sns:us-east-1:111111111111:my-topic	Subjek sampel	Pesan sampel	2020-12-02T00:33:32.272Z	https://sns.us-east-1.amazonaws.com/?Action=Berhenti berlangganan &=arn:aws:sns:us-east-1:111111111111:Subscript	{\"my_attribute\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

jenis	messageid	topicarn	subjek	pesan	timestamp	unsubscribeurl	messageattributes
						ionArn my-topik: 326deeeb-cbf4-45da-b92b-ca7 7a247813b	
Notifikasi	ab124832-a0d8-581d-9275-108 243c46114	arn:aws:sns:us-eas t-1:11111 1111111:my-topic	Subjek sampel 2	Pesan sampel 2	2020-12-03T00:18:11.129Z	https:// sns.us-eas t-1.amazonaws.com/ ? Action=Berhenti berlanggananan & =arn:aws: sns:us-east-1:1111111111: SubscriptionArn my-topik: 326deeeb-cbf4-45da-b92b-ca7 7a247813b	{\"my_attribute2\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

jenis	messageid	topicarn	subjek	pesan	timestamp	unsubscribeurl	messageattributes
Notifikasi	ce644832-a0d8-581d-9275-108243c46125	arn:aws:sns:us-east-1:111111111111:my-topic	Subjek sampel 3	Pesan sampel 3	2020-12-09T00:08:44.405Z	https://sns.us-east-1.amazonaws.com/?Action=Berhentiberlanggananan &=arn:aws:sns:us-east-1:111111111111:SubscriptionArn my-topik:326deeeb-cbf4-45da-b92b-ca77a247813b	{\"my_attribute3\":{\"Type\":\"String\",\"Value\":\"my_value\"}}

Untuk informasi selengkapnya tentang fan out notifikasi ke titik akhir Amazon Redshift, lihat Mengonfigurasi pengiriman dan analisis SNS pesan Amazon di tujuan Amazon Redshift.

Menganalisis SNS pesan Amazon yang disimpan di tujuan Amazon Redshift

Halaman ini menjelaskan cara menganalisis SNS pesan Amazon yang dikirim melalui aliran pengiriman Amazon Data Firehose ke tujuan Amazon Redshift.

Untuk menganalisis SNS pesan yang dikirim melalui aliran pengiriman Firehose ke tujuan Amazon Redshift

1. Konfigurasi sumber daya Amazon Redshift. Untuk instruksi, lihat Memulai dengan Amazon Redshift di Panduan Memulai Amazon Redshift.
2. Konfigurasi aliran pengiriman Anda. Untuk petunjuknya, lihat Memilih Amazon Redshift untuk Tujuan Anda di Panduan Pengembang Amazon Data Firehose.
3. Jalankan kueri. Untuk informasi selengkapnya, lihat Menanyakan database menggunakan editor kueri di Panduan Manajemen Amazon Redshift.

Kueri contoh

Untuk kueri contoh ini, asumsikan berikut ini:

- Pesan disimpan dalam tabel `notifications` di skema `public` default.
- `Timestamp`Properti dari SNS pesan disimpan dalam `timestamp` kolom tabel dengan tipe data kolom`timestamptz`.

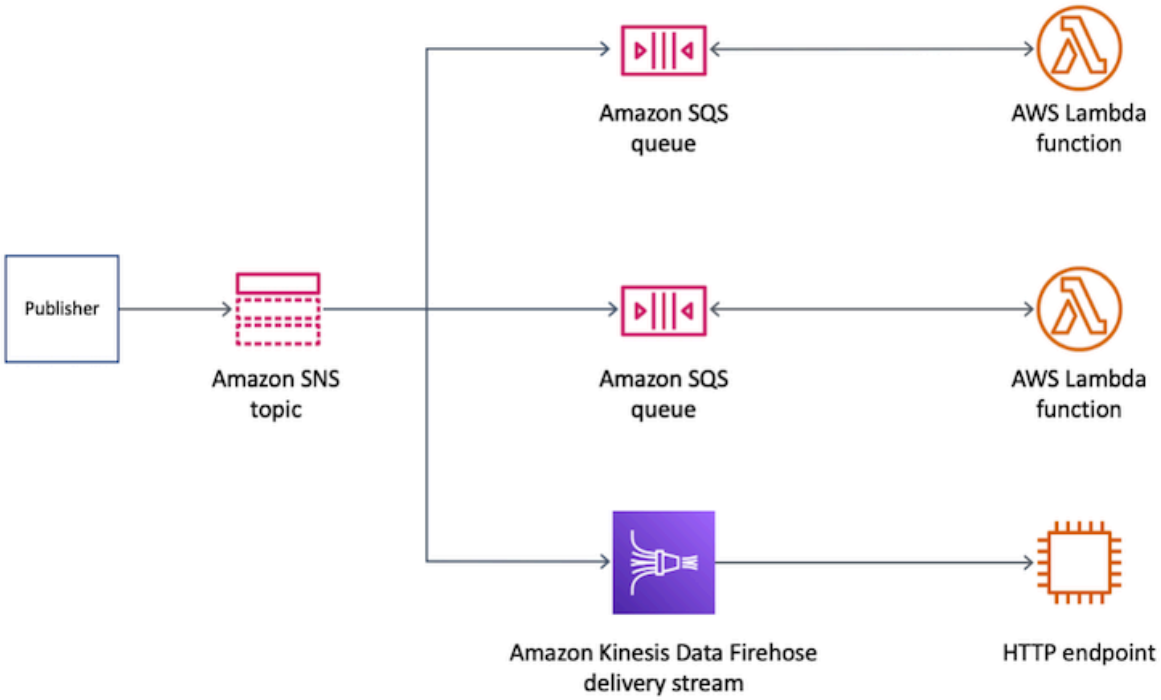
> **Note**
>
> Untuk mengubah JSON metadata untuk titik akhir Amazon Redshift, Anda dapat menggunakan perintah. SQL `COPY` Untuk informasi selengkapnya, lihat Menyalin dari JSON contoh dan Memuat dari JSON data menggunakan opsi 'auto ignorecase' di Panduan Pengembang Database Amazon Redshift.

Kueri berikut mengembalikan semua SNS pesan yang diterima dalam rentang tanggal yang ditentukan:

```
SELECT *
FROM public.notifications
WHERE timestamp > '2020-12-01T09:00:00.000Z' AND timestamp <
 '2020-12-02T09:00:00.000Z';
```

Mengonfigurasi pengiriman SNS pesan Amazon ke HTTP tujuan menggunakan Amazon Data Firehose

Bagian ini memberikan informasi tentang aliran pengiriman Amazon Data Firehose yang mempublikasikan data ke titik akhir. HTTP



Topik

- Format SNS notifikasi Amazon untuk pengiriman ke HTTP tujuan

Format SNS notifikasi Amazon untuk pengiriman ke HTTP tujuan

Berikut ini adalah contoh badan HTTP POST permintaan dari Amazon SNS yang dapat dikirim oleh aliran pengiriman Amazon Data Firehose ke titik akhirHTTP. SNSPemberitahuan dikodekan sebagai payload base64 di properti. `records`

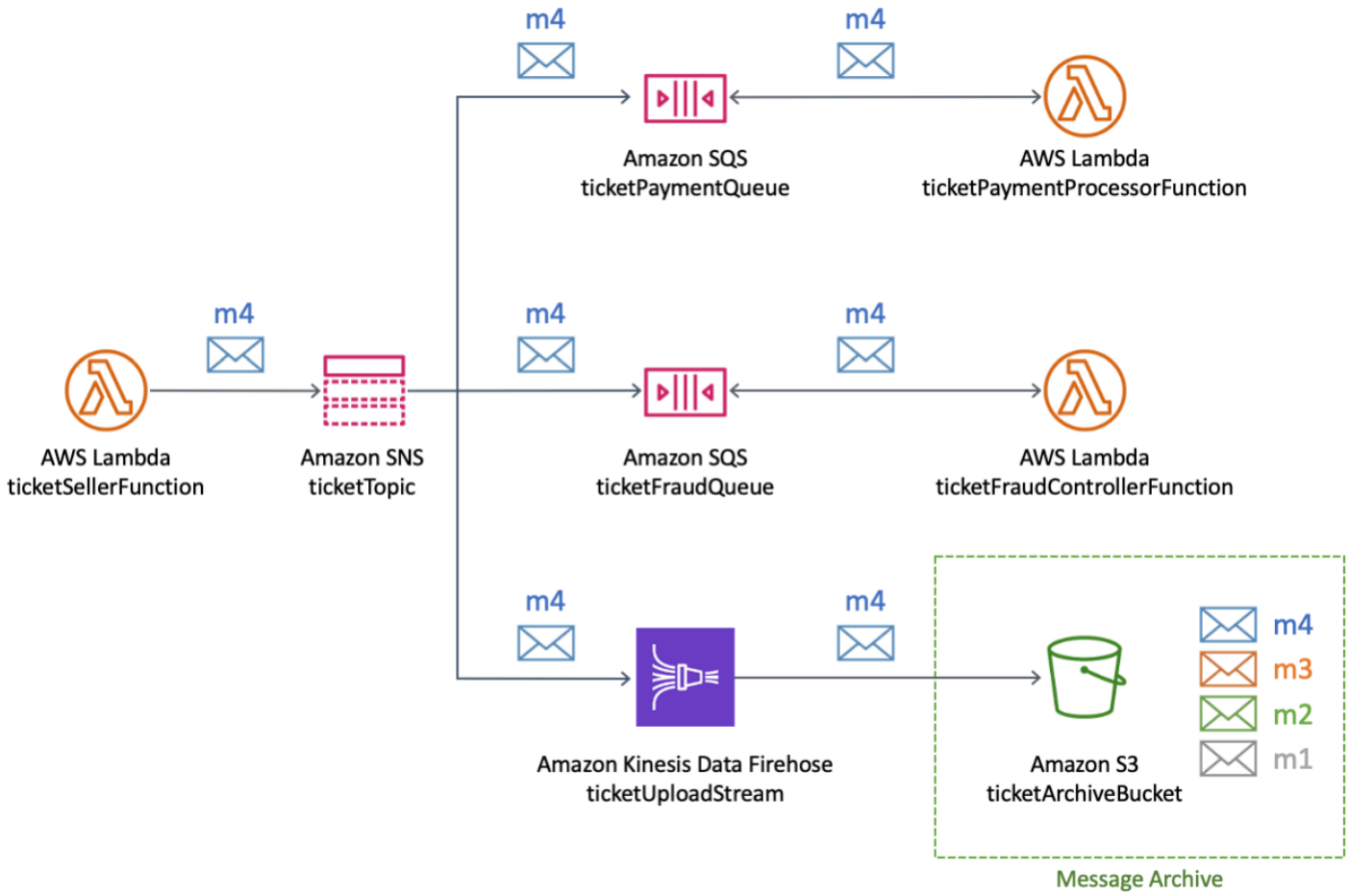
> **Note**
>
> Dalam contoh ini, pengiriman pesan mentah dinonaktifkan untuk pesan yang diterbitkan. Untuk informasi selengkapnya tentang pengiriman mentah, lihat Pengiriman pesan SNS mentah Amazon.


```
"body": {
    "requestId": "ebc9e8b2-fce3-4aef-a8f1-71698bf8175f",
    "timestamp": 1606255960435,
    "records": [
      {
        "data":
 "eyJUeXBlIjoiTm90aWZpY2F0aW9uIiwiTWVzc2FnZUlkIjoiMjFkMmUzOGQtMmNhYi01ZjYxLTliYTItYmJiYWFhYzg0M
      }
    ]
  }
```


Pengarsipan dan analitik SNS pesan Amazon: Contoh kasus penggunaan untuk platform tiket pesawat

Bagian ini menyediakan tutorial kasus penggunaan umum untuk pengarsipan dan analisis SNS pesan Amazon.

Pengaturan kasus penggunaan ini adalah platform tiket maskapai penerbangan yang beroperasi di lingkungan yang diatur. Platform ini tunduk pada kerangka kerja kepatuhan yang mengharuskan perusahaan untuk mengarsipkan semua penjualan tiket setidaknya selama lima tahun. Untuk memenuhi tujuan kepatuhan pada retensi data, perusahaan berlangganan aliran pengiriman Amazon Data Firehose ke topik yang adaSNS. Tujuan untuk aliran pengiriman adalah bucket Amazon Simple Storage Service (Amazon S3). Dengan konfigurasi ini, semua acara yang dipublikasikan ke SNS topik diarsipkan di bucket Amazon S3. Diagram berikut ini menunjukkan arsitektur konfigurasi ini:



Untuk menjalankan analitik dan mendapatkan wawasan tentang penjualan tiket, perusahaan menjalankan SQL kueri menggunakan Amazon Athena. Sebagai contoh, perusahaan dapat membuat kueri untuk mempelajari tentang tujuan yang paling populer dan selebaran yang paling sering muncul.

Untuk membuat AWS sumber daya untuk kasus penggunaan ini, Anda dapat menggunakan AWS Management Console atau AWS CloudFormation templat.

Topik

- Menyiapkan AWS sumber daya awal untuk pengarsipan dan analitik SNS pesan Amazon
- Mengatur aliran pengiriman Firehose untuk pengarsipan pesan Amazon SNS
- Berlangganan aliran pengiriman Firehose ke topik Amazon SNS
- Menguji dan menanyakan SNS konfigurasi Amazon untuk pengelolaan data yang efektif
- Mengotomatiskan pengarsipan SNS pesan Amazon dengan templat AWS CloudFormation

Menyiapkan AWS sumber daya awal untuk pengarsipan dan analitik SNS pesan Amazon

Halaman ini menjelaskan cara untuk membuat sumber daya berikut ini untuk pengarsipan pesan dan kasus penggunaan contoh analitik:

- Bucket Amazon Simple Storage Service (Amazon S3)
- Dua antrian Layanan Antrian Sederhana Amazon (AmazonSQS)
- SNSTopik Amazon
- Dua SQS langganan Amazon ke topik Amazon SNS

Untuk membuat sumber daya awal

1. Buat bucket Amazon S3:

   a. Buka konsol Amazon S3.

   b. Pilih Buat bucket.

   c. Untuk Nama bucket, masukkan nama yang unik secara global. Simpan bidang lainnya sebagai default.

   d. Pilih Buat bucket.

   Untuk informasi selengkapnya tentang bucket Amazon S3, lihat Membuat bucket di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon dan Bekerja dengan Bucket Amazon S3 di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.

2. Buat dua SQS antrian Amazon:

   a. Buka SQSkonsol Amazon.

   b. Pilih Buat antrean.

   c. Untuk Jenis, pilih Standar.

   d. Untuk Nama, masukkan **`ticketPaymentQueue`**.

   e. Di bawah Kebijakan akses, untuk Pilih metode, pilih Lanjutan.

   f. Di kotak JSON kebijakan, tempel kebijakan berikut:

```
{
  "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"
        }
      }
    }
  ]
}
```

Dalam kebijakan akses ini, ganti Akun AWS nomor (*123456789012*) dengan milik Anda sendiri, dan ubah AWS Wilayah (*us-east-1*) sesuai.

g. Pilih Buat antrean.

h. Ulangi langkah-langkah ini untuk membuat SQS antrean kedua bernama**ticketFraudQueue**.

Untuk informasi selengkapnya tentang membuat SQS antrian, lihat Membuat SQS antrian Amazon (konsol) di Panduan Pengembang Layanan Antrian Sederhana Amazon.

3. Buat SNS topik:

a. Buka halaman Topik SNS konsol Amazon.

b. Pilih Buat topik.

c. Di bawah Detail, untuk Jenis, pilih Standar.

d. Untuk Nama, masukkan **ticketTopic**.

e. Pilih Buat topik.

Untuk informasi selengkapnya tentang membuat SNS topik, lihatMembuat SNS topik Amazon.

4. Berlangganan kedua SQS antrian ke topik: SNS

a. Di SNSkonsol Amazon, pada halaman detail ticketTopictopik, pilih Buat langganan.

b. Di bawah Detail, untuk Protokol, pilih Amazon SQS.

c. Untuk Endpoint, pilih Amazon Resource Name (ARN) dari ticketPaymentQueueantrian.

d. Pilih Buat langganan.

e. Ulangi langkah-langkah ini untuk membuat langganan kedua ARN menggunakan ticketFraudQueueantrian.

Untuk informasi selengkapnya tentang berlangganan SNS topik, lihatMembuat langganan ke SNS topik Amazon. Anda juga dapat berlangganan SQS antrian ke SNS topik dari konsol AmazonSQS. Untuk informasi selengkapnya, lihat Berlangganan SQS antrian Amazon ke SNS topik Amazon (konsol) di Panduan Pengembang Layanan Antrian Sederhana Amazon.

Anda telah membuat sumber daya awal untuk kasus penggunaan contoh ini. Untuk melanjutkan, lihat Mengatur aliran pengiriman Firehose untuk pengarsipan pesan Amazon SNS.

Mengatur aliran pengiriman Firehose untuk pengarsipan pesan Amazon SNS

Halaman ini menjelaskan cara membuat aliran pengiriman Amazon Data Firehose untuk kasus penggunaan contoh pengarsipan pesan dan analisis.

Untuk membuat aliran pengiriman Firehose

1. Buka konsol layanan Amazon Kinesis.
2. Pilih Firehose lalu pilih Buat aliran pengiriman.
3. Pada halaman Aliran pengiriman baru, untuk Nama alitran pengiriman, masukkan **`ticketUploadStream`**, dan kemudian pilih Selanjutnya.
4. Pada halaman Proses rekaman, pilih Selanjutnya.
5. Pada halaman Pilih tujuan, lakukan hal berikut ini:

   a. Untuk Tujuan, pilih Amazon S3.

   b. Di bawah Tujuan S3, untuk Bucket S3, pilih bucket S3 yang Anda buat awalnya.

   c. Pilih Selanjutnya.
6. Pada halaman Konfigurasi pengaturan, untuk syarat buffer S3, lakukan hal berikut ini:

   - Untuk Ukuran buffer, masukkan **`1`**.
   - Untuk Interval buffer, masukkan **`60`**.

Dengan menggunakan nilai tersebut untuk buffer Amazon S3 memungkinkan Anda dengan cepat menguji konfigurasi. Syarat pertama yang dipenuhi memicu pengiriman data ke bucket S3.

7. Pada halaman Konfigurasi pengaturan, untuk Izin, pilih untuk membuat peran AWS Identity and Access Management (IAM) dengan izin yang diperlukan ditetapkan secara otomatis. Lalu, pilih Selanjutnya.
8. Pada halaman Tinjau, pilih Buat aliran pengiriman.
9. Dari halaman aliran pengiriman Kinesis Data Firehose, pilih aliran pengiriman yang baru saja Anda buat (). ticketUploadStream Pada tab Detail, perhatikan Amazon Resource Name (ARN) stream untuk nanti.

Untuk informasi selengkapnya tentang cara membuat aliran pengiriman, lihat Membuat Aliran Pengiriman Firehose Data Amazon di Panduan Pengembang Amazon Data Firehose. Untuk informasi selengkapnya tentang membuat IAM peran, lihat Membuat peran untuk mendelegasikan izin ke AWS layanan di IAMPanduan Pengguna.

Anda telah membuat aliran pengiriman Firehose dengan izin yang diperlukan. Untuk melanjutkan, lihat Berlangganan aliran pengiriman Firehose ke topik Amazon SNS.

Berlangganan aliran pengiriman Firehose ke topik Amazon SNS

Halaman ini menjelaskan cara untuk membuat berikut ini untuk pengarsipan pesan dan kasus penggunaan contoh analitik:

- Peran AWS Identity and Access Management (IAM) yang memungkinkan SNS langganan Amazon untuk menaruh catatan di aliran pengiriman Amazon Data Firehose
- Aliran pengiriman Firehose berlangganan topik SNS

Untuk membuat IAM peran untuk SNS langganan Amazon

1. Buka halaman Peran IAM konsol.
2. Pilih Buat peran.
3. Untuk Pilih tipe entitas tepercaya, pilih Layanan AWS .
4. Untuk Pilih kasus penggunaan, pilih SNS. Kemudian pilih Selanjutnya: Izin.
5. Pilih Selanjutnya: Tag.
6. Pilih Selanjutnya: Tinjau.

7. Pada halaman Tinjau, untuk Nama peran, masukkan **`ticketUploadStreamSubscriptionRole`**. Kemudian pilih Buat peran.
8. Saat peran dibuat, pilih namanya (ticketUploadStreamSubscriptionRole).
9. Pada halaman Ringkasan peran, pilih Tambahkan kebijakan inline.
10. Pada halaman Buat kebijakan, pilih JSONtab, lalu tempelkan kebijakan berikut ke dalam kotak:

```
{
    "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"
        }
    ]
}
```

    Dalam kebijakan ini, ganti Akun AWS nomor (*123456789012*) dengan milik Anda sendiri, dan ubah AWS Wilayah (*us-east-1*) sesuai.
11. Pilih Tinjau kebijakan.
12. Pada halaman Tinjau kebijakan, untuk Nama, masukkan **`FirehoseSnsPolicy`**. Kemudian pilih Buat kebijakan.
13. Pada halaman Ringkasan peran, perhatikan Peran ARN untuk nanti.

Untuk informasi selengkapnya tentang membuat IAM peran, lihat Membuat peran untuk mendelegasikan izin ke AWS layanan di IAMPanduan Pengguna.

Untuk berlangganan aliran pengiriman Firehose ke topik SNS

1. Buka halaman Topik SNS konsol Amazon.

2. Pada tab Berlangganan, pilih Buat berlangganan.
3. Di bawah Detail, untuk Protokol, pilih Amazon Data Firehose.
4. Untuk Endpoint, masukkan Amazon Resource Name (ARN) dari aliran ticketUploadStreampengiriman yang Anda buat sebelumnya. Misalnya, masukkan **`arn:aws:firehose:us-east-1:123456789012:deliverystream/ticketUploadStream`**.
5. Untuk peran Langganan ARN, masukkan ticketUploadStreamSubscriptionRoleIAMperan yang Anda buat sebelumnya. ARN Sebagai contoh, masukkan **`arn:aws:iam::123456789012:role/ticketUploadStreamSubscriptionRole`**.
6. Pilih kotak centang Aktifkan pengiriman pesan mentah.
7. Pilih Buat langganan.

Anda telah membuat langganan IAM peran dan SNS topik. Untuk melanjutkan, lihat Menguji dan menanyakan SNS konfigurasi Amazon untuk pengelolaan data yang efektif.

Menguji dan menanyakan SNS konfigurasi Amazon untuk pengelolaan data yang efektif

Halaman ini menjelaskan cara menguji kasus penggunaan contoh pengarsipan pesan dan analisis dengan menerbitkan pesan ke SNS topik Amazon. Instruksi termasuk kueri contoh yang dapat Anda jalankan dan menyesuaikan dengan kebutuhan Anda sendiri.

Untuk menguji konfigurasi Anda

1. Buka halaman Topik SNS konsol Amazon.
2. Pilih topik **`ticketTopic`**.
3. Pilih Terbitkan pesan.
4. Pada halaman Terbitkan pesan untuk topik, masukkan berikut ini untuk isi pesan. Tambahkan karakter baris baru di akhir pesan.

   ```
   {"BookingDate":"2020-12-15","BookingTime":"2020-12-15
    04:15:05","Destination":"Miami","FlyingFrom":"Vancouver","TicketNumber":"abcd1234"}
   ```

   Simpan semua pilihan lain sebagai default mereka.
5. Pilih Terbitkan pesan.

Untuk informasi selengkapnya tentang menerbitkan pesan, lihat Menerbitkan SNS pesan Amazon.

6. Setelah interval aliran pengiriman 60 detik, buka konsol Amazon Simple Storage Service (Amazon S3) dan pilih bucket Amazon S3 yang Anda buat awalnya.

   Pesan yang diterbitkan muncul dalam bucket.

Untuk kueri data

1. Buka konsol Amazon Athena.
2. Jalankan kueri.

   Sebagai contoh, asumsikan bahwa tabel `notifications` di skema `default` berisi data berikut ini:

```
{"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"}
```

   Untuk menemukan tujuan teratas, jalankan kueri berikut ini:

```
SELECT destination
FROM default.notifications
GROUP BY destination
ORDER BY count(*) desc
LIMIT 1;
```

   Untuk kueri tiket yang terjual selama tanggal dan rentang waktu tertentu, jalankan kueri seperti berikut ini:

```
SELECT *
FROM default.notifications
WHERE bookingtime
  BETWEEN TIMESTAMP '2020-12-15 10:00:00'
```

```
  AND TIMESTAMP '2020-12-15 12:00:00';
```

Anda dapat menyesuaikan kedua kueri sampel untuk kebutuhan Anda sendiri. Untuk informasi selengkapnya tentang menggunakan Athena untuk menjalankan kueri, lihat Memulai di Panduan Pengguna Amazon Athena.

Membersihkan

Untuk menghindari menimbulkan biaya penggunaan setelah Anda selesai melakukan pengujian, hapus sumber daya berikut ini yang Anda buat selama tutorial:

- SNSLangganan Amazon
- SNSTopik Amazon
- Antrian Layanan Antrian Sederhana Amazon (AmazonSQS)
- Bucket Amazon S3
- Aliran pengiriman Amazon Data Firehose
- AWS Identity and Access Management (IAM) peran dan kebijakan

Mengotomatiskan pengarsipan SNS pesan Amazon dengan templat AWS CloudFormation

Untuk mengotomatiskan penerapan kasus penggunaan contoh pengarsipan SNS pesan Amazon dan analisis, Anda dapat menggunakan templat berikut: YAML

```
---
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
```

SNSPemberitahuan Amazon Fanout ke fungsi Lambda untuk pemrosesan otomatis

Amazon SNS dan AWS Lambda terintegrasi sehingga Anda dapat menjalankan fungsi Lambda dengan notifikasi AmazonSNS. Ketika pesan dipublikasikan ke SNS topik yang memiliki fungsi Lambda berlangganan, fungsi Lambda dipanggil dengan muatan pesan yang diterbitkan. Fungsi Lambda menerima payload pesan sebagai parameter input dan dapat memanipulasi informasi dalam pesan, mempublikasikan pesan ke SNS topik lain, atau mengirim pesan ke layanan lain. AWS

Amazon SNS juga mendukung atribut status pengiriman pesan untuk pemberitahuan pesan yang dikirim ke titik akhir Lambda. Untuk informasi selengkapnya, lihat Status pengiriman SNS pesan Amazon.

Topik

- Prasyarat untuk mengintegrasikan Amazon dengan SNS fungsi Lambda di seluruh wilayah
- Berlangganan fungsi Lambda ke topik Amazon SNS

Prasyarat untuk mengintegrasikan Amazon dengan SNS fungsi Lambda di seluruh wilayah

Untuk menjalankan fungsi Lambda menggunakan notifikasi SNS Amazon, Anda memerlukan yang berikut ini:

- Fungsi Lambda
- SNSTopik Amazon

Untuk informasi tentang membuat fungsi Lambda yang akan digunakan dengan AmazonSNS, lihat Menggunakan Lambda dengan Amazon. SNS Untuk informasi tentang membuat SNS topik Amazon, lihat Membuat topik.

Saat Anda menggunakan Amazon SNS untuk mengirimkan pesan dari wilayah keikutsertaan ke wilayah yang diaktifkan secara default, Anda harus mengubah kebijakan yang dibuat dalam fungsi AWS Lambda dengan mengganti prinsipal dengan. `sns.amazonaws.com` `sns.<opt-in-region>.amazonaws.com`

Misalnya, jika Anda ingin berlangganan fungsi Lambda di AS Timur (Virginia N.) ke SNS topik di Asia Pasifik (Hong Kong), ubah prinsipal dalam kebijakan fungsi AWS Lambda menjadi. `sns.ap-east-1.amazonaws.com` Wilayah keikutsertaan mencakup semua wilayah yang diluncurkan setelah 20 Maret 2019, yang meliputi Asia Pacific (Hong Kong), Middle East (Bahrain), UE (Milano), dan Africa (Cape Town). Wilayah yang diluncurkan sebelum 20 Maret 2019 diaktifkan secara default.

> **Note**
>
> AWS tidak mendukung pengiriman lintas wilayah ke Lambda dari wilayah yang diaktifkan secara default ke wilayah keikutsertaan. Selain itu, penerusan SNS pesan lintas wilayah dari wilayah keikutsertaan ke wilayah keikutsertaan lainnya tidak didukung.

Berlangganan fungsi Lambda ke topik Amazon SNS

1. Masuk ke SNSkonsol Amazon.
2. Di panel navigasi, pilih Topik.
3. Pada Topics (Topik), pilih sebuah topik.
4. Di bagian Subscriptions (Berlangganan), pilih Create subscription (Buat langganan).
5. Pada halaman Create subscription (Buat langganan), di bagian Details (Rincian), lakukan hal berikut:

    a. Verifikasi Topik yang dipilihARN.
    b. Untuk Protokol pilih AWS Lambda.
    c. Untuk Endpoint masukkan ARN fungsi.

d. Pilih Buat langganan.

Ketika pesan dipublikasikan ke SNS topik yang memiliki fungsi Lambda berlangganan, fungsi Lambda dipanggil dengan muatan pesan yang diterbitkan. Untuk informasi tentang cara menggunakan AWS Lambda dengan AmazonSNS, termasuk tutorial, lihat Menggunakan AWS Lambda dengan Amazon SNS.

SNSPemberitahuan Fanout Amazon ke SQS antrian Amazon untuk pemrosesan asinkron

Amazon SNS bekerja sama dengan Amazon Simple Queue Service (AmazonSQS). Layanan ini memberikan manfaat yang berbeda bagi developer. Amazon SNS memungkinkan aplikasi untuk mengirim pesan penting waktu ke beberapa pelanggan melalui mekanisme “push”, menghilangkan kebutuhan untuk memeriksa atau “polling” secara berkala untuk pembaruan. Amazon SQS adalah layanan antrian pesan yang digunakan oleh aplikasi terdistribusi untuk bertukar pesan melalui model polling, dan dapat digunakan untuk memisahkan komponen pengiriman dan penerima—tanpa mengharuskan setiap komponen tersedia secara bersamaan. Menggunakan Amazon SNS dan Amazon SQS bersama-sama, pesan dapat dikirim ke aplikasi yang memerlukan pemberitahuan segera dari suatu peristiwa, dan juga bertahan dalam SQS antrian Amazon untuk aplikasi lain untuk diproses di lain waktu.

Saat Anda berlangganan SQS antrian Amazon ke SNS topik Amazon, Anda dapat mempublikasikan pesan ke topik tersebut dan Amazon SNS mengirimkan SQS pesan Amazon ke antrian berlangganan. SQSPesan Amazon berisi subjek dan pesan yang dipublikasikan ke topik bersama dengan metadata tentang pesan dalam dokumen. JSON SQSPesan Amazon akan terlihat mirip dengan JSON dokumen berikut.

```
{
   "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"
}
```

Berlangganan SQS antrian Amazon ke topik Amazon SNS

Untuk mengaktifkan SNS topik Amazon untuk mengirim pesan ke SQS antrian Amazon, lakukan salah satu hal berikut:

- Gunakan SQSkonsol Amazon, yang menyederhanakan prosesnya. Untuk informasi selengkapnya, lihat Berlangganan SQS antrian Amazon ke SNS topik Amazon di Panduan Pengembang Layanan Antrian Sederhana Amazon.
- Ikuti langkah-langkah ini:
  1. Dapatkan Amazon Resource Name (ARN) dari antrian yang ingin Anda kirimi pesan dan topik yang Anda inginkan untuk berlangganan antrean.
  2. Berikan `sqs:SendMessage` izin ke SNS topik Amazon sehingga dapat mengirim pesan ke antrian.
  3. Berlangganan antrian ke SNS topik Amazon.
  4. Berikan IAM pengguna atau izin Akun AWS yang sesuai untuk mempublikasikan ke SNS topik Amazon dan membaca pesan dari SQS antrian Amazon.
  5. Uji dengan menerbitkan pesan ke topik dan membaca pesan dari antrian.

Untuk mempelajari cara mengatur topik untuk mengirim pesan ke antrean yang ada di AWS akun lain, lihat. Mengirim SNS pesan Amazon ke SQS antrian Amazon di akun yang berbeda

Untuk melihat AWS CloudFormation template yang membuat topik yang mengirim pesan ke dua antrian, lihat. Otomatiskan SQS pesan Amazon SNS ke Amazon dengan AWS CloudFormation

Langkah 1: Dapatkan antrian dan topik ARN

Saat berlangganan antrian ke topik Anda, Anda memerlukan salinan ARN untuk antrian. Demikian pula, ketika memberikan izin untuk topik untuk mengirim pesan ke antrian, Anda akan memerlukan salinan ARN untuk topik tersebut.

Untuk mendapatkan antrianARN, Anda dapat menggunakan SQS konsol Amazon atau GetQueueAttributesAPItindakan.

Untuk mendapatkan antrian ARN dari konsol Amazon SQS

1. Masuk ke AWS Management Console dan buka SQS konsol Amazon di https://console.aws.amazon.com/sqs/.
2. Pilih kotak untuk antrian yang ingin ARN Anda dapatkan.
3. Dari bagian Detail, salin ARN nilainya sehingga Anda dapat menggunakannya untuk berlangganan SNS topik Amazon.

Untuk mendapatkan topikARN, Anda dapat menggunakan SNS konsol Amazon, `sns-get-topic-attributes` perintah, atau `GetQueueAttributes` API tindakan.

Untuk mendapatkan topik ARN dari SNS konsol Amazon

1. Masuk ke SNSkonsol Amazon.
2. Pada panel navigasi, pilih topik yang ingin ARN Anda dapatkan.
3. Dari bagian Detail, salin ARNnilainya sehingga Anda dapat menggunakannya untuk memberikan izin bagi SNS topik Amazon untuk mengirim pesan ke antrian.

Langkah 2: Berikan izin ke SNS topik Amazon untuk mengirim pesan ke SQS antrian Amazon

Agar SNS topik Amazon dapat mengirim pesan ke antrian, Anda harus menetapkan kebijakan pada antrian yang memungkinkan SNS topik Amazon melakukan tindakan. `sqs:SendMessage`

Sebelum Anda berlangganan antrean ke topik, Anda memerlukan topik dan antrean. Jika Anda belum membuat topik atau antrean, buat sekarang. Untuk informasi selengkapnya, lihat Membuat topik, dan lihat Membuat antrian di Panduan Pengembang Layanan Antrian Sederhana Amazon.

Untuk menetapkan kebijakan pada antrean, Anda dapat menggunakan SQS konsol Amazon atau SetQueueAttributesAPItindakan. Sebelum memulai, pastikan Anda memiliki topik yang ingin Anda izinkan untuk mengirim pesan ke antrian. ARN Jika Anda berlangganan antrian ke beberapa topik, kebijakan Anda harus berisi satu `Statement` elemen untuk setiap topik.

Untuk menetapkan SendMessage kebijakan pada antrian menggunakan konsol Amazon SQS

1. Masuk ke AWS Management Console dan buka SQS konsol Amazon di https://console.aws.amazon.com/sqs/.

2. Pilih kotak untuk antrean kebijakan yang ingin Anda tetapkan, pilih tab Kebijakan akses, lalu pilih Edit.
3. Di Kebijakan akses, tentukan siapa yang dapat mengakses antrean Anda.

   - Tambahkan kondisi yang memungkinkan tindakan untuk topik.
   - Setel `Principal` menjadi SNS layanan Amazon, seperti yang ditunjukkan pada contoh di bawah ini.
   - Gunakan `aws:SourceArn`atau kunci kondisi `aws:SourceAccount`global untuk melindungi dari skenario wakil yang membingungkan. Untuk menggunakan kunci kondisi ini, tetapkan nilainya ke ARN topik Anda. Jika antrian Anda berlangganan beberapa topik, Anda dapat menggunakannya `aws:SourceAccount` sebagai gantinya.

   Misalnya, kebijakan berikut memungkinkan MyTopic untuk mengirim pesan ke MyQueue.

```
{
  "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"
        }
      }
    }
  ]
}
```

Langkah 3: Berlangganan antrian ke topik Amazon SNS

Untuk mengirim pesan ke antrian melalui topik, Anda harus berlangganan antrian ke topik AmazonSNS. Anda menentukan antrian dengan nyaARN. Untuk berlangganan topik, Anda dapat menggunakan SNS konsol Amazon, `sns-subscribe` CLI perintah, atau `Subscribe` API tindakan. Sebelum Anda mulai, pastikan Anda memiliki antrian yang ingin Anda berlangganan. ARN

1. Masuk ke SNSkonsol Amazon.
2. Di panel navigasi, pilih Topik.
3. PadaTopik, pilih topik.
4. Pada ***MyTopic*** halaman, di halaman Langganan, pilih Buat langganan.
5. Pada Create subscription (Buat langganan), di halaman Details (Rincian), lakukan hal berikut:

   a. Verifikasi Topiknya ARN.

   b. Untuk Protokol, pilih Amazon SQS.

   c. Untuk Endpoint, masukkan ARN SQS antrean Amazon.

   d. Pilih Buat Langganan.

   Saat langganan dikonfirmasi, ID Langganan baru Anda akan menampilkan ID langganannya. Jika pemilik antrean membuat langganan, langganan secara otomatis dikonfirmasi dan langganan harus segera aktif.

   Biasanya, Anda akan berlangganan antrean Anda sendiri ke topik Anda sendiri di akun Anda sendiri. Namun, Anda juga dapat berlangganan antrean dari akun yang berbeda ke topik Anda. Jika pengguna yang membuat langganan bukan pemilik antrean (misalnya, jika pengguna dari akun A berlangganan antrean dari akun B ke topik di akun A), langganan harus dikonfirmasi. Untuk informasi selengkapnya tentang berlangganan antrean dari akun lain dan mengonfirmasi langganan, lihat Mengirim SNS pesan Amazon ke SQS antrian Amazon di akun yang berbeda.

Langkah 4: Memberikan pengguna izin untuk topik yang sesuai dan tindakan antrian

Anda harus menggunakan AWS Identity and Access Management (IAM) untuk mengizinkan hanya pengguna yang sesuai untuk mempublikasikan ke SNS topik Amazon dan membaca/menghapus pesan dari antrian AmazonSQS. Untuk informasi selengkapnya tentang mengontrol tindakan pada topik dan antrian bagi IAM pengguna, lihatMenggunakan kebijakan berbasis identitas dengan Amazon SNS, serta Manajemen identitas dan akses SQS di Amazon dalam Panduan Pengembang Layanan Antrian Sederhana Amazon.

Ada dua cara untuk mengontrol akses ke topik atau antrean:

- Tambahkan kebijakan ke IAM pengguna atau grup. Cara termudah untuk memberikan pengguna izin untuk topik atau antrean adalah untuk membuat grup dan menambahkan kebijakan yang sesuai untuk grup dan kemudian menambahkan pengguna ke grup tersebut. Lebih mudah

menambahkan dan menghapus pengguna dari grup daripada melacak kebijakan yang Anda tetapkan pada pengguna individual.

- Menambahkan kebijakan ke topik atau antrean. Jika Anda ingin memberikan izin ke topik atau antrian ke AWS akun lain, satu-satunya cara yang dapat Anda lakukan adalah dengan menambahkan kebijakan yang memiliki prinsipal yang ingin Akun AWS Anda berikan izin.

Anda harus menggunakan metode pertama untuk sebagian besar kasus (menerapkan kebijakan untuk grup dan mengelola izin untuk pengguna dengan menambahkan atau menghapus pengguna yang sesuai ke grup). Jika Anda perlu memberikan izin kepada pengguna di akun lain, Anda harus menggunakan metode kedua.

Menambahkan kebijakan ke IAM pengguna atau grup

Jika Anda menambahkan kebijakan berikut ke IAM pengguna atau grup, Anda akan memberikan izin kepada pengguna atau anggota grup tersebut untuk melakukan `sns:Publish` tindakan pada topik tersebut MyTopic.

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic"
    }
  ]
}
```

Jika Anda menambahkan kebijakan berikut ke IAM pengguna atau grup, Anda akan memberikan izin kepada pengguna atau anggota grup tersebut untuk melakukan `sqs:ReceiveMessage` dan `sqs:DeleteMessage` tindakan pada antrian MyQueue 1 dan MyQueue 2.

```
{
  "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"
      ]
    }
  ]
}
```

Menambahkan kebijakan ke topik atau antrean

Contoh kebijakan berikut menunjukkan bagaimana memberikan izin akun lain untuk topik dan antrean.

> **Note**
>
> Saat Anda memberikan Akun AWS akses lain ke sumber daya di akun Anda, Anda juga memberi IAM pengguna yang memiliki izin akses tingkat admin (akses wildcard) ke sumber daya tersebut. Semua IAM pengguna lain di akun lain secara otomatis ditolak akses ke sumber daya Anda. Jika Anda ingin memberikan IAM pengguna tertentu dalam Akun AWS akses tersebut ke sumber daya Anda, akun atau IAM pengguna dengan akses tingkat admin harus mendelegasikan izin untuk sumber daya tersebut kepada pengguna tersebut. IAM Untuk informasi selengkapnya tentang delegasi lintas akun, lihat Mengaktifkan Akses Lintas Akun di Panduan Menggunakan. IAM

Jika Anda menambahkan kebijakan berikut ke topik MyTopic di akun 123456789012, Anda akan memberi akun 111122223333 izin untuk melakukan tindakan pada topik tersebut. `sns:Publish`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111122223333"
      },
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic"
    }
  ]
}
```

Jika Anda menambahkan kebijakan berikut ke antrian MyQueue di akun 123456789012, Anda akan memberi akun 111122223333 izin untuk melakukan dan tindakan pada antrian tersebut. `sqs:ReceiveMessage` `sqs:DeleteMessage`

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "111122223333"
      },
      "Action": [
        "sqs:DeleteMessage",
        "sqs:ReceiveMessage"
      ],
      "Resource": [
        "arn:aws:sqs:us-east-2:123456789012:MyQueue"
      ]
    }
  ]
}
```

Langkah 5: Uji langganan antrean topik

Anda dapat menguji langganan antrean topik dengan menerbitkan topik dan melihat pesan yang dikirim topik ke antrean.

Untuk mempublikasikan ke topik menggunakan SNS konsol Amazon

1. Dengan menggunakan kredensi Akun AWS atau IAM pengguna dengan izin untuk mempublikasikan ke topik, masuk ke AWS Management Console dan buka SNS konsol Amazon di. https://console.aws.amazon.com/sns/
2. Pada panel navigasi, pilih topik dan pilih Publish to Topic (Publikasikan ke topik).
3. Di kotak Subject (Subjek), masukkan subjek (misalnya, **`Testing publish to queue`**) di kotak Pesan, masukkan beberapa teks (misalnya, **`Hello world!`**), dan pilih Publish Message (Publikasikan Pesan). Pesan berikut muncul: Pesan Anda telah berhasil dipublikasikan.

Untuk melihat pesan dari topik menggunakan SQS konsol Amazon

1. Menggunakan kredensional IAM pengguna Akun AWS atau dengan izin untuk melihat pesan dalam antrian, masuk ke AWS Management Console dan buka konsol Amazon SQS di. https://console.aws.amazon.com/sqs/
2. Pilih antrian yang berlangganan topik.
3. Pilih Kirim dan terima pesan, lalu pilih Poll untuk pesan. Sebuah pesan dengan jenis Pemberitahuan akan muncul.
4. Di kolom Body (Isi), pilih More Details (Detail lebih lanjut). Kotak Rincian Pesan berisi JSON dokumen yang berisi subjek dan pesan yang Anda terbitkan ke topik. Pesannya terlihat mirip dengan JSON dokumen berikut.

```
{
  "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. Pilih Close (Tutup). Anda telah berhasil menerbitkan topik yang mengirimkan pesan pemberitahuan ke antrean.

Otomatiskan SQS pesan Amazon SNS ke Amazon dengan AWS CloudFormation

AWS CloudFormation memungkinkan Anda untuk menggunakan file template untuk membuat dan mengkonfigurasi kumpulan AWS sumber daya bersama-sama sebagai satu unit. Bagian ini memiliki contoh templat yang membuatnya mudah untuk menyebarkan topik yang mempublikasikan ke antrean. Template menangani langkah-langkah penyiapan untuk Anda dengan membuat dua

antrian, membuat topik dengan langganan antrian, menambahkan kebijakan ke antrian sehingga topik dapat mengirim pesan ke antrian, dan membuat IAM pengguna dan grup untuk mengontrol akses ke sumber daya tersebut.

Untuk informasi selengkapnya tentang penerapan AWS sumber daya menggunakan AWS CloudFormation templat, lihat Memulai di Panduan AWS CloudFormation Pengguna.

Menggunakan AWS CloudFormation template untuk mengatur topik dan antrian dalam Akun AWS

Contoh template membuat SNS topik Amazon yang dapat mengirim pesan ke dua SQS antrian Amazon dengan izin yang sesuai untuk anggota satu IAM grup untuk mempublikasikan ke topik dan yang lain untuk membaca pesan dari antrian. Template juga membuat IAM pengguna yang ditambahkan ke setiap grup.

Anda menyalin isi templat ke dalam file. Anda juga dapat mengunduh template dari halaman AWS CloudFormation Template. Pada halaman template, pilih Browse sample templates by AWS service lalu pilih Amazon Simple Queue Service.

M ySNSTopic diatur untuk mempublikasikan ke dua titik akhir berlangganan, yang merupakan dua SQS antrian Amazon (MyQueue1 dan 2). MyQueue MyPublishTopicGroup adalah IAM grup yang anggotanya memiliki izin untuk mempublikasikan ke M ySNSTopic menggunakan API tindakan Publikasikan atau perintah sns-publish. Template menciptakan IAM pengguna MyPublishUser dan MyQueueUser dan memberi mereka profil login dan kunci akses. Pengguna yang membuat tumpukan dengan templat ini menentukan password untuk profil login sebagai parameter input. Template membuat kunci akses untuk dua IAM pengguna dengan MyPublishUserKey dan MyQueueUserKey. AddUserToMyPublishTopicGroup MyPublishUser menambah MyPublishTopicGroup sehingga pengguna akan memiliki izin yang ditetapkan ke grup.

M yRDMessage QueueGroup adalah IAM grup yang anggotanya memiliki izin untuk membaca dan menghapus pesan dari dua SQS antrian Amazon menggunakan ReceiveMessagedan DeleteMessageAPItindakan. AddUserToMyQueueGroup menambahkan MyQueueUser ke M yRDMessage QueueGroup sehingga pengguna akan memiliki izin yang ditetapkan ke grup. MyQueuePolicy memberikan izin bagi M ySNSTopic untuk mempublikasikan notifikasi ke dua antrian.

Daftar berikut menunjukkan isi AWS CloudFormation template.

```
{
  "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"]
            }
          ]
        ]
      }
    }
  }
}
```

SNSPemberitahuan Fanout Amazon ke titik akhir HTTPS

Anda dapat menggunakan Amazon SNS untuk mengirim pesan notifikasi ke satu HTTP atau beberapa HTTPS titik akhir. Saat Anda berlangganan titik akhir ke suatu topik, Anda dapat mempublikasikan pemberitahuan ke topik tersebut dan Amazon SNS mengirimkan HTTP POST permintaan yang mengirimkan konten notifikasi ke titik akhir berlangganan. Saat Anda berlangganan titik akhir, Anda memilih apakah Amazon SNS menggunakan HTTP atau HTTPS mengirim POST permintaan ke titik akhir. Jika Anda menggunakanHTTPS, maka Anda dapat memanfaatkan dukungan di Amazon SNS untuk hal berikut:

- Indikasi Nama Server (SNI) —Ini memungkinkan Amazon SNS untuk mendukung HTTPS titik akhir yang memerlukanSNI, seperti server yang memerlukan beberapa sertifikat untuk menghosting beberapa domain. Untuk informasi selengkapnyaSNI, lihat Indikasi Nama Server.

- Otentikasi Akses Dasar dan Intisari —Ini memungkinkan Anda menentukan nama pengguna dan kata sandi dalam HTTP POST permintaan, seperti `https://user:password@domain.com` atau `https://user@domain.com` Nama pengguna dan kata sandi dienkripsi melalui koneksi yang dibuat saat SSL menggunakan. HTTPS URL HTTPS Hanya nama domain yang dikirim dalam plaintext. Untuk informasi selengkapnya tentang Autentikasi Akses Dasar dan Intisari, lihat RFC -2617.

  > **Important**
  >
  > Amazon saat ini SNS tidak mendukung titik akhir pribadi HTTP (S). HTTPSURLshanya dapat diambil dari SNS `GetSubscriptionAttributes` API tindakan Amazon, untuk prinsipal yang telah Anda berikan akses. API

> **Note**
>
> Layanan klien harus dapat mendukung respons header `HTTP/1.1 401 Unauthorized`

Permintaan berisi subjek dan pesan yang dipublikasikan ke topik bersama dengan metadata tentang pemberitahuan dalam dokumen. JSON Permintaan akan terlihat mirip dengan HTTP POST permintaan berikut. Untuk detail tentang HTTP header dan JSON format badan permintaan, lihat HTTP/HTTPSheader danHTTP/JSONformat HTTPS pemberitahuan.

```
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"
}
```

Topik

Berlangganan HTTPS titik akhir ke topik Amazon SNS

Halaman-halaman di bagian ini menjelaskan cara berlangganan titik akhir HTTP /S ke SNS topik Amazon.

Topik

Langkah 1: Pastikan titik akhir Anda siap untuk memproses pesan Amazon SNS

Sebelum Anda berlangganan HTTP atau HTTPS titik akhir Anda ke suatu topik, Anda harus memastikan bahwa HTTPS titik akhir HTTP atau memiliki kemampuan untuk menangani HTTP POST permintaan yang SNS digunakan Amazon untuk mengirim konfirmasi langganan dan pesan pemberitahuan. Biasanya, ini berarti membuat dan menyebarkan aplikasi web (misalnya, servlet Java jika host endpoint Anda menjalankan Linux dengan Apache dan Tomcat) yang memproses permintaan dari Amazon. HTTP SNS Saat Anda berlangganan HTTP titik akhir, Amazon SNS mengirimkannya permintaan konfirmasi berlangganan. Titik akhir Anda harus siap untuk menerima dan memproses permintaan ini saat Anda membuat langganan karena Amazon SNS mengirimkan permintaan ini pada saat itu. Amazon tidak SNS akan mengirim pemberitahuan ke titik akhir sampai Anda mengonfirmasi langganan. Setelah Anda mengonfirmasi langganan, Amazon SNS akan mengirim pemberitahuan ke titik akhir saat tindakan publikasi dilakukan pada topik berlangganan.

Menyiapkan titik akhir Anda untuk memproses konfirmasi berlangganan dan pesan notifikasi

1. Kode Anda harus membaca HTTP header HTTP POST permintaan yang SNS dikirimkan Amazon ke titik akhir Anda. Kode Anda harus mencari bidang header`x-amz-sns-message-type`, yang memberi tahu Anda jenis pesan yang SNS dikirimkan Amazon kepada Anda.

Dengan melihat header, Anda dapat menentukan jenis pesan tanpa harus mengurai isi HTTP permintaan. Ada dua jenis yang perlu Anda tangani: `SubscriptionConfirmation` dan `Notification`. Pesan `UnsubscribeConfirmation` hanya digunakan saat langganan dihapus dari topik.

Untuk detail tentang HTTP header, lihatHTTP/HTTPSheader. HTTPPOSTPermintaan berikut adalah contoh pesan konfirmasi berlangganan.

```
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. Kode Anda harus mengurai JSON dokumen di badan HTTP POST permintaan dan teks/polos tipe konten untuk membaca pasangan nama-nilai yang membentuk pesan Amazon. SNS Gunakan JSON parser yang menangani konversi representasi lolos dari karakter kontrol kembali ke nilai ASCII karakter mereka (misalnya, mengubah\nke karakter baris baru). Anda dapat menggunakan JSON parser yang ada seperti JSONProsesor Jackson atau menulis sendiri. Untuk mengirim teks dalam bidang subjek dan pesan sebagai validJSON, Amazon SNS harus

mengonversi beberapa karakter kontrol ke representasi yang lolos yang dapat disertakan dalam dokumen. JSON Saat Anda menerima JSON dokumen di badan POST permintaan yang dikirim ke titik akhir Anda, Anda harus mengonversi karakter yang diloloskan kembali ke nilai karakter aslinya jika Anda menginginkan representasi yang tepat dari subjek asli dan pesan yang dipublikasikan ke topik tersebut. Hal ini penting jika Anda ingin memverifikasi tanda tangan notifikasi karena tanda tangan menggunakan pesan dan subjek dalam bentuk aslinya sebagai bagian dari string untuk ditandatangani.

3. Kode Anda harus memverifikasi keaslian pemberitahuan, konfirmasi berlangganan, atau pesan konfirmasi berhenti berlangganan yang dikirim oleh Amazon. SNS Menggunakan informasi yang terkandung dalam SNS pesan Amazon, titik akhir Anda dapat membuat ulang tanda tangan sehingga Anda dapat memverifikasi konten pesan dengan mencocokkan tanda tangan Anda dengan tanda tangan yang SNS dikirim Amazon dengan pesan tersebut. Untuk informasi selengkapnya tentang memverifikasi tanda tangan pesan, lihat Memverifikasi tanda tangan pesan Amazon SNS.

4. Berdasarkan jenis yang ditentukan oleh bidang header`x-amz-sns-message-type`, kode Anda harus membaca JSON dokumen yang terdapat di badan HTTP permintaan dan memproses pesan. Berikut adalah panduan untuk menangani dua jenis utama pesan:

   SubscriptionConfirmation

   Baca nilainya `SubscribeURL` dan kunjungi ituURL. Untuk mengonfirmasi langganan dan mulai menerima pemberitahuan di titik akhir, Anda harus mengunjungi `SubscribeURL` URL (misalnya, dengan mengirim HTTP GET permintaan keURL). Lihat contoh HTTP permintaan di langkah sebelumnya untuk melihat seperti apa `SubscribeURL` tampilannya. Untuk informasi lebih lanjut tentang format `SubscriptionConfirmation`, lihat HTTP/JSONformat konfirmasi HTTPS berlangganan. Ketika Anda mengunjungiURL, Anda akan mendapatkan kembali respons yang terlihat seperti XML dokumen berikut. Dokumen mengembalikan langganan ARN untuk titik akhir dalam `ConfirmSubscriptionResult` elemen.

```
<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>
```

Sebagai alternatif untuk mengunjungi`SubscribeURL`, Anda dapat mengonfirmasi langganan menggunakan ConfirmSubscriptiontindakan dengan `Token` set ke nilai yang sesuai dalam `SubscriptionConfirmation` pesan. Jika Anda ingin mengizinkan hanya pemilik topik dan pemilik langganan untuk dapat berhenti berlangganan titik akhir, Anda memanggil tindakan `ConfirmSubscription` dengan tanda tangan AWS .

Pemberitahuan

Baca nilai `Subject` dan `Message` untuk mendapatkan informasi notifikasi yang dipublikasikan ke topik.

Untuk detail tentang format pesan `Notification`, lihat HTTP/HTTPSheader. HTTPPOSTPermintaan berikut adalah contoh pesan notifikasi yang dikirim ke endpoint example.com.

```
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. Pastikan titik akhir Anda merespons HTTP POST pesan dari Amazon SNS dengan kode status yang sesuai. Koneksi akan habis dalam waktu sekitar 15 detik. Jika titik akhir Anda tidak merespons sebelum waktu koneksi habis, atau jika titik akhir Anda mengembalikan kode status di luar kisaran 200—4 xx, Amazon SNS akan menganggap pengiriman pesan sebagai upaya yang gagal.
6. Pastikan kode Anda dapat menangani percobaan ulang pengiriman pesan dari AmazonSNS. Jika Amazon SNS tidak menerima respons yang berhasil dari titik akhir Anda, Amazon mencoba mengirimkan pesan lagi. Hal ini berlaku untuk semua pesan, termasuk pesan konfirmasi berlangganan. Secara default, jika pengiriman awal pesan gagal, Amazon SNS mencoba hingga tiga percobaan ulang dengan penundaan antara upaya gagal yang ditetapkan pada 20 detik.

   > **Note**
   >
   > Waktu permintaan pesan habis setelah sekitar 15 detik. Ini berarti bahwa, jika kegagalan pengiriman pesan disebabkan oleh batas waktu, Amazon SNS mencoba lagi selama sekitar 35 detik setelah upaya pengiriman sebelumnya. Anda dapat menetapkan kebijakan pengiriman yang berbeda untuk titik akhir.

   Amazon SNS menggunakan bidang `x-amz-sns-message-id` header untuk mengidentifikasi secara unik setiap pesan yang dipublikasikan ke topik AmazonSNS. Dengan membandingkan pesan IDs yang telah diproses dengan pesan masuk, Anda dapat menentukan apakah pesan tersebut merupakan upaya coba lagi.
7. Jika Anda berlangganan HTTPS endpoint, pastikan endpoint Anda memiliki sertifikat server dari Certificate Authority (CA) tepercaya. Amazon hanya SNS akan mengirim pesan ke HTTPS titik akhir yang memiliki sertifikat server yang ditandatangani oleh CA yang dipercaya oleh AmazonSNS.
8. Terapkan kode yang telah Anda buat untuk menerima SNS pesan Amazon. Saat Anda melangganken titik akhir, titik akhir harus siap untuk menerima setidaknya pesan konfirmasi berlangganan.

Langkah 2: Berlangganan HTTPS endpoint HTTP /ke topik Amazon SNS

Untuk mengirim pesan ke HTTP atau HTTPS titik akhir melalui topik, Anda harus berlangganan titik akhir ke topik AmazonSNS. Anda menentukan titik akhir menggunakan nyaURL. Untuk berlangganan topik, Anda dapat menggunakan SNS konsol Amazon, perintah sns-subscribe, atau tindakan BerlanggananAPI. Sebelum memulai, pastikan Anda memiliki titik akhir yang ingin Anda berlangganan dan titik akhir Anda siap menerima pesan konfirmasi dan pemberitahuan seperti yang dijelaskan pada Langkah 1. URL

Untuk berlangganan HTTP atau HTTPS titik akhir topik menggunakan konsol Amazon SNS

1. Masuk ke SNSkonsol Amazon.
2. Di panel navigasi, pilih Topik.
3. Pilih langganan Buat.
4. Dalam daftar drop-down Protocol, pilih HTTPatau HTTPS.
5. Di kotak Endpoint, tempelkan titik akhir yang ingin Anda kirimi pesan topik, lalu pilih Buat langganan. URL
6. Pesan konfirmasi ditampilkan. Pilih Tutup.

   ID Langganan baru Anda ditampilkan PendingConfirmation. Saat Anda mengonfirmasi langganan, ID langganan akan menampilkan ID langganan.

Langkah 3: Konfirmasikan SNS langganan Amazon Anda

Untuk mengonfirmasi SNS langganan AWS Amazon, ikuti langkah-langkah berikut untuk memastikan titik akhir Anda berhasil menerima pesan. Proses ini melibatkan pengaturan titik akhir Anda untuk menangani pesan konfirmasi yang masuk, mengambil konfirmasi yang diperlukanURL, dan mengonfirmasi langganan melalui cara otomatis atau manual.

1. Pesan konfirmasi berlangganan. Setelah Anda berlangganan titik akhir Anda ke SNS topik Amazon, Amazon SNS mengirimkan pesan konfirmasi ke titik akhir tersebut. Pesan ini berisi`SubscribeURL`, yang Anda butuhkan untuk mengkonfirmasi langganan.
2. Ambil kembali. **`SubscribeURL`** Titik akhir Anda harus memiliki kode yang mendengarkan dan memproses pesan masuk. Kode ini harus mengekstrak `SubscribeURL` dari pesan konfirmasi. Pesan konfirmasi biasanya tiba sebagai JSON muatan dengan `SubscribeURL` kunci.
3. Konfirmasikan langganan. Ada dua cara untuk mengonfirmasi langganan:

- Konfirmasi otomatis. Kode endpoint Anda dapat secara otomatis mengunjungi `SubscribeURL` untuk mengonfirmasi langganan. Pendekatan ini mengharuskan titik akhir Anda untuk membuat HTTP GET permintaan ke yang URL disediakan.
- Konfirmasi manual. Jika konfirmasi otomatis tidak diatur, Anda dapat secara manual mengunjungi `SubscribeURL` menggunakan browser web. Langkah ini melibatkan menyalin URL dari pesan dan menempelkannya ke bilah alamat browser Anda.

4. Verifikasi status langganan. Setelah Anda mengonfirmasi langganan dengan mengunjungi`SubscribeURL`, Amazon SNS mengirimkan respons yang menyertakan XML dokumen dengan elemen yang disebut`SubscriptionArn`. Elemen ini berisi Amazon Resource Name (ARN) untuk langganan, yang menunjukkan bahwa langganan aktif.
5. Gunakan SNS konsol Amazon. Anda juga dapat memverifikasi status langganan menggunakan AWS Management Console. Arahkan ke SNS dasbor Amazon, dan di bawah bagian Langganan, temukan langganan Anda. Langganan yang dikonfirmasi akan menampilkannyaARN, sedangkan langganan yang belum dikonfirmasi akan ditampilkan`PendingConfirmation`.

Langkah 4: Opsional - Tetapkan kebijakan pengiriman untuk SNS langganan Amazon

Secara default, jika pengiriman awal pesan gagal, Amazon SNS mencoba hingga tiga percobaan ulang dengan penundaan antara upaya gagal yang ditetapkan pada 20 detik. Sebagaimana dibahas dalam Langkah 1, titik akhir Anda harus memiliki kode yang dapat menangani pesan yang dicoba kembali. Dengan menyetel kebijakan pengiriman pada topik atau langganan, Anda dapat mengontrol frekuensi dan interval Amazon SNS akan mencoba kembali pesan yang gagal. Anda juga dapat menentukan jenis konten untuk notifikasi HTTP /S Anda di`DeliveryPolicy`. Untuk informasi selengkapnya, lihat Membuat kebijakan pengiriman HTTP /S.

Langkah 5: Opsional - Berikan izin pengguna untuk mempublikasikan ke topik Amazon SNS

Secara default, pemilik topik memiliki izin untuk memublikasikan ke topik. Untuk mengaktifkan pengguna atau aplikasi lain untuk mempublikasikan ke topik, Anda harus menggunakan AWS Identity and Access Management (IAM) untuk memberikan izin publikasi ke topik. Untuk informasi selengkapnya tentang memberikan izin untuk SNS tindakan Amazon kepada IAM pengguna, lihatMenggunakan kebijakan berbasis identitas dengan Amazon SNS.

Ada dua cara untuk mengontrol akses ke topik:

- Tambahkan kebijakan ke IAM pengguna atau grup. Cara termudah untuk memberikan pengguna izin untuk mengakses topik adalah membuat grup dan menambahkan kebijakan yang sesuai ke grup dan kemudian menambahkan pengguna ke grup tersebut. Menambahkan dan menghapus pengguna dari grup jauh lebih mudah daripada melacak kebijakan yang Anda tetapkan pada pengguna individual.
- Menambahkan kebijakan ke topik. Jika Anda ingin memberikan izin untuk mengakses topik kepada akun AWS lain, satu-satunya cara yang dapat Anda lakukan adalah dengan menambahkan kebijakan yang sebagai prinsipnya memiliki Akun AWS yang ingin Anda berikan izin.

Anda harus menggunakan metode pertama untuk sebagian besar kasus (menerapkan kebijakan pada grup dan mengelola izin untuk pengguna dengan menambahkan atau menghapus pengguna yang sesuai ke grup). Jika Anda perlu memberikan izin kepada pengguna di akun lain, gunakan metode kedua.

Jika Anda menambahkan kebijakan berikut ke IAM pengguna atau grup, Anda akan memberikan izin kepada pengguna atau anggota grup tersebut untuk melakukan `sns:Publish` tindakan pada topik tersebut MyTopic.

```
{
  "Statement":[{
    "Sid":"AllowPublishToMyTopic",
    "Effect":"Allow",
    "Action":"sns:Publish",
    "Resource":"arn:aws:sns:us-east-2:123456789012:MyTopic"
  }]
}
```

Kebijakan contoh berikut menunjukkan cara memberikan izin untuk mengakses topik kepada akun lain.

> **Note**
>
> Saat Anda memberikan Akun AWS akses lain ke sumber daya di akun Anda, Anda juga memberi IAM pengguna yang memiliki izin akses tingkat admin (akses wildcard) ke sumber daya tersebut. Semua IAM pengguna lain di akun lain secara otomatis ditolak akses ke sumber daya Anda. Jika Anda ingin memberikan IAM pengguna tertentu dalam Akun AWS akses tersebut ke sumber daya Anda, akun atau IAM pengguna dengan akses tingkat admin harus mendelegasikan izin untuk sumber daya tersebut kepada pengguna tersebut. IAM

> Untuk informasi selengkapnya tentang delegasi lintas akun, lihat Mengaktifkan Akses Lintas Akun di Panduan Menggunakan. IAM

Jika Anda menambahkan kebijakan berikut ke topik MyTopic di akun 123456789012, Anda akan memberi akun 111122223333 izin untuk melakukan tindakan pada topik tersebut. `sns:Publish`

```
{
  "Statement":[{
    "Sid":"Allow-publish-to-topic",
    "Effect":"Allow",
      "Principal":{
        "AWS":"111122223333"
      },
    "Action":"sns:Publish",
    "Resource":"arn:aws:sns:us-east-2:123456789012:MyTopic"
  }]
}
```

Langkah 6: Kirim SNS pesan Amazon ke titik HTTPS akhirHTTP/

Anda dapat mengirimkan pesan ke langganan topik dengan memublikasikan ke topik. Untuk mempublikasikan ke suatu topik, Anda dapat menggunakan SNS konsol Amazon, `sns-publish` CLI perintah, atau `Publish`API.

Jika Anda mengikuti Langkah 1, kode yang Anda deploy di titik akhir Anda harus memproses notifikasi.

Untuk mempublikasikan ke topik menggunakan SNS konsol Amazon

1. Dengan menggunakan kredensi Akun AWS atau IAM pengguna dengan izin untuk mempublikasikan ke topik, masuk ke AWS Management Console dan buka SNS konsol Amazon di. https://console.aws.amazon.com/sns/
2. Pada panel navigasi, pilih Topik dan kemudian pilih topik.
3. Pilih tombol Publikasikan pesan.
4. Di kotak Subjek, masukkan subjek (misalnya, **`Testing publish to my endpoint`**).
5. Di kotak Pesan, masukkan beberapa teks (misalnya, **`Hello world!`**), dan pilih Publikasikan pesan.

Pesan berikut muncul: Pesan Anda telah berhasil dipublikasikan.

Memverifikasi tanda tangan pesan Amazon SNS

Untuk memverifikasi keaslian pesan yang dikirim ke HTTP titik akhir Anda oleh AmazonSNS, Anda dapat memverifikasi tanda tangan pesan. Ada dua kasus di mana kami merekomendasikan untuk memverifikasi keaslian pesan. Pertama, ketika Amazon SNS mengirim pesan ke HTTP titik akhir Anda bahwa Anda berlangganan topik. Kedua, ketika Amazon SNS mengirimi Anda pesan konfirmasi ke HTTP titik akhir Anda setelah eksekusi `Subscribe` atau `Unsubscribe` API tindakan.

Anda harus melakukan hal berikut saat memverifikasi pesan yang dikirim oleh AmazonSNS:

- Selalu gunakan HTTPS saat mendapatkan sertifikat dari AmazonSNS.
- Validasi keaslian sertifikat.
- Verifikasi sertifikat diterima dari AmazonSNS.
- Jika memungkinkan, gunakan salah satu yang didukung Amazon AWS SDKs SNS untuk memvalidasi dan memverifikasi pesan.
- Validasi bahwa SNS pesan Amazon diterima dari yang Anda inginkan`TopicArn`.

Amazon SNS mendukung dua versi tanda tangan pesan:

- `SignatureVersion`1: Amazon SNS membuat tanda tangan berdasarkan SHA1hash pesan.
- `SignatureVersion`2: Amazon SNS membuat tanda tangan berdasarkan SHA256hash pesan.

Untuk mengonfigurasi versi tanda tangan pesan pada SNS topik Amazon

Secara default, SNS topik Amazon menggunakan `SignatureVersion` 1. Untuk memilih algoritma hashing pada SNS topik Amazon Anda, baik `SignatureVersion` 1 (SHA1) atau `SignatureVersion` 2 (SHA256), Anda dapat menggunakan `SetTopicAttributes` API tindakan.

Contoh kode berikut menunjukkan cara mengatur atribut topik `SignatureVersion` menggunakan AWS CLI:

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-east-2:123456789012:MyTopic \
    --attribute-name SignatureVersion \
```

```
    --attribute-value 2
```

Untuk memverifikasi tanda tangan SNS pesan Amazon saat menggunakan HTTP permintaan berbasis kueri

1. Ekstrak pasangan nama-nilai dari JSON dokumen di badan HTTP POST permintaan yang SNS dikirimkan Amazon ke titik akhir Anda. Anda akan menggunakan nilai-nilai dari beberapa pasangan nama-nilai untuk membuat string-to-sign. Saat memverifikasi tanda tangan SNS pesan Amazon, penting bagi Anda untuk mengonversi karakter kontrol yang diloloskan ke representasi karakter aslinya dalam nilai `Message` dan`Subject`. Nilai-nilai ini harus dalam bentuk aslinya saat Anda menggunakannya sebagai bagian dari string-to-sign. Untuk informasi tentang cara mengurai JSON dokumen, lihatLangkah 1: Pastikan titik akhir Anda siap untuk memproses pesan Amazon SNS.

   Ini `SignatureVersion` memberi tahu Anda versi tanda tangan yang digunakan oleh Amazon SNS untuk menghasilkan tanda tangan pesan. Dari versi tanda tangan, Anda dapat menentukan persyaratan untuk cara menghasilkan tanda tangan. Untuk notifikasi, Amazon SNS saat ini mendukung versi tanda tangan 1 dan 2. Bagian ini menyediakan langkah-langkah untuk memverifikasi tanda tangan menggunakan versi tanda tangan ini.
2. Dapatkan sertifikat X509 yang SNS digunakan Amazon untuk menandatangani pesan. Nilai `SigningCertURL` menunjuk ke lokasi sertifikat X509 yang digunakan untuk membuat tanda tangan digital untuk pesan. Ambil sertifikat dari lokasi ini.
3. Ekstraksi kunci publik dari sertifikat. Kunci publik dari sertifikat yang ditentukan oleh `SigningCertURL` digunakan untuk memverifikasi keaslian dan integritas pesan.
4. Tentukan jenis pesan. Format string-to-sign tergantung pada jenis pesan, yang ditentukan oleh nilai `Type`.
5. Buat string-to-sign. String-to-sign adalah daftar pasangan nama-nilai tertentu yang dibatasi karakter baris baru dari pesan. Setiap pasangan nilai diwakili oleh nama terlebih dahulu diikuti dengan karakter baris baru, diikuti dengan nilai, dan diakhiri dengan karakter baris baru. Pasangan nama-nilai harus tercantum dalam urutan byte-sort.

   Tergantung pada jenis pesan, string-to-sign harus memiliki pasangan nama-nilai berikut.

   Pemberitahuan

   Pesan notifikasi harus berisi pasangan nama-nilai berikut:

```
Message
MessageId
Subject (if included in the message)
Timestamp
TopicArn
Type
```

Contoh berikut adalah string-to-sign untuk `Notification`.

```
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 dan UnsubscribeConfirmation

Pesan `SubscriptionConfirmation` dan `UnsubscribeConfirmation` harus berisi pasangan nama-nilai berikut:

```
Message
MessageId
SubscribeURL
Timestamp
Token
TopicArn
Type
```

Contoh berikut adalah string-to-sign untuk `SubscriptionConfirmation`.

```
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. Dekodekan nilai `Signature` dari format Base64. Pesan mengirimkan tanda tangan dalam nilai `Signature`, yang dienkodekan sebagai Base64. Sebelum Anda membandingkan nilai tanda tangan dengan tanda tangan yang telah Anda hitung, pastikan bahwa Anda mendekode nilai `Signature` dari Base64 sehingga Anda membandingkan nilai-nilai menggunakan format yang sama.
7. Hasilkan nilai hash turunan dari SNS pesan Amazon. Kirim SNS pesan Amazon, dalam format kanonik, ke algoritma hash yang sama yang digunakan untuk menghasilkan tanda tangan.
   a. Jika 1, gunakan SHA1sebagai algoritma hash. `SignatureVersion`
   b. Jika 2, gunakan SHA256sebagai algoritma hash. `SignatureVersion`
8. Hasilkan nilai hash yang ditegaskan dari pesan Amazon. SNS Nilai hash yang ditegaskan adalah hasil dari penggunaan nilai kunci publik (dari langkah 3) untuk mendekripsi tanda tangan yang dikirimkan dengan pesan Amazon. SNS
9. Verifikasi keaslian dan integritas SNS pesan Amazon. Bandingkan nilai hash yang diturunkan (dari langkah 7) dengan nilai hash yang dinyatakan (dari langkah 8). Jika nilainya identik, maka penerima diyakinkan bahwa pesan belum dimodifikasi saat dalam perjalanan dan pesan harus berasal dari AmazonSNS. Jika nilai-nilai tersebut tidak identik, pesan tidak boleh dipercaya oleh penerima.

Mengurai format SNS pesan Amazon

Amazon SNS menggunakan format berikut.

Topik

- HTTP/HTTPSheader
- HTTP/JSONformat konfirmasi HTTPS berlangganan
- HTTP/JSONformat HTTPS pemberitahuan
- HTTP/HTTPSberhenti berlangganan format konfirmasi JSON
- SetSubscriptionAttributesJSONformat kebijakan pengiriman
- SetTopicAttributes JSONformat kebijakan pengiriman

HTTP/HTTPSheader

Saat Amazon SNS mengirimkan konfirmasi langganan, pemberitahuan, atau pesan konfirmasi berhenti berlangganan keHTTP/HTTPStitik akhir, Amazon mengirimkan POST pesan dengan sejumlah nilai header SNS khusus Amazon. Anda dapat menggunakan nilai header untuk tugas seperti mengidentifikasi jenis pesan tanpa harus mengurai badan JSON pesan untuk membaca `Type` nilainya. Secara default, Amazon SNS mengirimkan semua notifikasi ke titik akhir HTTP /S dengan `Content-Type` disetel `ketext/plain; charset=UTF-8`. Untuk memilih `Content-Type` selain teks/polos (default), lihat `headerContentType` di. Membuat kebijakan pengiriman HTTP /S

`x-amz-sns-message-type`

Jenis pesan. Nilai yang mungkin adalah `SubscriptionConfirmation`, `Notification`, dan `UnsubscribeConfirmation`.

`x-amz-sns-message-id`

Universally Unique Identifier (UUID), unik untuk setiap pesan yang diterbitkan. Untuk pemberitahuan bahwa Amazon SNS mengirim ulang selama percobaan ulang, ID pesan dari pesan asli digunakan.

`x-amz-sns-topic-arn`

Nama Sumber Daya Amazon (ARN) untuk topik tempat pesan ini diterbitkan.

`x-amz-sns-subscription-arn`

ARNUntuk berlangganan titik akhir ini.

HTTPPOSTHeader berikut adalah contoh header untuk `Notification` pesan ke HTTP titik akhir.

```
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/JSONformat konfirmasi HTTPS berlangganan

Setelah Anda berlangganan HTTP/HTTPS endpoint, Amazon SNS sends a subscription confirmation message to the HTTP/HTTPS titik akhir. Pesan ini berisi `SubscribeURL` nilai yang harus Anda kunjungi untuk mengonfirmasi langganan (sebagai alternatif, Anda dapat menggunakan `Token` nilainya dengan `ConfirmSubscription`).

> **Note**
>
> Amazon SNS tidak mengirim pemberitahuan ke titik akhir ini hingga langganan dikonfirmasi

Pesan konfirmasi langganan adalah POST pesan dengan badan pesan yang berisi JSON dokumen dengan pasangan nama-nilai berikut.

`Type`

Jenis pesan. Untuk konfirmasi berlangganan, tipenya adalah `SubscriptionConfirmation`.

`MessageId`

Universally Unique Identifier (UUID), unik untuk setiap pesan yang diterbitkan. Untuk pesan yang SNS dikirim ulang Amazon selama percobaan ulang, ID pesan dari pesan asli digunakan.

`Token`

Nilai yang dapat Anda gunakan dengan `ConfirmSubscription`tindakan untuk mengonfirmasi langganan. Atau, Anda dapat mengunjungi `SubscribeURL`.

`TopicArn`

Amazon Resource Name (ARN) untuk topik yang menjadi langganan endpoint ini.

`Message`

String yang menggambarkan pesan. Untuk konfirmasi berlangganan, string ini terlihat seperti ini:

```
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`

URLYang harus Anda kunjungi untuk mengonfirmasi langganan. Sebagai alternatif, Anda dapat menggunakan `ConfirmSubscription`tindakan `Token` dengan untuk mengonfirmasi langganan.

`Timestamp`

Waktu (GMT) ketika konfirmasi berlangganan dikirim.

`SignatureVersion`

Versi SNS tanda tangan Amazon yang digunakan.

- Jika `SignatureVersion` adalah 1, `Signature` adalah `SHA1withRSA` tanda tangan yang dikodekan Base64 dari`Message`,,, `MessageIdType`, `Timestamp` dan nilai. `TopicArn`
- Jika `SignatureVersion` adalah 2, `Signature` adalah `SHA256withRSA` tanda tangan yang dikodekan Base64 dari`Message`,,, `MessageIdType`, `Timestamp` dan nilai. `TopicArn`

`Signature`

Base64 dikodekan `SHA1withRSA` atau `SHA256withRSA` tanda tangan dari`Message`,,, `MessageIdType`, `Timestamp` dan nilai. `TopicArn`

`SigningCertURL`

URLKe sertifikat yang digunakan untuk menandatangani pesan.

HTTP`POST`Pesan berikut adalah contoh `SubscriptionConfirmation` pesan ke HTTP titik akhir.

```
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/JSONformat HTTPS pemberitahuan

Saat Amazon SNS mengirimkan notifikasi ke langganan HTTP atau HTTPS titik akhir, POST pesan yang dikirim ke titik akhir memiliki badan pesan yang berisi JSON dokumen dengan pasangan nama-nilai berikut.

`Type`

Jenis pesan. Untuk notifikasi, jenisnya adalah `Notification`.

`MessageId`

Universally Unique Identifier (UUID), unik untuk setiap pesan yang diterbitkan. Untuk pemberitahuan bahwa Amazon SNS mengirim ulang selama percobaan ulang, ID pesan dari pesan asli digunakan.

`TopicArn`

Nama Sumber Daya Amazon (ARN) untuk topik tempat pesan ini diterbitkan.

`Subject`

`Subject`Parameter yang ditentukan saat pemberitahuan dipublikasikan ke topik.

> **Note**
>
> Ini adalah parameter opsional. Jika tidak `Subject` ditentukan, maka pasangan nama-nilai ini tidak muncul dalam dokumen iniJSON.

`Message`

`Message`Nilai yang ditentukan saat pemberitahuan dipublikasikan ke topik.

`Timestamp`

Waktu (GMT) ketika pemberitahuan dipublikasikan.

`SignatureVersion`

Versi SNS tanda tangan Amazon yang digunakan.

- Jika `SignatureVersion` adalah 1, `Signature` adalah `SHA1withRSA` tanda tangan yang dikodekan Base64 dari`Message`,, `Subject` (jika ada)`MessageId`,,`Type`, `Timestamp` dan nilai. `TopicArn`
- Jika `SignatureVersion` adalah 2, `Signature` adalah `SHA256withRSA` tanda tangan yang dikodekan Base64 dari`Message`,`MessageId`, `Subject` (jika ada),`Type`, `Timestamp` dan nilai. `TopicArn`

`Signature`

Base64 dikodekan `SHA1withRSA` atau `SHA256withRSA` tanda tangan dari`Message`,`MessageId`, `Subject` (jika ada),,`Type`, `Timestamp` dan nilai. `TopicArn`

`SigningCertURL`

URLKe sertifikat yang digunakan untuk menandatangani pesan.

`UnsubscribeURL`

A URL yang dapat Anda gunakan untuk berhenti berlangganan titik akhir dari topik ini. Jika Anda mengunjungi iniURL, Amazon berhenti SNS berlangganan titik akhir dan berhenti mengirim pemberitahuan ke titik akhir ini.

HTTPPOSTPesan berikut adalah contoh `Notification` pesan ke HTTP titik akhir.

```
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/HTTPSberhenti berlangganan format konfirmasi JSON

Setelah HTTPS titik akhirHTTP/berhenti berlangganan dari suatu topik, Amazon SNS mengirimkan pesan konfirmasi berhenti berlangganan ke titik akhir.

Pesan konfirmasi berhenti berlangganan adalah POST pesan dengan badan pesan yang berisi JSON dokumen dengan pasangan nama-nilai berikut.

`Type`

Jenis pesan. Untuk konfirmasi berhenti berlangganan, jenisnya adalah `UnsubscribeConfirmation`.

`MessageId`

Universally Unique Identifier (UUID), unik untuk setiap pesan yang diterbitkan. Untuk pesan yang SNS dikirim ulang Amazon selama percobaan ulang, ID pesan dari pesan asli digunakan.

`Token`

Nilai yang dapat Anda gunakan dengan `ConfirmSubscription`tindakan untuk mengonfirmasi ulang langganan. Atau, Anda dapat mengunjungi `SubscribeURL`.

`TopicArn`

Amazon Resource Name (ARN) untuk topik tempat titik akhir ini berhenti berlangganan.

`Message`

String yang menggambarkan pesan. Untuk konfirmasi berhenti berlangganan, string ini terlihat seperti ini:

```
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`

URLYang harus Anda kunjungi untuk mengonfirmasi ulang langganan. Sebagai alternatif, Anda dapat menggunakan `ConfirmSubscription`tindakan `Token` dengan untuk mengonfirmasi ulang langganan.

`Timestamp`

Waktu (GMT) ketika konfirmasi berhenti berlangganan dikirim.

`SignatureVersion`

Versi SNS tanda tangan Amazon yang digunakan.

- Jika `SignatureVersion` adalah 1, `Signature` adalah `SHA1withRSA` tanda tangan yang dikodekan Base64 dari`Message`,,, `MessageIdType`, `Timestamp` dan nilai. `TopicArn`
- Jika `SignatureVersion` adalah 2, `Signature` adalah `SHA256withRSA` tanda tangan yang dikodekan Base64 dari`Message`,,, `MessageIdType`, `Timestamp` dan nilai. `TopicArn`

`Signature`

Base64 dikodekan `SHA1withRSA` atau `SHA256withRSA` tanda tangan dari`Message`,,, `MessageIdType`, `Timestamp` dan nilai. `TopicArn`

`SigningCertURL`

URLKe sertifikat yang digunakan untuk menandatangani pesan.

HTTPPOSTPesan berikut adalah contoh `UnsubscribeConfirmation` pesan ke HTTP titik akhir.

```
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"
}
```

SetSubscriptionAttributesJSONformat kebijakan pengiriman

Jika Anda mengirim permintaan ke `SetSubscriptionAttributes` tindakan dan mengatur `AttributeName` parameter ke nilai`DeliveryPolicy`, nilai `AttributeValue` parameter harus JSON objek yang valid. Sebagai contoh, contoh berikut menetapkan kebijakan pengiriman untuk 5 jumlah pengulangan.

```
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}}
...
```

Gunakan JSON format berikut untuk nilai `AttributeValue` parameter.

```
{
    "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"
    }
}
```

Untuk informasi lebih lanjut tentang `SetSubscriptionAttribute` tindakan tersebut, buka SetSubscriptionAttributesdi APIReferensi Layanan Pemberitahuan Sederhana Amazon. Untuk informasi selengkapnya tentang header HTTP tipe konten yang didukung, lihat. Membuat kebijakan pengiriman HTTP /S

SetTopicAttributes JSONformat kebijakan pengiriman

Jika Anda mengirim permintaan ke `SetTopicAttributes` tindakan dan mengatur `AttributeName` parameter ke nilai`DeliveryPolicy`, nilai `AttributeValue` parameter harus JSON objek yang valid. Sebagai contoh, contoh berikut menetapkan kebijakan pengiriman untuk 5 jumlah pengulangan.

```
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}}}
```

```
...
```

Gunakan JSON format berikut untuk nilai `AttributeValue` parameter.

```
{
    "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"
        }
    }
}
```

Untuk informasi lebih lanjut tentang `SetTopicAttribute` tindakan tersebut, buka SetTopicAttributesdi APIReferensi Layanan Pemberitahuan Sederhana Amazon. Untuk informasi selengkapnya tentang header HTTP tipe konten yang didukung, lihat. Membuat kebijakan pengiriman HTTP /S

Acara Fanout Amazon ke SNS AWS Event Fork Pipelines

Untuk pengarsipan dan analitik acara, Amazon SNS sekarang merekomendasikan untuk menggunakan integrasi aslinya dengan Amazon Data Firehose. Anda dapat berlangganan aliran pengiriman Firehose ke SNS topik, yang memungkinkan Anda mengirim pemberitahuan ke titik akhir pengarsipan dan analitik seperti bucket Amazon Simple Storage Service (Amazon S3), tabel Amazon Redshift, Amazon Service (Service), dan banyak lagi OpenSearch . OpenSearc h Menggunakan Amazon SNS dengan aliran pengiriman Firehose adalah solusi yang dikelola sepenuhnya dan tanpa kode yang tidak mengharuskan Anda menggunakan fungsi. AWS Lambda Untuk informasi selengkapnya, lihat Aliran pengiriman Fanout ke Firehose.

Anda dapat menggunakan Amazon SNS untuk membangun aplikasi berbasis peristiwa yang menggunakan layanan pelanggan untuk melakukan pekerjaan secara otomatis sebagai respons terhadap peristiwa yang dipicu oleh layanan penayang. Pola arsitektur ini dapat membuat layanan lebih dapat digunakan kembali, dapat dioperasikan, dan dapat diskalakan. Namun demikian, pola tersebut dapat menjadi padat karya untuk fork pemrosesan peristiwa ke dalam alur yang mengatasi persyaratan penanganan peristiwa umum, seperti penyimpanan peristiwa, cadangan, pencarian, analitik, dan ulangan.

Untuk mempercepat pengembangan aplikasi berbasis peristiwa, Anda dapat berlangganan pipeline penanganan acara—yang didukung oleh pipa Event Fork— ke topik Amazon. AWS SNS AWS Event Fork Pipelines adalah rangkaian aplikasi bersarang sumber terbuka, berdasarkan Model Aplikasi AWS Tanpa Server (AWS SAM), yang dapat Anda gunakan langsung dari rangkaian AWS Event Fork Pipelines (pilih Tampilkan aplikasi yang membuat IAM peran khusus atau kebijakan sumber daya) ke akun Anda. AWS

Untuk kasus penggunaan AWS Event Fork Pipelines, lihatMenyebarkan dan menguji aplikasi sampel pipeline fork SNS event Amazon.

Topik

- Bagaimana AWS Event Fork Pipelines bekerja
- Menyebarkan Pipa Garpu AWS Acara
- Menyebarkan dan menguji aplikasi sampel pipeline fork SNS event Amazon
- Berlangganan AWS Event Fork Pipelines ke topik Amazon SNS

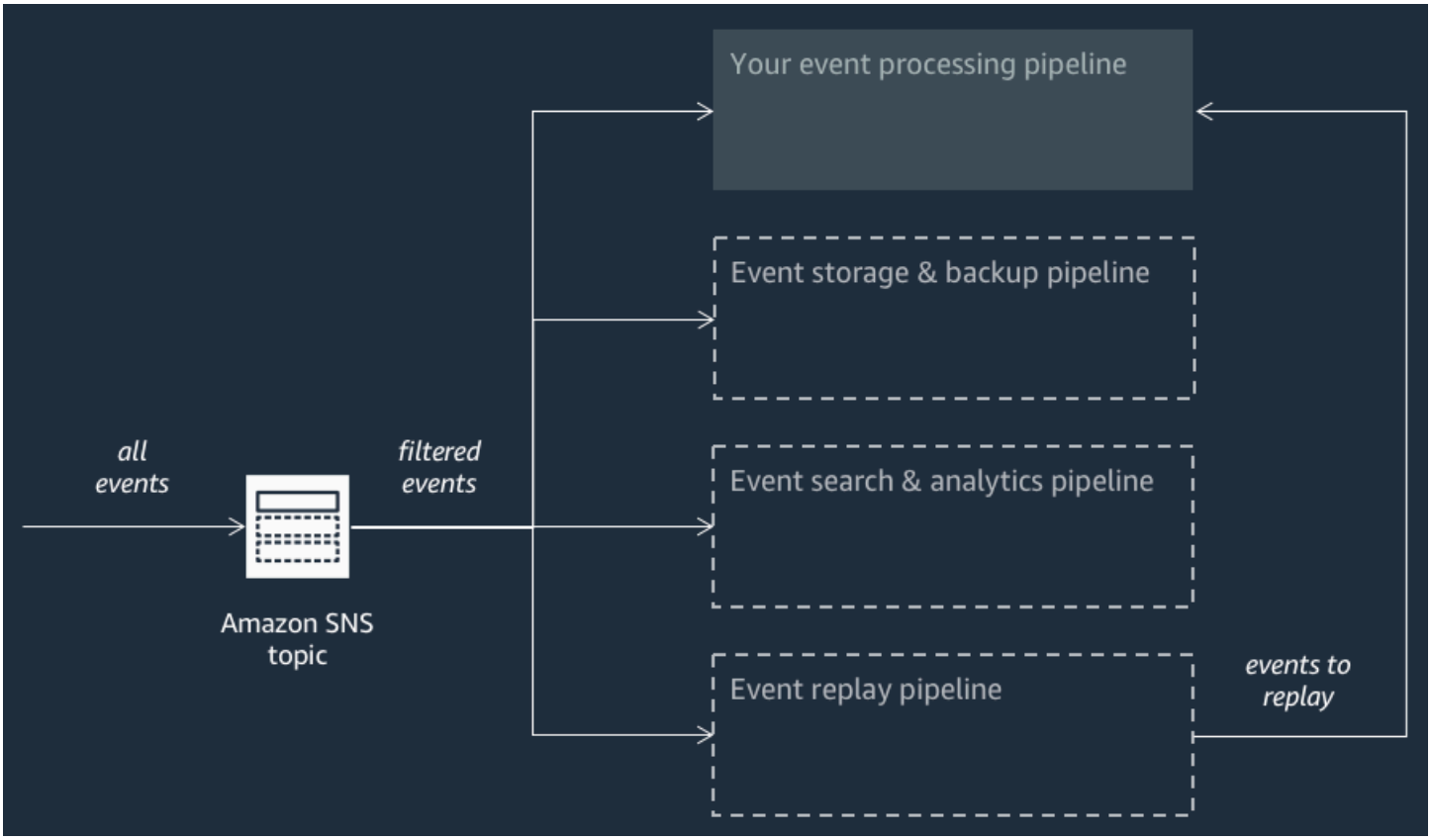
Bagaimana AWS Event Fork Pipelines bekerja

AWS Event Fork Pipelines adalah pola desain tanpa server. Namun, ini juga merupakan rangkaian aplikasi tanpa server bersarang berdasarkan AWS SAM (yang dapat Anda terapkan langsung dari AWS Serverless Application Repository (AWS SAR) ke Anda untuk memperkaya platform berbasis peristiwa Anda Akun AWS). Anda dapat men-deploy aplikasi bersarang tersebut secara individual, sesuai kebutuhan arsitektur Anda.

Topik

- Penyimpanan peristiwa dan alur cadangan
- Pencarian peristiwa dan alur analitik
- Alur ulangan peristiwa

Diagram berikut menunjukkan aplikasi AWS Event Fork Pipelines yang dilengkapi dengan tiga aplikasi bersarang. Anda dapat menyebarkan saluran pipa apa pun dari rangkaian AWS Event Fork Pipelines AWS SAR secara independen, sesuai kebutuhan arsitektur Anda.



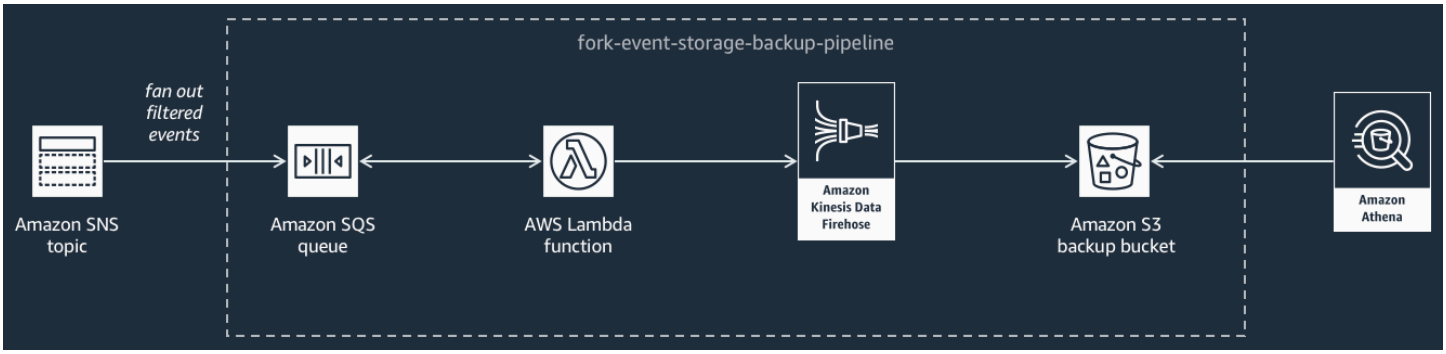
Setiap pipeline berlangganan SNS topik Amazon yang sama, memungkinkan dirinya untuk memproses peristiwa secara paralel karena peristiwa ini dipublikasikan ke topik tersebut. Setiap alur bersifat independen dan dapat mengatur Kebijakan Filter Berlangganan miliknya sendiri. Hal ini mengizinkan alur untuk memproses hanya subset dari peristiwa yang menjadi perhatian (bukan semua peristiwa yang diterbitkan untuk topik).

> **Note**
>
> Karena Anda menempatkan tiga AWS Event Fork Pipelines di samping pipeline pemrosesan acara reguler Anda (mungkin sudah berlangganan SNS topik Amazon Anda), Anda tidak perlu mengubah bagian mana pun dari penerbit pesan Anda saat ini untuk memanfaatkan Pipelines Fork AWS Event di beban kerja yang ada.

Penyimpanan peristiwa dan alur cadangan

Diagram berikut menunjukkan Penyimpanan Peristiwa dan Alur Cadangan. Anda dapat berlangganan pipeline ini ke SNS topik Amazon Anda untuk secara otomatis mencadangkan peristiwa yang mengalir melalui sistem Anda.

Pipeline ini terdiri dari SQS antrian Amazon yang menyangga peristiwa yang dikirimkan oleh SNS topik Amazon, AWS Lambda fungsi yang secara otomatis melakukan polling untuk peristiwa ini dalam antrian dan mendorongnya ke aliran Amazon Data Firehose, dan bucket Amazon S3 yang secara tahan lama mencadangkan peristiwa yang dimuat oleh aliran.

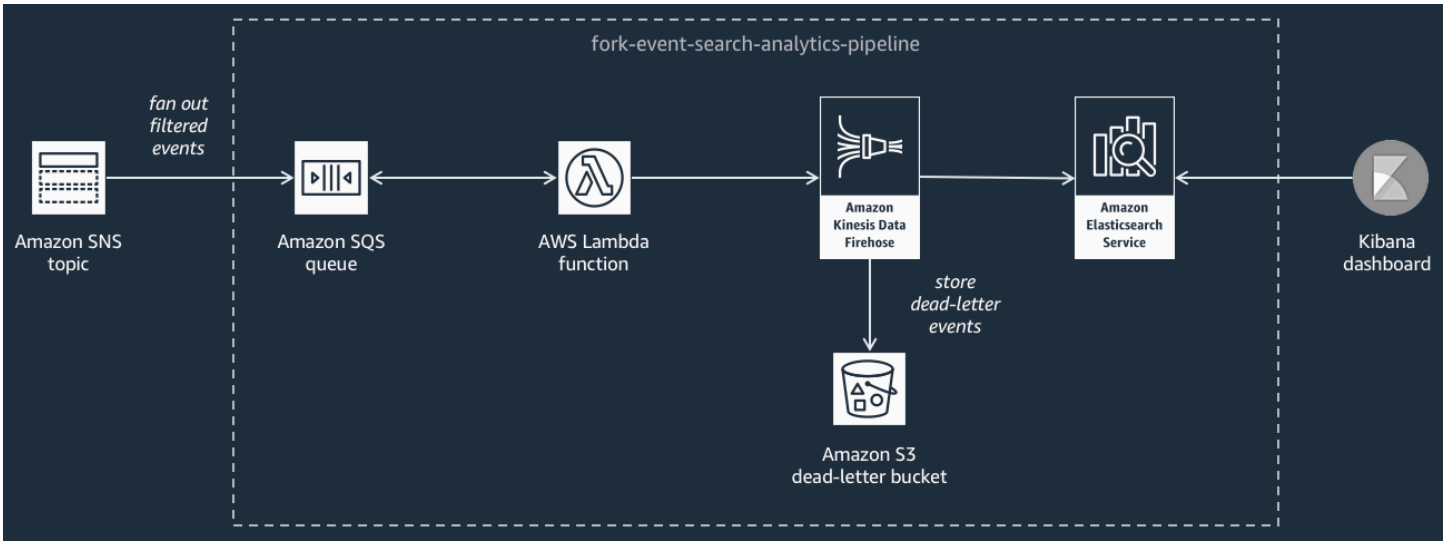


Untuk penyetelan halus perilaku pengaliran Firehose Anda, Anda dapat mengonfigurasinya untuk membuffer, mengubah, dan mengompres peristiwa Anda sebelum memuatnya ke dalam bucket. Saat acara dimuat, Anda dapat menggunakan Amazon Athena untuk menanyakan bucket menggunakan kueri standarSQL. Anda juga dapat mengonfigurasi alur untuk menggunakan kembali bucket Amazon S3 yang ada atau membuat yang baru.

Pencarian peristiwa dan alur analitik

Diagram berikut ini menunjukkan Pencarian Peristiwa dan Alur Analitik. Anda dapat berlangganan pipeline ini ke SNS topik Amazon Anda untuk mengindeks peristiwa yang mengalir melalui sistem Anda di domain penelusuran dan kemudian menjalankan analitik pada mereka.

Pipeline ini terdiri dari SQS antrian Amazon yang menyangga peristiwa yang disampaikan oleh SNS topik Amazon, AWS Lambda fungsi yang melakukan polling peristiwa dari antrian dan mendorongnya ke aliran Amazon Data Firehose, domain OpenSearch Layanan Amazon yang mengindeks peristiwa yang dimuat oleh aliran Firehose, dan bucket Amazon S3 yang menyimpan peristiwa mati yang tidak dapat diindeks dalam pencarian domain.



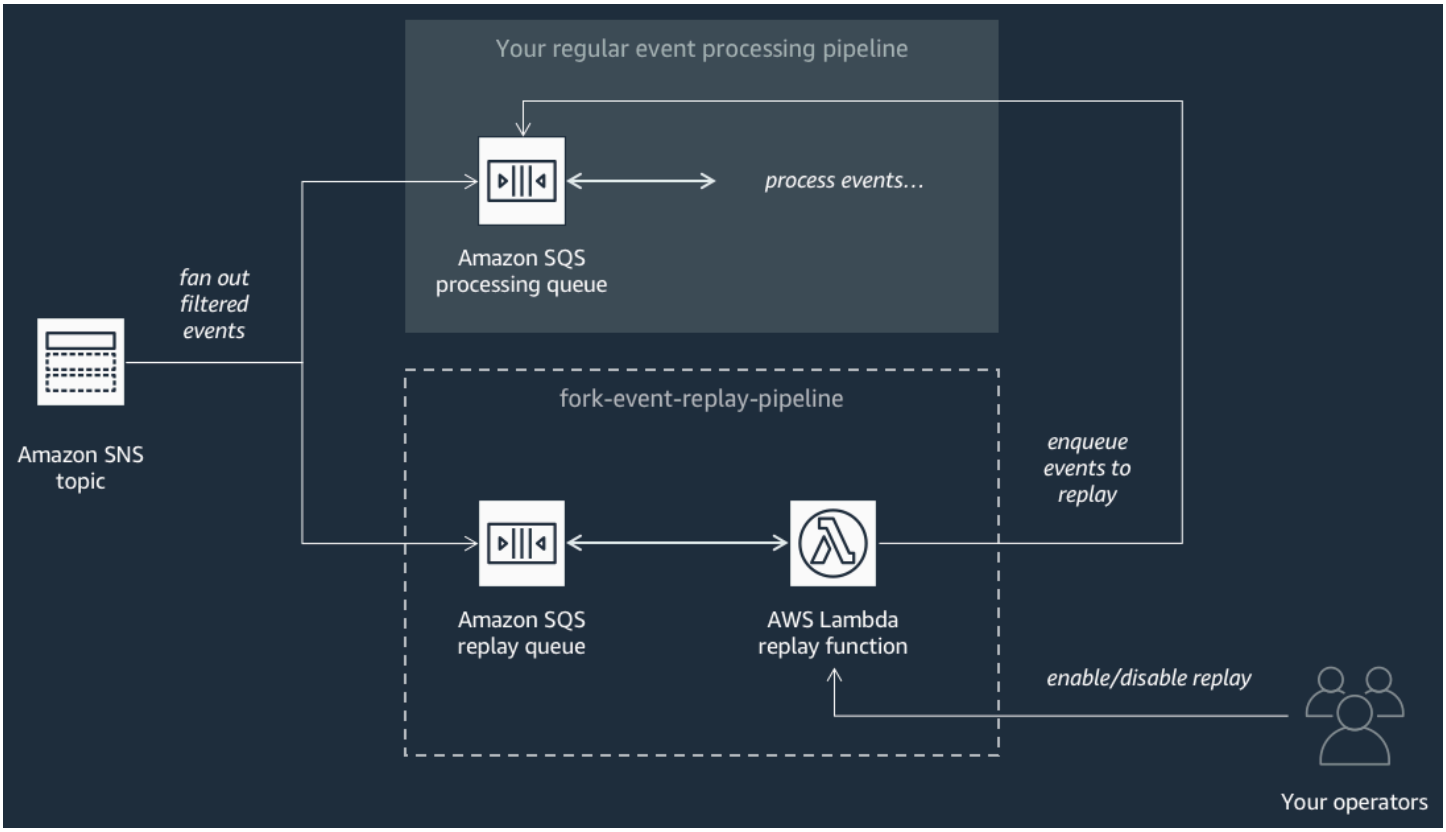
Untuk penyetelan halus pengaliran Firehose Anda dalam syarat buffering peristiwa, transformasi, dan kompresi, Anda dapat mengonfigurasi alur ini.

Anda juga dapat mengonfigurasi apakah pipeline harus menggunakan kembali domain yang ada di OpenSearch domain Anda Akun AWS atau membuat yang baru untuk Anda. Saat peristiwa diindeks dalam domain pencarian, Anda dapat menggunakan Kibana untuk menjalankan analitik pada peristiwa Anda dan memperbarui dasbor visual secara waktu nyata.

Alur ulangan peristiwa

Diagram berikut ini menunjukkan Alur Ulangan Peristiwa. Untuk merekam peristiwa yang telah diproses oleh sistem Anda selama 14 hari terakhir (misalnya ketika platform Anda perlu pulih dari kegagalan), Anda dapat berlangganan pipeline ini ke SNS topik Amazon Anda dan kemudian memproses ulang acara.

Pipeline ini terdiri dari SQS antrean Amazon yang menyangga peristiwa yang dikirimkan oleh SNS topik Amazon, dan AWS Lambda fungsi yang melakukan polling peristiwa dari antrian dan memindahkannya kembali ke pipeline pemrosesan acara reguler Anda, yang juga berlangganan topik Anda.



> **Note**
>
> Secara default, fungsi ulangan dinonaktifkan, tidak memindahkan kembali peristiwa Anda. Jika Anda perlu memproses ulang peristiwa, Anda harus mengaktifkan antrean SQS pemutaran ulang Amazon sebagai sumber peristiwa untuk fungsi AWS Lambda pemutaran ulang.

Menyebarkan Pipa Garpu AWS Acara

Suite AWS Event Fork Pipelines (pilih Tampilkan aplikasi yang membuat IAM peran kustom atau kebijakan sumber daya) tersedia sebagai grup aplikasi publik di AWS Serverless Application Repository, tempat Anda dapat menerapkan dan mengujinya secara manual menggunakan konsol.AWS Lambda Untuk informasi tentang penerapan pipeline menggunakan AWS Lambda konsol, lihat. Berlangganan AWS Event Fork Pipelines ke topik Amazon SNS

Dalam skenario produksi, kami sarankan untuk menyematkan AWS Event Fork Pipelines dalam template aplikasi Anda secara keseluruhan. AWS SAM Fitur aplikasi bersarang memungkinkan Anda melakukan ini dengan menambahkan sumber daya `AWS::Serverless::Application`

ke AWS SAM template Anda, mereferensikan AWS SAR `ApplicationId` dan aplikasi bersarang`SemanticVersion`.

Misalnya, Anda dapat menggunakan Event Storage dan Backup Pipeline sebagai aplikasi bersarang dengan menambahkan YAML cuplikan berikut ke `Resources` bagian template Anda. AWS SAM

```
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
```

Saat Anda menentukan nilai parameter, Anda dapat menggunakan fungsi AWS CloudFormation intrinsik untuk mereferensikan sumber daya lain di template Anda. Misalnya, dalam YAML cuplikan di atas, `TopicArn` parameter mereferensikan `AWS::SNS::Topic` sumber daya`MySNSTopic`, yang didefinisikan di tempat lain dalam template. AWS SAM Untuk informasi selengkapnya, lihat Referensi Fungsi Intrinsik di Panduan Pengguna AWS CloudFormation .

> **Note**
>
> Halaman AWS Lambda konsol untuk AWS SAR aplikasi Anda menyertakan tombol Salin sebagai SAM Sumber Daya, yang menyalin yang YAML diperlukan untuk menyarangkan AWS SAR aplikasi ke clipboard.

Menyebarkan dan menguji aplikasi sampel pipeline fork SNS event Amazon

Untuk mempercepat pengembangan aplikasi berbasis peristiwa, Anda dapat berlangganan pipeline penanganan acara—yang didukung oleh pipa Event Fork— ke topik Amazon. AWS SNS AWS Event Fork Pipelines adalah rangkaian aplikasi bersarang sumber terbuka, berdasarkan Model Aplikasi AWS Tanpa Server (AWS SAM), yang dapat Anda gunakan langsung dari rangkaian AWS Event Fork Pipelines (pilih Tampilkan aplikasi yang membuat IAM peran khusus atau kebijakan sumber daya) ke akun Anda. AWS Untuk informasi selengkapnya, lihat Bagaimana AWS Event Fork Pipelines bekerja.

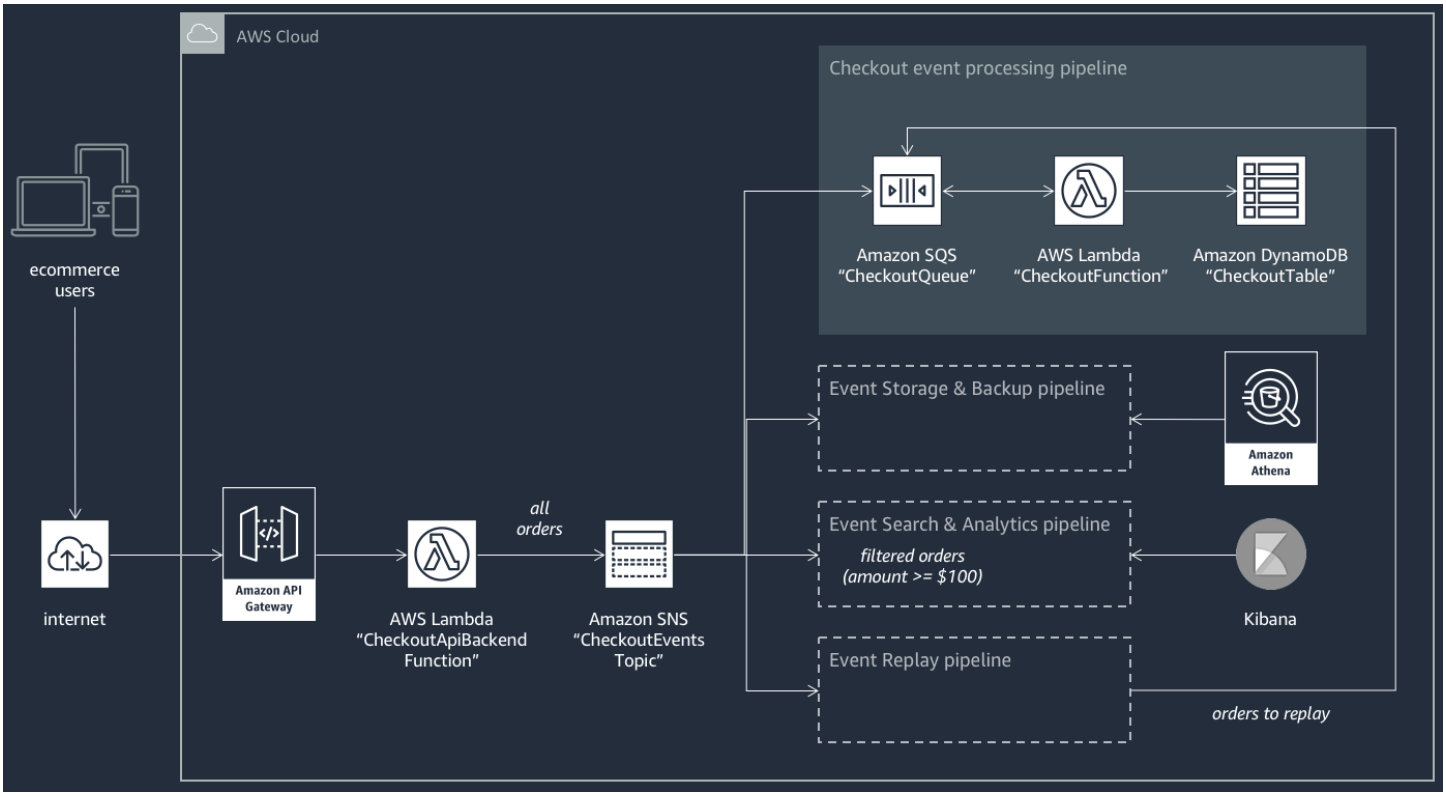
Halaman ini menunjukkan bagaimana Anda dapat menggunakan AWS Management Console untuk menyebarkan dan menguji aplikasi sampel AWS Event Fork Pipelines.

> **Important**
>
> Untuk menghindari biaya yang tidak diinginkan setelah Anda selesai menerapkan aplikasi sampel AWS Event Fork Pipelines, hapus tumpukannya. AWS CloudFormation Untuk informasi selengkapnya, lihat Menghapus Tumpukan pada Konsol AWS CloudFormation di Panduan Pengguna AWS CloudFormation .

Topik

AWS Contoh kasus penggunaan Event Fork Pipelines

Skenario berikut menjelaskan aplikasi e-commerce tanpa server berbasis peristiwa yang menggunakan AWS Event Fork Pipelines. Anda dapat menggunakan contoh aplikasi e-commerce ini di AWS Serverless Application Repository dan kemudian menyebarkannya di AWS Lambda konsol Anda Akun AWS , di mana Anda dapat mengujinya dan memeriksa kode sumbernya. GitHub



Aplikasi e-commerce ini menerima pesanan dari pembeli melalui RESTful API host oleh API Gateway dan didukung oleh AWS Lambda fungsi tersebut`CheckoutApiBackendFunction`. Fungsi ini menerbitkan semua pesanan yang diterima ke SNS topik Amazon bernama `CheckoutEventsTopic` yang, pada gilirannya, mengirimkan pesanan ke empat saluran pipa yang berbeda.

Alur pertama adalah alur pemrosesan checkout reguler yang dirancang dan diimplementasikan oleh pemilik aplikasi perdagangan elektronik. Pipeline ini memiliki SQS antrian Amazon `CheckoutQueue` yang menyangga semua pesanan yang diterima, AWS Lambda fungsi bernama `CheckoutFunction` yang melakukan polling antrian untuk memproses pesanan ini, dan tabel DynamoDB yang menyimpan semua pesanan yang ditempatkan dengan aman. `CheckoutTable`

Menerapkan Pipa Garpu AWS Acara

Komponen dari aplikasi perdagangan elektronik menangani logika bisnis inti. Namun demikian, pemilik aplikasi perdagangan elektronik juga perlu mengatasi hal berikut:

- Kepatuhan—cadangan yang aman dan terkompresi yang dienkripsi saat tidak bergerak dan sanitasi informasi sensitif
- Ketahanan—ulangan pesanan terbaru dalam kasus terganggunya proses pemenuhan

- Ketertelusuran—menjalankan analitik dan membuat metrik pada pesanan yang dibuat

Alih-alih menerapkan logika pemrosesan peristiwa ini, pemilik aplikasi dapat berlangganan AWS Event Fork Pipelines ke topik `CheckoutEventsTopic` Amazon SNS

- Penyimpanan peristiwa dan alur cadangandikonfigurasi untuk mengubah data untuk menghapus detail kartu kredit, menyangga data selama 60 detik, mengompresnya menggunakanGZIP, dan mengenkripsi menggunakan kunci terkelola pelanggan default untuk Amazon S3. Kunci ini dikelola oleh AWS dan didukung oleh AWS Key Management Service (AWS KMS).

  Untuk informasi selengkapnya, lihat Memilih Amazon S3 Untuk Tujuan Anda, Transformasi Data Firehose Data Amazon, dan Mengonfigurasi Pengaturan di Panduan Pengembang Amazon Data Firehose.
- Pencarian peristiwa dan alur analitik dikonfigurasi dengan mengindeks durasi coba lagi 30 detik, bucket untuk menyimpan pesanan yang gagal untuk diindekskan di domain pencarian, dan kebijakan filter untuk membatasi set pesanan yang diindeks.

  Untuk informasi selengkapnya, lihat Memilih OpenSearch Layanan untuk Tujuan Anda di Panduan Pengembang Amazon Data Firehose.
- Alur ulangan peristiwadikonfigurasi dengan bagian SQS antrian Amazon dari pipeline pemrosesan pesanan reguler yang dirancang dan diimplementasikan oleh pemilik aplikasi e-commerce.

  Untuk informasi selengkapnya, lihat Nama Antrian dan URL di Panduan Pengembang Layanan Antrian Sederhana Amazon.

Kebijakan JSON filter berikut diatur dalam konfigurasi untuk Event Search dan Analytics Pipeline. Ini hanya akan mencocokkan pesanan yang masuk di mana jumlah total adalah $100 atau lebih tinggi. Untuk informasi selengkapnya, lihat SNSPemfilteran pesan Amazon.

```
{
   "amount": [{ "numeric": [ ">=", 100 ] }]
}
```

Dengan menggunakan pola AWS Event Fork Pipelines, pemilik aplikasi e-commerce dapat menghindari overhead pengembangan yang sering mengikuti pengkodean logika undifferentiating untuk penanganan event. Sebagai gantinya, dia bisa menyebarkan AWS Event Fork Pipelines langsung dari AWS Serverless Application Repository ke dalam dirinya. Akun AWS

Langkah 1: Menyebarkan contoh aplikasi Amazon SNS

1. Masuk ke konsol AWS Lambda tersebut.
2. Pada panel navigasi, pilih Fungsi dan kemudian pilih Buat fungsi.
3. Pada halaman Buat fungsi, lakukan hal berikut ini:
   a. Pilih Jelajahi repositori aplikasi tanpa server, Aplikasi publik, Tampilkan aplikasi yang membuat IAM peran khusus atau kebijakan sumber daya.
   b. Cari untuk `fork-example-ecommerce-checkout-api` dan kemudian pilih aplikasi.
4. Pada halaman fork-example-ecommerce-checkout-api, lakukan hal berikut:
   a. Di bagian Pengaturan aplikasi, masukkan Nama aplikasi(sebagai contoh, `fork-example-ecommerce-my-app`).

      > **Note**
      > - Untuk menemukan sumber daya Anda dengan mudah nanti, simpan prefiks `fork-example-ecommerce`.
      > - Untuk setiap deployment, nama aplikasi harus unik. Jika Anda menggunakan kembali nama aplikasi, penerapan hanya akan memperbarui AWS CloudFormation tumpukan yang digunakan sebelumnya (bukan membuat yang baru).

   b. (Opsional) Masukkan salah satu LogLevelpengaturan berikut untuk eksekusi fungsi Lambda aplikasi Anda:
      - `DEBUG`
      - `ERROR`
      - `INFO` (default)
      - `WARNING`
5. Pilih Saya mengakui bahwa aplikasi ini membuat IAM peran khusus, kebijakan sumber daya, dan menerapkan aplikasi bersarang. dan kemudian, di bagian bawah halaman, pilih Deploy.

Pada status Deployment untuk fork-example-ecommerce -***my-app***halaman, Lambda menampilkan status Aplikasi Anda sedang digunakan.

Di bagian Sumber Daya, AWS CloudFormation mulailah membuat tumpukan dan menampilkan PROGRESS status CREATE_IN_ untuk setiap sumber daya. Ketika proses selesai, AWS CloudFormation menampilkan COMPLETE status CREATE_.

> **Note**
>
> Mungkin perlu waktu 20-30 menit bagi semua sumber daya untuk di-deploy.

Setelah deployment selesai, Lambda menampilkan status Aplikasi Anda telah di-deploy.

Langkah 2: Menjalankan aplikasi sampel SNS -linked

1. Di AWS Lambda konsol, pada panel navigasi, pilih Aplikasi.
2. Pada halaman Aplikasi, di bidang pencarian, cari untuk `serverlessrepo-fork-example-ecommerce-`*`my-app`* dan kemudian pilih aplikasi.
3. Di bagian Sumber Daya, lakukan hal berikut ini:

   a. Untuk menemukan sumber daya yang tipenya ApiGatewayRestApi, urutkan sumber daya berdasarkan Jenis, misalnya`ServerlessRestApi`, lalu perluas sumber daya.

   b. Dua sumber daya bersarang ditampilkan, dari jenis ApiGatewayDeployment dan ApiGateway Stage.

   c. Salin tautan Prod API endpoint dan tambahkan `/checkout` ke dalamnya, misalnya:

   ```
   https://abcdefghij.execute-api.us-east-2.amazonaws.com/Prod/checkout
   ```

4. Salin berikut ini JSON ke file bernama`test_event.json`.

   ```
   {
      "id": 15311,
      "date": "2019-03-25T23:41:11-08:00",
      "status": "confirmed",
      "customer": {
         "id": 65144,
     "quantity": 2,
         "price": 25.00,
         "subtotal": 50.00
      }]
   }
   ```

5. Untuk mengirim HTTPS permintaan ke API titik akhir Anda, teruskan payload peristiwa sampel sebagai input dengan menjalankan `curl` perintah, misalnya:

```
curl -d "$(cat test_event.json)" https://abcdefghij.execute-api.us-
east-2.amazonaws.com/Prod/checkout
```

APIMengembalikan respon kosong berikut, menunjukkan eksekusi sukses:

```
{ }
```

Langkah 3: Memverifikasi SNS aplikasi Amazon dan kinerja pipeline

Langkah 1: Memverifikasi eksekusi pipeline checkout sampel

1. Masuk ke konsol Amazon DynamoDB.
2. Pada panel navigasi, pilih Tabel.
3. Cari untuk `serverlessrepo-fork-example` dan pilih `CheckoutTable`.
4. Pada halaman detail tabel, pilih Item dan kemudian pilih item yang dibuat.

   Atribut yang tersimpan akan ditampilkan.

Langkah 2: Memverifikasi eksekusi penyimpanan acara dan pipa cadangan

1. Masuk ke konsol Amazon S3.
2. Pada panel navigasi, pilih Bucket.
3. Cari untuk `serverlessrepo-fork-example` dan kemudian pilih `CheckoutBucket`.
4. Navigasikan hierarki direktori hingga Anda menemukan file dengan ekstensi `.gz`.
5. Untuk mengunduh file, pilih Tindakan, Buka.
6. Alur dikonfigurasi dengan fungsi Lambda yang membersihkan informasi kartu kredit untuk alasan kepatuhan.

   Untuk memverifikasi bahwa JSON payload yang disimpan tidak berisi informasi kartu kredit, dekompresi file.

Langkah 3: Memverifikasi eksekusi pencarian acara dan pipeline analitik

1. Masuk ke konsol OpenSearch Layanan.
2. Pada panel navigasi, di bawah Domain saya, pilih domain dengan prefiks `serverl-analyt`.
3. Pipeline dikonfigurasi dengan kebijakan filter SNS langganan Amazon yang menetapkan kondisi pencocokan numerik.

   Untuk memverifikasi bahwa acara diindeks karena mengacu pada pesanan yang nilainya lebih tinggi dari USD $100, pada analitik server- ***abcdefgh1ijk***halaman, pilih Indeks, checkout_events.

Langkah 4: Memverifikasi eksekusi pipeline replay acara

1. Masuk ke SQSkonsol Amazon.
2. Dalam daftar antrean, cari untuk `serverlessrepo-fork-example` dan pilih `ReplayQueue`.
3. Pilih Kirim dan terima pesan.
4. Dalam Kirim dan terima pesan di fork-example-ecommerce -***my-app***... ReplayP- - ReplayQueue***123ABCD4E5F6***kotak dialog, pilih Poll untuk pesan.
5. Untuk memverifikasi bahwa peristiwa diantrekan, pilih Detail Selengkapnya di samping pesan yang muncul di antrean.

Langkah 4: Mensimulasikan masalah dan memutar ulang acara untuk pemulihan

Langkah 1: Aktifkan masalah simulasi dan kirim permintaan kedua API

1. Masuk ke konsol AWS Lambda tersebut.
2. Pada panel navigasi, pilih Fungsi.
3. Cari untuk `serverlessrepo-fork-example` dan pilih `CheckoutFunction`.
4. Pada fork-example-ecommerce -***my-app***-CheckoutFunction-***ABCDEF***... page, di bagian Environment variables, atur ENABLED variabel BUG_ ke true lalu pilih Save.
5. Salin berikut ini JSON ke file bernama`test_event_2.json`.

```
{
    "id": 9917,
    "date": "2019-03-26T21:11:10-08:00",
    "status": "confirmed",
```

```
    "customer": {
       "id": 56999,
"quantity": 1,
       "price": 75.00,
       "subtotal": 75.00
    }]
 }
```

6. Untuk mengirim HTTPS permintaan ke API titik akhir Anda, teruskan payload peristiwa sampel sebagai input dengan menjalankan `curl` perintah, misalnya:

```
curl -d "$(cat test_event_2.json)" https://abcdefghij.execute-api.us-
east-2.amazonaws.com/Prod/checkout
```

APIMengembalikan respon kosong berikut, menunjukkan eksekusi sukses:

```
{ }
```

Langkah 2: Verifikasi korupsi data simulasi

1. Masuk ke konsol Amazon DynamoDB.
2. Pada panel navigasi, pilih Tabel.
3. Cari untuk `serverlessrepo-fork-example` dan pilih `CheckoutTable`.
4. Pada halaman detail tabel, pilih Item dan kemudian pilih item yang dibuat.

   Atribut yang disimpan ditampilkan, beberapa ditandai sebagai CORRUPTED!

Langkah 3: Nonaktifkan masalah simulasi

1. Masuk ke konsol AWS Lambda tersebut.
2. Pada panel navigasi, pilih Fungsi.
3. Cari untuk `serverlessrepo-fork-example` dan pilih `CheckoutFunction`.
4. Pada fork-example-ecommerce -***my-app***-CheckoutFunction-***ABCDEF***... halaman, di bagian variabel Lingkungan, atur ENABLED variabel BUG_ ke false dan kemudian pilih Simpan.

Langkah 4: Aktifkan replay untuk memulihkan dari masalah

1. Di AWS Lambda konsol, pada panel navigasi, pilih Fungsi.
2. Cari untuk `serverlessrepo-fork-example` dan pilih `ReplayFunction`.
3. Perluas bagian Desainer, pilih SQSubin dan kemudian, di SQSbagian, pilih Diaktifkan.

> **Note**
>
> Dibutuhkan sekitar 1 menit agar pemicu sumber SQS peristiwa Amazon diaktifkan.

4. Pilih Simpan.
5. Untuk melihat atribut yang dipulihkan, kembali ke konsol Amazon DynamoDB.
6. Untuk menonaktifkan pemutaran ulang, kembali ke AWS Lambda konsol dan nonaktifkan pemicu sumber SQS peristiwa Amazon untuk`ReplayFunction`.

Berlangganan AWS Event Fork Pipelines ke topik Amazon SNS

Untuk mempercepat pengembangan aplikasi berbasis peristiwa, Anda dapat berlangganan pipeline penanganan acara—yang didukung oleh pipa Event Fork— ke topik Amazon. AWS SNS AWS Event Fork Pipelines adalah rangkaian aplikasi bersarang sumber terbuka, berdasarkan Model Aplikasi AWS Tanpa Server (AWS SAM), yang dapat Anda gunakan langsung dari rangkaian AWS Event Fork Pipelines (pilih Tampilkan aplikasi yang membuat IAM peran khusus atau kebijakan sumber daya) ke akun Anda. AWS Untuk informasi selengkapnya, lihat Bagaimana AWS Event Fork Pipelines bekerja.

Bagian ini menunjukkan bagaimana Anda dapat menggunakan AWS Management Console untuk menyebarkan pipeline dan kemudian berlangganan AWS Event Fork Pipelines ke topik AmazonSNS. Sebelum Anda mulai, buat SNS topik Amazon.

Untuk menghapus sumber daya yang terdiri dari pipeline, cari pipeline di halaman Aplikasi di AWS Lambda konsol, perluas bagian SAM templat, pilih CloudFormationtumpukan, lalu pilih Tindakan Lain, Hapus Tumpukan.

Topik

- Menyebarkan dan berlangganan Event Storage dan Backup Pipeline ke Amazon SNS
- Menyebarkan dan berlangganan Event Search dan Analytics Pipeline ke Amazon SNS
- Menerapkan Pipeline Event Replay dengan integrasi Amazon SNS

Menyebarkan dan berlangganan Event Storage dan Backup Pipeline ke Amazon SNS

Untuk pengarsipan dan analitik acara, Amazon SNS sekarang merekomendasikan untuk menggunakan integrasi aslinya dengan Amazon Data Firehose. Anda dapat berlangganan aliran pengiriman Firehose ke SNS topik, yang memungkinkan Anda mengirim pemberitahuan ke titik akhir pengarsipan dan analitik seperti bucket Amazon Simple Storage Service (Amazon S3), tabel Amazon Redshift, Amazon Service (Service), dan banyak lagi OpenSearch . OpenSearc h Menggunakan Amazon SNS dengan aliran pengiriman Firehose adalah solusi yang dikelola sepenuhnya dan tanpa kode yang tidak mengharuskan Anda menggunakan fungsi. AWS Lambda Untuk informasi selengkapnya, lihat Aliran pengiriman Fanout ke Firehose.

Halaman ini menunjukkan cara menerapkan Event Storage dan Backup Pipeline dan berlangganan ke SNS topik Amazon. Proses ini secara otomatis mengubah AWS SAM template yang terkait dengan pipeline menjadi AWS CloudFormation tumpukan, dan kemudian menyebarkan tumpukan ke dalam file Anda Akun AWS. Proses ini juga menciptakan dan mengonfigurasi rangkaian sumber daya yang terdiri atas Penyimpanan Peristiwa dan Alur Cadangan, termasuk yang berikut ini:

- SQSAntrian Amazon
- Fungsi Lambda
- Aliran pengiriman Firehose
- Bucket cadangan Amazon S3

Untuk informasi selengkapnya tentang mengonfigurasi aliran dengan bucket S3 sebagai tujuan, lihat `S3DestinationConfiguration` di Referensi Firehose Data Amazon. API

Untuk informasi selengkapnya tentang mengubah peristiwa dan tentang mengonfigurasi buffering peristiwa, kompresi peristiwa, dan enkripsi peristiwa, lihat Membuat Aliran Pengiriman Firehose Data Amazon di Panduan Pengembang Amazon Data Firehose.

Untuk informasi selengkapnya tentang filter peristiwa, lihat Kebijakan filter SNS langganan Amazon dalam panduan ini.

1. Masuk ke konsol AWS Lambda tersebut.
2. Pada panel navigasi, pilih Fungsi dan kemudian pilih Buat fungsi.
3. Pada halaman Buat fungsi, lakukan hal berikut ini:

a. Pilih Jelajahi repositori aplikasi tanpa server, Aplikasi publik, Tampilkan aplikasi yang membuat IAM peran khusus atau kebijakan sumber daya.

b. Cari untuk `fork-event-storage-backup-pipeline` dan kemudian pilih aplikasi.

4. Pada halaman fork-event-storage-backup-pipeline, lakukan hal berikut:

a. Di bagian Pengaturan aplikasi, masukkan Nama aplikasi(sebagai contoh, `my-app-backup`).

> **Note**
>
> - Untuk setiap deployment, nama aplikasi harus unik. Jika Anda menggunakan kembali nama aplikasi, penerapan hanya akan memperbarui AWS CloudFormation tumpukan yang digunakan sebelumnya (bukan membuat yang baru).

b. (Opsional) Untuk BucketArn, masukkan bucket S3 tempat acara masuk dimuat. ARN Jika Anda tidak memasukkan nilai, bucket S3 baru akan dibuat di AWS akun Anda.

c. (Opsional) Untuk DataTransformationFunctionArn, masukkan fungsi Lambda yang melaluinya peristiwa yang masuk diubah. ARN Jika Anda tidak memasukkan nilai, pengubahan data dinonaktifkan.

d. (Opsional) Masukkan salah satu LogLevelpengaturan berikut untuk eksekusi fungsi Lambda aplikasi Anda:

- `DEBUG`
- `ERROR`
- `INFO` (default)
- `WARNING`

e. Untuk TopicArn, masukkan ARN SNS topik Amazon tempat instance pipa garpu ini akan berlangganan.

f. (Opsional) Untuk StreamBufferingIntervalInSecondsdan StreamBufferingSizeInMBs, masukkan nilai untuk mengonfigurasi buffering peristiwa yang masuk. Jika Anda tidak memasukkan nilai berapa pun, 300 detik dan 5 MB digunakan.

g. (Opsional) Masukkan salah satu StreamCompressionFormatpengaturan berikut untuk mengompresi peristiwa yang masuk:

- `GZIP`
- `SNAPPY`
- `UNCOMPRESSED` (default)
- `ZIP`

h. (Opsional) Untuk StreamPrefix, masukkan awalan string untuk memberi nama file yang disimpan di bucket cadangan S3. Jika Anda tidak memasukkan nilai, prefiks tidak digunakan.

i. (Opsional) Untuk SubscriptionFilterPolicy, masukkan kebijakan filter SNS langganan Amazon, dalam JSON format, yang akan digunakan untuk memfilter peristiwa yang masuk. Kebijakan filter menentukan peristiwa mana yang diindeks dalam indeks OpenSearch Layanan. Jika Anda tidak memasukkan nilai, tidak ada pemfilteran yang digunakan (semua peristiwa diindeks).

j. (Opsional) Untuk SubscriptionFilterPolicyScope, masukkan string `MessageBody` atau `MessageAttributes` untuk mengaktifkan pemfilteran pesan berbasis muatan atau atribut.

k. Pilih Saya mengakui bahwa aplikasi ini membuat IAM peran khusus, kebijakan sumber daya, dan menerapkan aplikasi bersarang. dan kemudian pilih Deploy.

Pada status Deployment untuk ***my-app***halaman, Lambda menampilkan status Aplikasi Anda sedang digunakan.

Di bagian Sumber Daya, AWS CloudFormation mulailah membuat tumpukan dan menampilkan PROGRESS status CREATE_IN_ untuk setiap sumber daya. Ketika proses selesai, AWS CloudFormation menampilkan COMPLETE status CREATE_.

Setelah deployment selesai, Lambda menampilkan status Aplikasi Anda telah di-deploy.

Pesan yang dipublikasikan ke SNS topik Amazon Anda disimpan di bucket cadangan S3 yang disediakan oleh pipeline Penyimpanan Acara dan Pencadangan secara otomatis.

Menyebarkan dan berlangganan Event Search dan Analytics Pipeline ke Amazon SNS

Untuk pengarsipan dan analitik acara, Amazon SNS sekarang merekomendasikan untuk menggunakan integrasi aslinya dengan Amazon Data Firehose. Anda dapat berlangganan aliran pengiriman Firehose ke SNS topik, yang memungkinkan Anda mengirim pemberitahuan ke titik akhir pengarsipan dan analitik seperti bucket Amazon Simple Storage Service (Amazon S3),

tabel Amazon Redshift, Amazon Service (Service), dan banyak lagi OpenSearch . OpenSearc h Menggunakan Amazon SNS dengan aliran pengiriman Firehose adalah solusi yang dikelola sepenuhnya dan tanpa kode yang tidak mengharuskan Anda menggunakan fungsi. AWS Lambda Untuk informasi selengkapnya, lihat Aliran pengiriman Fanout ke Firehose.

Halaman ini menunjukkan cara menerapkan Event Search dan Analytics Pipeline dan berlangganan ke SNS topik Amazon. Proses ini secara otomatis mengubah AWS SAM template yang terkait dengan pipeline menjadi AWS CloudFormation tumpukan, dan kemudian menyebarkan tumpukan ke dalam file Anda Akun AWS. Proses ini juga membuat dan mengonfigurasi rangkaian sumber daya yang terdiri atas Pencarian Peristiwa dan Alur Analitik, termasuk yang berikut ini:

- SQSAntrian Amazon
- Fungsi Lambda
- Aliran pengiriman Firehose
- Domain OpenSearch Layanan Amazon
- Bucket suta mati Amazon S3

Untuk informasi selengkapnya tentang mengonfigurasi aliran dengan indeks sebagai tujuan, lihat `ElasticsearchDestinationConfiguration` di Referensi APIFirehose Data Amazon.

Untuk informasi selengkapnya tentang mengubah peristiwa dan tentang mengonfigurasi buffering peristiwa, kompresi peristiwa, dan enkripsi peristiwa, lihat Membuat Aliran Pengiriman Firehose Data Amazon di Panduan Pengembang Amazon Data Firehose.

Untuk informasi selengkapnya tentang filter peristiwa, lihat Kebijakan filter SNS langganan Amazon dalam panduan ini.

1. Masuk ke konsol AWS Lambda tersebut.
2. Pada panel navigasi, pilih Fungsi dan kemudian pilih Buat fungsi.
3. Pada halaman Buat fungsi, lakukan hal berikut ini:
   a. Pilih Jelajahi repositori aplikasi tanpa server, Aplikasi publik, Tampilkan aplikasi yang membuat IAM peran khusus atau kebijakan sumber daya.
   b. Cari untuk `fork-event-search-analytics-pipeline` dan kemudian pilih aplikasi.
4. Pada halaman fork-event-search-analytics-pipeline, lakukan hal berikut:

a. Di bagian Pengaturan aplikasi, masukkan Nama aplikasi(sebagai contoh, `my-app-search`).

> **Note**
>
> Untuk setiap deployment, nama aplikasi harus unik. Jika Anda menggunakan kembali nama aplikasi, penerapan hanya akan memperbarui AWS CloudFormation tumpukan yang digunakan sebelumnya (bukan membuat yang baru).

b. (Opsional) Untuk DataTransformationFunctionArn, masukkan fungsi Lambda yang digunakan untuk mengubah peristiwa yang masuk. ARN Jika Anda tidak memasukkan nilai, pengubahan data dinonaktifkan.

c. (Opsional) Masukkan salah satu LogLevelpengaturan berikut untuk eksekusi fungsi Lambda aplikasi Anda:

- `DEBUG`
- `ERROR`
- `INFO` (default)
- `WARNING`

d. (Opsional) Untuk SearchDomainArn, masukkan domain OpenSearch Layanan, klaster yang mengonfigurasi fungsionalitas komputasi dan penyimpanan yang diperlukan. ARN Jika Anda tidak memasukkan nilai, domain baru dibuat dengan konfigurasi default.

e. Untuk TopicArn, masukkan ARN SNS topik Amazon tempat instance pipa garpu ini akan berlangganan.

f. Untuk SearchIndexName, masukkan nama indeks OpenSearch Layanan untuk pencarian acara dan analitik.

> **Note**
>
> Kuota berikut ini berlaku untuk nama indeks:
>
> - Tidak dapat menyertakan huruf besar
> - Tidak dapat menyertakan karakter berikut ini: `\ / * ? " < > | ` , #`
> - Tidak dapat dimulai dengan karakter berikut ini: `- + _`
> - Tidak boleh sebagai berikut: `. ..`

- Tidak boleh lebih dari 80 karakter
- Tidak boleh lebih dari 255 byte
- Tidak dapat berisi titik dua (dari OpenSearch Layanan 7.0)

g. (Opsional) Masukkan salah satu SearchIndexRotationPeriodpengaturan berikut untuk periode rotasi indeks OpenSearch Layanan:

- `NoRotation` (default)
- `OneDay`
- `OneHour`
- `OneMonth`
- `OneWeek`

Rotasi indeks menambahkan timestamp untuk nama indeks, yang memfasilitasi kedaluwarsanya data lama.

h. Untuk SearchTypeName, masukkan nama jenis OpenSearch Layanan untuk mengatur acara dalam indeks.

> **Note**
>
> - OpenSearch Nama tipe layanan dapat berisi karakter apa pun (kecuali byte nol) tetapi tidak dapat dimulai dengan. _
> - Untuk OpenSearch Layanan 6.x, hanya ada satu jenis per indeks. Jika Anda menentukan tipe baru untuk indeks yang sudah ada yang sudah memiliki tipe lain, Firehose akan menampilkan error runtime.

i. (Opsional) Untuk StreamBufferingIntervalInSecondsdan StreamBufferingSizeInMBs, masukkan nilai untuk mengonfigurasi buffering peristiwa yang masuk. Jika Anda tidak memasukkan nilai berapa pun, 300 detik dan 5 MB digunakan.

j. (Opsional) Masukkan salah satu StreamCompressionFormatpengaturan berikut untuk mengompresi peristiwa yang masuk:

- `GZIP`
- `SNAPPY`

- `UNCOMPRESSED` (default)
- `ZIP`

k. (Opsional) Untuk StreamPrefix, masukkan awalan string untuk memberi nama file yang disimpan di bucket huruf mati S3. Jika Anda tidak memasukkan nilai, prefiks tidak digunakan.

l. (Opsional) Untuk StreamRetryDurationInSecons, masukkan durasi coba lagi untuk kasus ketika Firehose tidak dapat mengindeks peristiwa dalam OpenSearch indeks Layanan. Jika Anda tidak memasukkan nilai, maka 300 detik akan digunakan.

m. (Opsional) Untuk SubscriptionFilterPolicy, masukkan kebijakan filter SNS langganan Amazon, dalam JSON format, yang akan digunakan untuk memfilter peristiwa yang masuk. Kebijakan filter menentukan peristiwa mana yang diindeks dalam indeks OpenSearch Layanan. Jika Anda tidak memasukkan nilai, tidak ada pemfilteran yang digunakan (semua peristiwa diindeks).

n. Pilih Saya mengakui bahwa aplikasi ini membuat IAM peran khusus, kebijakan sumber daya, dan menerapkan aplikasi bersarang. dan kemudian pilih Deploy.

Pada status Deployment untuk ***`my-app-search`***halaman, Lambda menampilkan status Aplikasi Anda sedang digunakan.

Di bagian Sumber Daya, AWS CloudFormation mulailah membuat tumpukan dan menampilkan PROGRESS status CREATE_IN_ untuk setiap sumber daya. Ketika proses selesai, AWS CloudFormation menampilkan COMPLETE status CREATE_.

Setelah deployment selesai, Lambda menampilkan status Aplikasi Anda telah di-deploy.

Pesan yang dipublikasikan ke SNS topik Amazon Anda diindeks dalam indeks OpenSearch Layanan yang disediakan oleh pipeline Event Search dan Analytics secara otomatis. Jika alur tidak dapat mengindeks peristiwa, alur akan menyimpannya dalam bucket surat mati S3.

Menerapkan Pipeline Event Replay dengan integrasi Amazon SNS

Halaman ini menunjukkan cara menerapkan Event Replay Pipeline dan berlangganan ke topik AmazonSNS. Proses ini secara otomatis mengubah AWS SAM template yang terkait dengan pipeline menjadi AWS CloudFormation tumpukan, dan kemudian menyebarkan tumpukan ke dalam file Anda Akun AWS. Proses ini juga membuat dan mengonfigurasi kumpulan sumber daya yang terdiri dari Event Replay Pipeline, termasuk SQS antrian Amazon dan fungsi Lambda.

Untuk informasi selengkapnya tentang filter peristiwa, lihat Kebijakan filter SNS langganan Amazon dalam panduan ini.

1. Masuk ke konsol AWS Lambda tersebut.
2. Pada panel navigasi, pilih Fungsi dan kemudian pilih Buat fungsi.
3. Pada halaman Buat fungsi, lakukan hal berikut ini:

   a. Pilih Jelajahi repositori aplikasi tanpa server, Aplikasi publik, Tampilkan aplikasi yang membuat IAM peran khusus atau kebijakan sumber daya.

   b. Cari untuk `fork-event-replay-pipeline` dan kemudian pilih aplikasi.

4. Pada fork-event-replay-pipelinehalaman, lakukan hal berikut:

   a. Di bagian Pengaturan aplikasi, masukkan Nama aplikasi(sebagai contoh, `my-app-replay`).

   > **Note**
   >
   > Untuk setiap deployment, nama aplikasi harus unik. Jika Anda menggunakan kembali nama aplikasi, penerapan hanya akan memperbarui AWS CloudFormation tumpukan yang digunakan sebelumnya (bukan membuat yang baru).

   b. (Opsional) Masukkan salah satu LogLevelpengaturan berikut untuk eksekusi fungsi Lambda aplikasi Anda:

      - `DEBUG`
      - `ERROR`
      - `INFO` (default)
      - `WARNING`

   c. (Opsional) Untuk ReplayQueueRetentionPeriodInSeconds, masukkan jumlah waktu, dalam detik, di mana antrian SQS replay Amazon menyimpan pesan. Jika Anda tidak memasukkan nilai, 1.209.600 detik (14 hari) akan digunakan.

   d. Untuk TopicArn, masukkan ARN SNS topik Amazon tempat instance pipa garpu ini akan berlangganan.

   e. Untuk DestinationQueueName, masukkan nama SQS antrian Amazon tempat fungsi pemutaran ulang Lambda meneruskan pesan.

f. (Opsional) Untuk SubscriptionFilterPolicy, masukkan kebijakan filter SNS langganan Amazon, dalam JSON format, yang akan digunakan untuk memfilter peristiwa yang masuk. Kebijakan filter memutuskan peristiwa mana yang akan dibuffer untuk ulangan. Jika Anda tidak memasukkan nilai, tidak ada pemfilteran digunakan (semua peristiwa dibuffer untuk ulangan).

g. Pilih Saya mengakui bahwa aplikasi ini membuat IAM peran khusus, kebijakan sumber daya, dan menerapkan aplikasi bersarang. dan kemudian pilih Deploy.

Pada status Deployment untuk ***my-app-replay***halaman, Lambda menampilkan status Aplikasi Anda sedang digunakan.

Di bagian Sumber Daya, AWS CloudFormation mulailah membuat tumpukan dan menampilkan PROGRESS status CREATE_IN_ untuk setiap sumber daya. Ketika proses selesai, AWS CloudFormation menampilkan COMPLETE status CREATE_.

Setelah deployment selesai, Lambda menampilkan status Aplikasi Anda telah di-deploy.

Pesan yang dipublikasikan ke SNS topik Amazon Anda di-buffer untuk diputar ulang dalam SQS antrean Amazon yang disediakan oleh Event Replay Pipeline secara otomatis.

> **Note**
>
> Secara default, ulangan dinonaktifkan. Untuk mengaktifkan pemutaran ulang, navigasikan ke halaman fungsi di konsol Lambda, perluas bagian Desainer, pilih SQSubin dan kemudian, di SQSbagian, pilih Diaktifkan.

Menggunakan Amazon EventBridge Scheduler dengan Amazon SNS

Amazon EventBridge Scheduler adalah penjadwal tanpa server yang memungkinkan Anda membuat, menjalankan, dan mengelola tugas dari satu layanan terpusat dan terkelola. Dengan EventBridge Scheduler, Anda dapat membuat jadwal menggunakan Cron dan ekspresi tingkat untuk pola berulang, atau mengonfigurasi pemanggilan satu kali. Anda dapat mengatur jendela waktu fleksibel untuk pengiriman, menentukan batas coba lagi, dan mengatur waktu retensi maksimum untuk API pemanggilan yang gagal.

Halaman ini menjelaskan cara menggunakan EventBridge Scheduler untuk mempublikasikan pesan dari SNS topik Amazon sesuai jadwal.

Topik

Mengatur peran eksekusi

Saat Anda membuat jadwal baru, EventBridge Scheduler harus memiliki izin untuk menjalankan API operasi targetnya atas nama Anda. Anda memberikan izin ini ke EventBridge Scheduler menggunakan peran eksekusi. Kebijakan izin yang Anda lampirkan ke peran eksekusi jadwal menentukan izin yang diperlukan. Izin ini bergantung pada target yang ingin API Anda panggil EventBridge Scheduler.

Bila Anda menggunakan konsol EventBridge Scheduler untuk membuat jadwal, seperti dalam prosedur berikut, EventBridge Scheduler secara otomatis mengatur peran eksekusi berdasarkan target yang Anda pilih. Jika Anda ingin membuat jadwal menggunakan salah satu EventBridge SchedulerSDKs, atau AWS CLI AWS CloudFormation, Anda harus memiliki peran eksekusi yang ada yang memberikan izin EventBridge Scheduler yang diperlukan untuk memanggil target. Untuk informasi selengkapnya tentang mengatur peran eksekusi secara manual untuk jadwal Anda, lihat Mengatur peran eksekusi di Panduan Pengguna EventBridge Penjadwal.

Buat jadwal

Untuk membuat jadwal dengan menggunakan konsol

1. Buka konsol Amazon EventBridge Scheduler di https://console.aws.amazon.com/scheduler/rumah.
2. Pada halaman Jadwal, pilih Buat jadwal.
3. Pada halaman Tentukan detail jadwal, di bagian Nama jadwal dan deskripsi, lakukan hal berikut:
    a. Untuk nama Jadwal, masukkan nama untuk jadwal Anda. Misalnya, **`MyTestSchedule`**.
    b. (Opsional) Untuk Deskripsi, masukkan deskripsi untuk jadwal Anda. Misalnya, **`My first schedule`**.

c. Untuk grup Jadwal, pilih grup jadwal dari daftar dropdown. Jika Anda tidak memiliki grup, pilih default. Untuk membuat grup jadwal, pilih buat jadwal Anda sendiri.

   Anda menggunakan grup jadwal untuk menambahkan tag ke grup jadwal.

4. • Pilih opsi jadwal Anda.

Kejadian	Lakukan ini...	
Jadwal satu kali Jadwal satu kali memanggil target hanya sekali pada tanggal dan waktu yang Anda tentukan.	Untuk tanggal dan waktu, lakukan hal berikut: • Masukkan tanggal yang valid dalam `YYYY/MM/DD` format. • Masukkan stempel waktu dalam format 24 jam`hh:mm`. • Untuk Timezone, pilih zona waktu.	
Jadwal berulang Jadwal berulang memanggil target pada tingkat yang Anda tentukan menggunakan cron ekspresi atau ekspresi tingkat.	a. Untuk jenis Jadwal, lakukan salah satu hal berikut: • Untuk menggunakan ekspresi cron untuk menentukan jadwal, pilih Jadwal berbasis Cron dan masukkan ekspresi cron. • Untuk menggunakan ekspresi laju untuk menentukan jadwal, pilih Jadwal berbasis tarif dan masukkan ekspresi laju.	

Kejadian	Lakukan ini...	
	Untuk informasi selengkapnya tentang ekspresi cron dan rate, lihat Menjadwalkan jenis pada EventBridge Scheduler di Panduan Pengguna EventBridge Penjadwal Amazon. b. Untuk jendela waktu Fleksibel, pilih Nonaktif untuk mematikan opsi, atau pilih salah satu jendela waktu yang telah ditentukan sebelumny a. Misalnya, jika Anda memilih 15 menit dan Anda menetapkan jadwal berulang untuk memanggil targetnya setiap jam sekali, jadwal berjalan dalam 15 menit setelah dimulainya setiap jam.	

5. (Opsional) Jika Anda memilih Jadwal berulang pada langkah sebelumnya, di bagian Jangka Waktu, lakukan hal berikut:

   a. Untuk Timezone, pilih zona waktu.

   b. Untuk Tanggal dan waktu mulai, masukkan tanggal yang valid dalam `YYYY/MM/DD` format, lalu tentukan stempel waktu dalam format 24 jam`hh:mm`.

   c. Untuk Tanggal dan waktu berakhir, masukkan tanggal yang valid dalam `YYYY/MM/DD` format, lalu tentukan stempel waktu dalam format 24 jam`hh:mm`.

6. Pilih Berikutnya.
7. Pada halaman Pilih target, pilih AWS API operasi yang dipanggil EventBridge Scheduler:

   a. Pilih Amazon SNS Publish.
   b. Di bagian Publikasikan, pilih SNS topik atau pilih Buat baru SNS topik.
   c. (Opsional) Masukkan JSON muatan. Jika Anda tidak memasukkan payload, EventBridge Scheduler menggunakan peristiwa kosong untuk menjalankan fungsi.
8. Pilih Berikutnya.
9. Pada halaman Pengaturan, lakukan hal berikut:

   a. Untuk mengaktifkan jadwal, di bawah Status jadwal, alihkan Aktifkan jadwal.
   b. Untuk mengonfigurasi kebijakan coba ulang untuk jadwal Anda, di bawah Kebijakan Coba ulang dan antrean huruf mati (DLQ), lakukan hal berikut:

      - Beralih Coba Lagi.
      - Untuk usia maksimum acara, masukkan jam maksimum dan min yang harus disimpan oleh EventBridge Scheduler untuk menyimpan acara yang belum diproses.
      - Waktu maksimum adalah 24 jam.
      - Untuk percobaan ulang Maksimum, masukkan jumlah maksimum kali EventBridge Scheduler mencoba ulang jadwal jika target mengembalikan kesalahan.

        Nilai maksimumnya adalah 185 percobaan ulang.

      Dengan kebijakan coba lagi, jika jadwal gagal memanggil targetnya, EventBridge Scheduler menjalankan kembali jadwal. Jika dikonfigurasi, Anda harus mengatur waktu retensi maksimum dan mencoba ulang untuk jadwal.
   c. Pilih tempat EventBridge Scheduler menyimpan acara yang tidak terkirim.

Opsi antrian huruf mati () DLQ	Lakukan ini...	
Jangan simpan	Pilih Tidak Ada.	
Simpan acara di tempat yang sama Akun AWS	a. Pilih Pilih SQS antrian Amazon di my Akun AWS as a DLQ.	

Opsi antrian huruf mati () DLQ	Lakukan ini...	
di mana Anda membuat jadwal	b. Pilih Nama Sumber Daya Amazon (ARN) dari SQS antrian Amazon.	
Simpan acara di tempat yang berbeda Akun AWS dari tempat Anda membuat jadwal	a. Pilih Tentukan SQS antrian Amazon di yang lain Akun AWS sebagai DLQ. b. Masukkan Nama Sumber Daya Amazon (ARN) dari SQS antrian Amazon.	

d. Untuk menggunakan kunci yang dikelola pelanggan untuk mengenkripsi input target Anda, di bawah Enkripsi, pilih Sesuaikan pengaturan enkripsi (lanjutan).

   Jika Anda memilih opsi ini, masukkan KMS kunci yang ada ARN atau pilih Buat AWS KMS key untuk menavigasi ke AWS KMS konsol. Untuk informasi selengkapnya tentang cara EventBridge Scheduler mengenkripsi data Anda saat istirahat, lihat Enkripsi saat istirahat di Panduan Pengguna EventBridge Penjadwal Amazon.

e. Agar EventBridge Scheduler membuat peran eksekusi baru untuk Anda, pilih Buat peran baru untuk jadwal ini. Kemudian, masukkan nama untuk nama Peran. Jika Anda memilih opsi ini, EventBridge Scheduler melampirkan izin yang diperlukan untuk target template Anda ke peran.

10. Pilih Berikutnya.
11. Di halaman Tinjau dan buat jadwal, tinjau detail jadwal Anda. Di setiap bagian, pilih Edit untuk kembali ke langkah itu dan mengedit detailnya.
12. Pilih Buat jadwal.

    Anda dapat melihat daftar jadwal baru dan yang sudah ada di halaman Jadwal. Di bawah kolom Status, verifikasi bahwa jadwal baru Anda Diaktifkan.

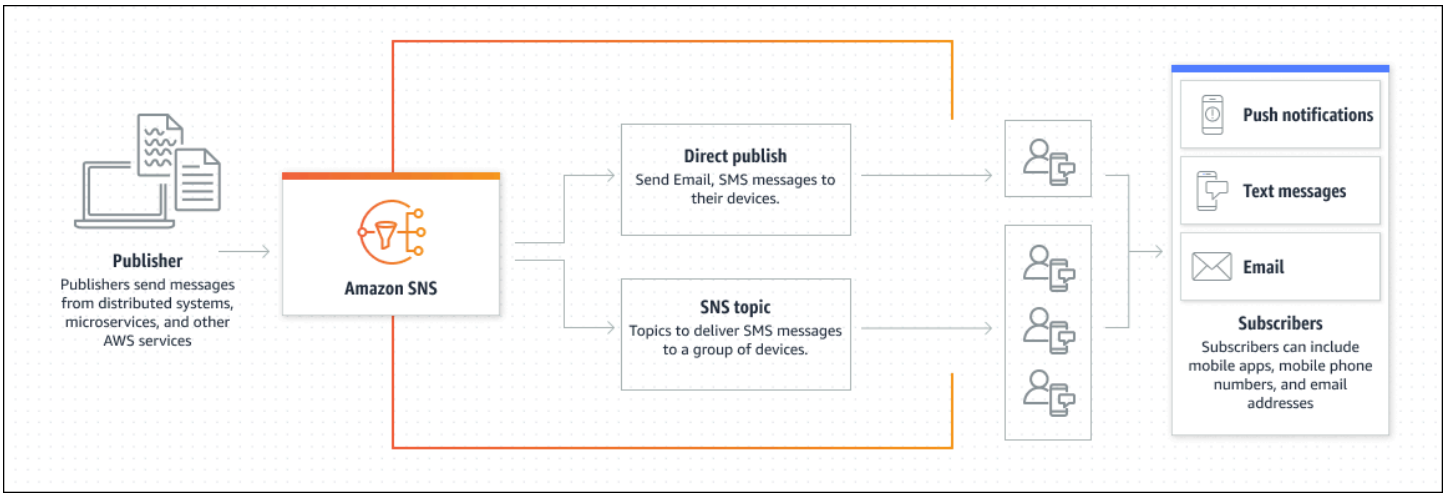
Sumber daya terkait

Untuk informasi selengkapnya tentang EventBridge Scheduler, lihat berikut ini:

- EventBridge Panduan Pengguna Scheduler
- EventBridge Referensi Scheduler API
- EventBridge Penetapan Harga Scheduler

Menggunakan Amazon SNS untuk application-to-person pengiriman pesan

Pesan Amazon SNS application-to-person (A2P) memungkinkan Anda mengirimkan notifikasi dan peringatan langsung ke perangkat seluler pelanggan Anda melalui SMS (Layanan Pesan Singkat). Dengan menggunakan fitur ini, Anda dapat mengirim pemberitahuan push ke aplikasi seluler, pesan teks ke nomor ponsel, dan email teks biasa ke alamat email. Selain itu, Anda memiliki fleksibilitas untuk mendistribusikan pesan dengan menggunakan topik untuk menjangkau beberapa penerima, atau mempublikasikan pesan langsung ke titik akhir seluler individual untuk komunikasi yang dipersonalisasi.



Topik berikut menjelaskan cara menggunakan Amazon SNS untuk pemberitahuan pengguna dengan pelanggan seperti aplikasi seluler, nomor ponsel, dan alamat email:

Topik

- Pesan teks seluler dengan Amazon SNS
- Mengirim notifikasi push seluler dengan Amazon SNS
- Pengaturan dan manajemen langganan SNS email Amazon

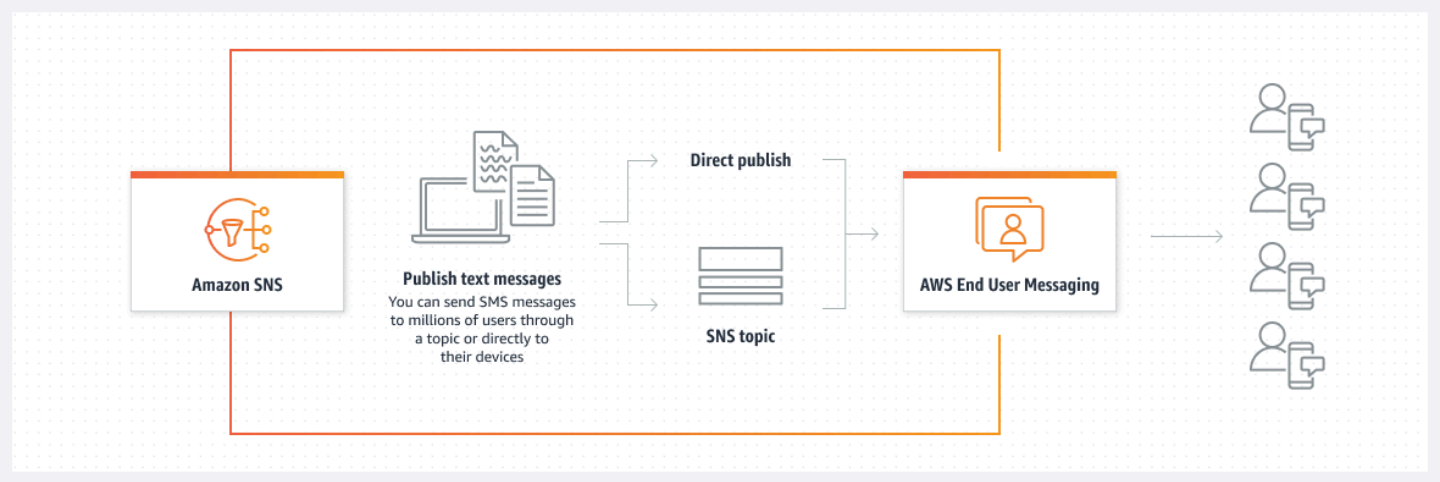
Pesan teks seluler dengan Amazon SNS

> **Important**
>
> Panduan SNS SMS Pengembang Amazon telah diperbarui. Amazon SNS telah terintegrasi dengan AWS Olah Pesan Pengguna Akhir SMSpengiriman SMS pesan. Panduan ini berisi informasi terbaru tentang cara membuat, mengonfigurasi, dan mengelola SNS SMS pesan Amazon Anda.

Amazon SNS mobile text messaging (SMS) dirancang untuk memfasilitasi pengiriman pesan ke berbagai platform, seperti web, mobile, dan aplikasi bisnis yang mendukungSMS. Pengguna dapat mengirim pesan ke satu atau beberapa nomor telepon dengan berlangganan topik, menyederhanakan proses distribusi.

SNSPesan Amazon dikirimkan oleh AWS Olah Pesan Pengguna Akhir SMS, yang memastikan transmisi pesan yang andal. Di Amazon SNSAPIs, Anda dapat mengatur berbagai properti seperti jenis pesan (promosi atau transaksional), batas pengeluaran bulanan, daftar opt-out, dan pengoptimalan pengiriman pesan.

AWS Olah Pesan Pengguna Akhir SMS menangani transmisi pesan ke nomor telepon tujuan melalui jaringan SMS pasokan globalnya. Ini mengelola perutean, status pengiriman, dan kepatuhan yang diperlukan dengan peraturan regional. Untuk mengakses SMS fitur tambahan seperti izin granular, kumpulan telepon, set konfigurasi, SMS simulator, dan aturan negara, lihat Panduan Pengguna.AWS Olah Pesan Pengguna Akhir SMS



Fitur utama berikut membantu Anda mengirim SNS SMS pesan Amazon yang dapat diskalakan dan mudah diperluas:

Sesuaikan preferensi pesan

Sesuaikan SMS pengiriman untuk Anda Akun AWS dengan mengatur SMS preferensi berdasarkan anggaran dan kasus penggunaan Anda. Misalnya, Anda dapat memilih apakah pesan Anda memprioritaskan efisiensi biaya atau pengiriman yang andal.

Tetapkan kuota pengeluaran

Sesuaikan SMS pengiriman Anda dengan menentukan kuota pengeluaran atau untuk pengiriman pesan individual dan kuota pengeluaran bulanan untuk Anda. Akun AWS Jika diwajibkan oleh undang-undang dan peraturan setempat (seperti AS dan Kanada), SMS penerima dapat memilih keluar, yang berarti bahwa mereka memilih untuk berhenti menerima SMS pesan dari Anda. Akun AWS Setelah penerima memilih keluar dari menerima pesan, Anda dapat, dengan batasan, memilih kembali nomor telepon sehingga Anda dapat melanjutkan pengiriman pesan.

Kirim SMS pesan secara global

Amazon SNS mendukung SMS pengiriman pesan di beberapa wilayah, memungkinkan Anda mengirim pesan ke lebih dari 240 negara dan wilayah.

Bagaimana Amazon SNS mengirimkan SMS pesan saya?

Saat Anda meminta Amazon SNS untuk mengirim SMS atas nama Anda, pesan dikirim menggunakan. AWS Olah Pesan Pengguna Akhir SMS Integrasi antara Amazon SNS dan AWS Olah Pesan Pengguna Akhir SMS menawarkan manfaat berikut:

IAMdan kebijakan sumber daya

Anda dapat memanfaatkan IAM dan kebijakan sumber daya untuk mengontrol dan mendistribusikan akses ke SMS sumber daya Anda di seluruh AWS layanan dan wilayah lain.

AWS Olah Pesan Pengguna Akhir SMS konfigurasi

Semua konfigurasi terkait ID originasi (pembuatan, pembaruan konfigurasi, penyediaan originasi baruIDs, mengubah templat pendaftaran) digunakan. AWS Olah Pesan Pengguna Akhir SMS

AWS Olah Pesan Pengguna Akhir SMS penagihan

Namun AWS Olah Pesan Pengguna Akhir SMS, semua SMS penagihan sudah selesai. Anda dapat mengkonsolidasikan AWS pengeluaran Anda untuk SMS beban kerja Anda, sambil membeli dan mengelola SMS sumber daya Anda di satu lokasi pusat.

Memulai dengan Amazon SNS SMS

> **Important**
>
> Panduan SNS SMS Pengembang Amazon telah diperbarui. Amazon SNS telah terintegrasi dengan AWS Olah Pesan Pengguna Akhir SMSpengiriman SMS pesan. Panduan ini berisi informasi terbaru tentang cara membuat, mengonfigurasi, dan mengelola SNS SMS pesan Amazon Anda.

Topik ini memandu Anda mengelola SMS kotak pasir dan mengonfigurasi IAM serta kebijakan berbasis sumber daya untuk memberikan SNS Amazon izin yang diperlukan untuk mengakses dan memanfaatkan. AWS Olah Pesan Pengguna Akhir SMS APIs

Topik

- Memulai dengan manajemen SNS SMS akses Amazon
- Prasyarat
- Menggunakan kotak SNS SMS pasir Amazon

Memulai dengan manajemen SNS SMS akses Amazon

> **Important**
>
> Panduan SNS SMS Pengembang Amazon telah diperbarui. Amazon SNS telah terintegrasi dengan AWS Olah Pesan Pengguna Akhir SMSpengiriman SMS pesan. Panduan ini berisi informasi terbaru tentang cara membuat, mengonfigurasi, dan mengelola SNS SMS pesan Amazon Anda.

Untuk mengaktifkan SMS perpesanan di AmazonSNS, Anda perlu memberi Amazon izin SNS yang diperlukan untuk mengakses SMS sumber daya Anda dan menelepon AWS Olah Pesan Pengguna Akhir SMS APIs atas nama Anda. Ada dua mekanisme utama yang mengontrol akses ini:

1. IAMKebijakan yang memberikan akses ke AWS Olah Pesan Pengguna Akhir SMS APIs
2. Kebijakan berbasis sumber daya untuk memberikan izin sumber daya AWS Olah Pesan Pengguna Akhir SMS

   Secara default, SMS sumber daya seperti daftar originasi IDs dan opt-out memiliki kebijakan sumber daya yang memberikan izin Amazon. SNS

Topik

- SMSIAMkebijakan
- Mengelola SNS IAM kebijakan Amazon khusus
- Kebijakan berbasis sumber daya

SMSIAMkebijakan

SMS AWS Identity and Access Management (IAM) kebijakan mengacu pada kebijakan yang memberikan Amazon izin SNS yang diperlukan untuk mengakses dan menggunakan AWS Olah Pesan Pengguna Akhir SMS APIs. Kebijakan ini menentukan tindakan yang diizinkan Amazon SNS untuk dilakukan saat berinteraksi dengan AWS Olah Pesan Pengguna Akhir SMS sumber daya, seperti mengirim SMS pesan.

1. Jika Anda tidak menggunakan peran Admin, lampirkan IAM kebijakan yang menyertakan `sms-voice`APIs.

   Jika Anda berada di kotak pasir, Anda dapat mulai mengirim SMS pesan ke nomor telepon tujuan terverifikasi tanpa menyetel kebijakan sumber daya tambahan.
2. Jika Anda meminta identitas originasi baru, pilih kebijakan yang sesuai di konsol. Ini memberi Amazon SNS dan AWS Olah Pesan Pengguna Akhir SMS akses ke sumber daya.
3. Jika Anda ingin menggunakan opt-out, daftar opt-out default tidak memiliki kebijakan sumber daya default. Anda harus mengonfigurasi kebijakan sumber daya secara manual AWS Olah Pesan Pengguna Akhir SMS untuk menggunakan Amazon SNSAPIs.

Gunakan IAM kebijakan berikut untuk mengakses semua yang SMS terkait APIs diSNS:

> **Note**
>
> `sms-voice:SendTextMessage`dan opt-out tidak APIs ada dalam contoh berikut.

```
{
    "Effect": "Allow",
    "Principal": { "Service": "sns.amazonaws.com" },
    "Action": [
        "sns:*",
        "sms-voice:CreateVerifiedDestinationPhoneNumber",
        "sms-voice:DeleteVerifiedDestinationPhoneNumber",
        "sms-voice:GetAccountTier",
        "sms-voice:DescribePhoneNumbers",
        "sms-voice:DescribeDestinationPhoneNumbers",
        "sms-voice:VerifyDestinationPhoneNumber",
        "sms-voice:DescribePhoneNumbers",
        "sms-voice:DescribeSpendLimits",
        "sms-voice:DescribeConfigurationSets",
        "sms-voice:SetTextMessageSpendLimitOverride",
        "sms-voice:UpdateRouteType",
        "sms-voice:UpdateSenderId"
     ]
    "Resource": "*",
    "Condition": { "StringEquals": { "aws:SourceAccount": "<owner account>" } }
}
```

Gunakan IAM kebijakan berikut untuk mengakses semua fungsionalitas terkait SMS (publikasi langsung) diSNS:

```
{
    "Effect": "Allow",
    "Principal": { "Service": "sns.amazonaws.com" },
    "Action": [
        "sns:CreateSMSSandboxPhoneNumber",
        "sns:DeleteSMSSandboxPhoneNumber",
        "sns:GetSMSSandboxPhoneNumber",
        "sns:ListSMSSandboxPhoneNumber",
        "sns:VerifySMSSandboxPhoneNumber",
        "sns:ListOriginationNumbers",
        "sns:CheckIfPhoneNumberIsOptedOut",
        "sns:GetSMSAttributes",
```

```
        "sns:SetSMSAttributes",
        "sns:Publish",
        "sms-voice:CreateVerifiedDestinationPhoneNumber",
        "sms-voice:DeleteVerifiedDestinationPhoneNumber",
        "sms-voice:GetAccountTier",
        "sms-voice:DescribePhoneNumbers",
        "sms-voice:DescribeDestinationPhoneNumbers",
        "sms-voice:VerifyDestinationPhoneNumber",
        "sms-voice:DescribePhoneNumbers",
        "sms-voice:DescribeSpendLimits",
        "sms-voice:DescribeConfigurationSets",
        "sms-voice:SetTextMessageSpendLimitOverride",
        "sms-voice:UpdateRouteType",
        "sms-voice:UpdateSenderId"
     ]
    "Resource": "*"
}
```

Mengelola SNS IAM kebijakan Amazon khusus

IAMKebijakan khusus memungkinkan Anda menentukan izin untuk IAM pengguna, grup, atau peran individual, memberikan atau membatasi akses ke AWS sumber daya dan tindakan tertentu. Saat mengelola SNS sumber daya Amazon, IAM kebijakan khusus memungkinkan Anda menyesuaikan izin akses sesuai dengan persyaratan keamanan dan operasional organisasi Anda.

Gunakan langkah-langkah berikut untuk mengelola IAM kebijakan khusus untuk AmazonSNS:

1. Masuk ke AWS Management Console dan buka IAM konsol di https://console.aws.amazon.com/iam/.
2. Dari panel navigasi, pilih Kebijakan.
3. Untuk membuat IAM kebijakan kustom baru, pilih Buat kebijakan dan pilih SNS. Untuk mengedit kebijakan yang ada, pilih kebijakan dari daftar dan pilih Edit kebijakan.
4. Di editor kebijakan, tentukan izin untuk mengakses sumber daya AmazonSNS. Anda dapat menentukan tindakan, sumber daya, dan kondisi berdasarkan kebutuhan spesifik Anda.
5. Untuk memberikan izin untuk SNS tindakan Amazon, sertakan SNS tindakan Amazon yang relevan seperti `sns:Publishsns:Subscribe`, dan `sns:DeleteTopic` dalam IAM kebijakan Anda. Tentukan ARN (Nama Sumber Daya Amazon) dari SNS topik Amazon yang menerapkan izin.

6. Tentukan IAM pengguna, grup, atau peran yang harus dilampirkan kebijakan tersebut. Anda dapat melampirkan kebijakan secara langsung ke IAM pengguna atau grup, atau mengaitkannya dengan IAM peran yang digunakan oleh Layanan AWS atau aplikasi.
7. Tinjau konfigurasi IAM kebijakan untuk memastikannya selaras dengan persyaratan kontrol akses Anda. Setelah diverifikasi, simpan perubahan kebijakan.
8. Lampirkan IAMkebijakan kustom ke IAM pengguna, grup, atau peran yang relevan di dalam Anda Akun AWS. Ini memberi mereka izin yang ditentukan dalam kebijakan untuk mengelola sumber daya AmazonSNS.

Kebijakan berbasis sumber daya

Kebijakan SNS berbasis sumber daya Amazon digunakan untuk mengontrol akses ke sumber daya SMS pesan dan mengelola izin untuk mengirim pesan atas nama Anda. Kebijakan ini menentukan siapa yang dapat melakukan tindakan pada sumber daya SMS pesan, seperti mengirim pesan atau mengelola identitas originasi.

Dengan mengonfigurasi kebijakan berbasis sumber daya, Anda dapat menentukan AWS identitas atau akun mana yang memiliki izin untuk mengakses dan berinteraksi dengan fungsionalitas pesan Amazon. SMS SNS Ini membantu memastikan keamanan dan kepatuhan dengan membatasi akses ke pengguna atau sistem yang berwenang sambil memungkinkan mereka untuk memanfaatkan kemampuan SMS pesan yang disediakan oleh Amazon. SNS

Identitas asal

Saat mengirim SMS pesan menggunakan AmazonSNS, Anda dapat mengidentifikasi diri Anda kepada penerima menggunakan identitas originasi. Gunakan kebijakan berbasis sumber daya berikut untuk mengirim SMS pesan menggunakan identitas originasi:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "sns.amazonaws.com"
            },
            "Action": "sms-voice:SendTextMessage",
            "Resource": "arn:aws:sms-voice:us-east-1:555555555555:phone-number/
phone-11aa2b3333c44444d55e6ffff77gggg8",
            "Condition": {
```

```
                "StringEquals": {
                    "aws:SourceAccount": "111111111111"
                }
            }
        }
    ]
}
```

Prasyarat

Amazon SNS merekomendasikan untuk memperbarui IAM kebijakan Anda untuk menyertakan tindakan berikut untuk memastikan kontrol dan visibilitas komprehensif atas SNS sumber daya Amazon Anda:

- `AmazonSNSFullAccess`
- `AmazonSNSReadOnly`

Menggunakan kotak SNS SMS pasir Amazon

SNSSMSAkun Amazon yang baru dibuat secara otomatis ditempatkan ke SMS kotak pasir untuk memastikan keamanan AWS pelanggan dan penerima dengan mengurangi risiko penipuan dan penyalahgunaan. Lingkungan ini berfungsi sebagai ruang yang aman untuk tujuan pengujian dan pengembangan. Saat beroperasi di dalam SMS kotak pasir, Anda memiliki akses ke semua SNS fitur Amazon tetapi tunduk pada batasan tertentu:

- Anda hanya dapat mengirim SMS pesan ke nomor telepon tujuan terverifikasi.
- Anda dapat memiliki hingga 10 nomor telepon tujuan terverifikasi.
- Anda dapat menghapus nomor telepon tujuan hanya setelah minimal 24 jam berlalu sejak verifikasi, atau upaya verifikasi terakhir.

Setelah akun Anda keluar dari kotak pasir, pembatasan ini dihapus, dan Anda dapat mengirim SMS pesan ke penerima mana pun.

Langkah pertama

SNSSMSAkun Amazon baru ditempatkan di SMS kotak pasir. Gunakan langkah-langkah berikut untuk membuat dan mengelola nomor telepon di kotak pasir Anda, membuat nomor originasi dan pengirimIDs, dan mendaftarkan perusahaan Anda.

1. Tambahkan nomor telepon tujuan ke SMS kotak pasir. Untuk detail tentang menambahkan, mengelola, dan memindahkan nomor telepon dari SNS SMS kotak pasir Amazon, lihatMenambahkan dan memverifikasi nomor telepon di kotak SNS SMS pasir Amazon.
2. Buat identitas originasi yang dilihat penerima di perangkat mereka saat Anda mengirimi mereka pesan. SMS Untuk mempelajari lebih lanjut tentang identitas originasi, termasuk berbagai jenis yang dapat Anda gunakan, lihat dokumentasi. Identitas originasi untuk pesan Amazon SNS SMS
3. Daftarkan perusahaan Anda. Beberapa negara mengharuskan Anda untuk mendaftarkan identitas perusahaan Anda untuk dapat membeli nomor telepon atau pengirim IDs dan meninjau pesan yang Anda kirim ke penerima di negara mereka. Untuk informasi tentang negara mana yang memerlukan pendaftaran, lihat Negara dan wilayah yang didukung untuk SMS mengirim pesan AWS Olah Pesan Pengguna Akhir SMS di Panduan AWS Olah Pesan Pengguna Akhir SMS Pengguna.
4. Kirim pesan Anda ke topik atau ponsel. Untuk informasi selengkapnya, lihat Mengirim SMS pesan menggunakan Amazon SNS.

Topik

- Menambahkan dan memverifikasi nomor telepon di kotak SNS SMS pasir Amazon
- Menghapus nomor telepon dari kotak pasir Amazon SNS SMS
- Pindah dari kotak SNS SMS pasir Amazon

Menambahkan dan memverifikasi nomor telepon di kotak SNS SMS pasir Amazon

Untuk mulai mengirim SMS pesan saat AWS akun Anda berada di SMSkotak pasir, Anda harus terlebih dahulu melakukan hal berikut:

1. Buat ID originasi. Seperti halnya akun yang tidak ada di SMS kotak pasir, ID originasi diperlukan sebelum Anda dapat mengirim SMS pesan ke penerima di beberapa negara atau wilayah. Untuk informasi selengkapnya, lihat Memilih nomor telepon atau ID pengirim di Panduan AWS Olah Pesan Pengguna Akhir SMS Pengguna.
2. Tambahkan nomor telepon tujuan ke SNS kotak pasir Amazon.
3. Verifikasi nomor telepon.

Cara menambahkan dan memverifikasi nomor telepon tujuan

1. Masuk ke SNSkonsol Amazon.

2. Di menu konsol, pilih wilayah yang mendukung SMS pengiriman pesan.
3. Di panel navigasi, pilih Pesan teks (SMS).
4. Pada halaman Pesan teks seluler (SMS), di bawah Nomor telepon tujuan Sandbox, pilih Tambahkan nomor telepon.
5. Di bawah Detail tujuan, masukkan kode negara dan nomor telepon, tentukan bahasa apa yang akan digunakan untuk pesan verifikasi, lalu pilih Tambahkan nomor telepon.

   Amazon SNS mengirimkan kata sandi satu kali (OTP) ke nomor telepon tujuan. Jika nomor telepon tujuan tidak menerima OTP dalam waktu 15 menit, pilih Kirim ulang kode verifikasi. Anda dapat mengirim OTP ke nomor telepon tujuan yang sama hingga lima kali setiap 24 jam.
6. Di kotak Kode verifikasi, masukkan nomor telepon yang OTP dikirim ke tujuan, lalu pilih Verifikasi nomor telepon.

   Nomor telepon tujuan dan status verifikasinya muncul di bagian Sandbox destination phone numbers (Nomor telepon tujuan sandbox). Jika status verifikasinya adalah Pending (Menunggu), verifikasi tidak berhasil. Ini dapat terjadi, misalnya, jika Anda tidak memasukkan kode negara dengan nomor telepon tersebut. Anda hanya dapat menghapus nomor telepon tujuan terverifikasi atau dalam status menunggu setelah 24 jam atau lebih sejak verifikasi atau upaya verifikasi terakhir.
7. Ulangi langkah-langkah ini di setiap Wilayah tempat Anda ingin menggunakan nomor telepon tujuan ini.

Memecahkan masalah tidak diterimanya teks OTP

Memecahkan masalah umum yang dapat mencegah nomor telepon menerima OTP teks.

- Batas SNS SMS pengeluaran Amazon: Jika Anda Akun AWS telah melampaui batas pengeluaran untuk mengirim SMS pesan, pesan lebih lanjut, termasuk OTP teks, mungkin tidak dikirimkan hingga batas tersebut ditingkatkan atau masalah penagihan diselesaikan.
- Nomor telepon yang tidak dipilih untuk SMS pemberitahuan: Di beberapa negara atau wilayah, penerima harus memilih untuk menerima SMS pesan dari kode pendek, yang biasanya digunakan untuk OTP teks. Jika nomor telepon penerima tidak dipilih, mereka tidak akan menerima OTP teks.
- Pembatasan atau penyaringan operator: Beberapa operator seluler mungkin memiliki batasan atau mekanisme penyaringan yang mencegah pengiriman jenis SMS pesan tertentu, termasuk OTP teks. Ini bisa disebabkan oleh kebijakan keamanan atau tindakan anti-spam yang diterapkan oleh operator.

- Nomor telepon tidak valid atau salah: Jika nomor telepon yang diberikan oleh penerima salah atau tidak valid, OTP teks tidak akan dikirimkan.
- Masalah jaringan: Masalah jaringan sementara atau pemadaman dapat mencegah pengiriman SMS pesan, termasuk OTP teks, ke telepon penerima.
- Pengiriman tertunda: Dalam beberapa kasus, SMS pesan mungkin mengalami keterlambatan pengiriman karena kemacetan jaringan atau faktor lainnya. OTPTeks pada akhirnya dapat dikirim, tetapi bisa ditunda di luar jangka waktu yang diharapkan.
- Penangguhan atau penghentian akun: Jika ada masalah dengan Anda Akun AWS, seperti non-pembayaran atau pelanggaran AWS persyaratan layanan, kemampuan SNS pengiriman pesan Amazon, termasuk OTP teks, dapat ditangguhkan atau dihentikan.

Menghapus nomor telepon dari kotak pasir Amazon SNS SMS

Anda dapat menghapus nomor telepon tujuan yang tertunda dan terverifikasi dari SMSkotak pasir.

> **Important**
>
> Anda hanya dapat menghapus nomor telepon 24 jam setelah memverifikasi nomor telepon, atau 24 jam setelah upaya verifikasi terakhir Anda.

Untuk menghapus nomor telepon tujuan dari kotak SMS pasir

1. Masuk ke SNSkonsol Amazon.
2. Di menu konsol, pilih wilayah yang mendukung SMS perpesanan tempat Anda menambahkan nomor telepon tujuan.
3. Di panel navigasi, pilih Pesan teks (SMS).
4. Pada halaman Mobile text messaging (SMS), navigasikan ke bagian nomor telepon tujuan Sandbox.
5. Pilih nomor telepon tertentu yang ingin Anda hapus, lalu pilih Hapus nomor telepon.
6. Untuk mengonfirmasi bahwa Anda ingin menghapus nomor telepon, masukkan **`delete me`**, lalu pilih Delete (Hapus).

   Pastikan bahwa 24 jam atau lebih telah berlalu sejak Anda memverifikasi atau mencoba memverifikasi nomor telepon tujuan sebelum melanjutkan dengan penghapusan.

7. Ulangi langkah-langkah ini di setiap Wilayah tempat Anda menambahkan nomor telepon tujuan ini dan tidak akan menggunakannya lagi.

Pindah dari kotak SNS SMS pasir Amazon

Memindahkan Anda Akun AWS keluar dari SMSkotak pasir mengharuskan Anda menambahkan, memverifikasi, dan menguji nomor telepon tujuan terlebih dahulu. Setelah melakukan ini, buat kasus dengan AWS Support.

Untuk meminta agar AWS akun Anda dipindahkan dari kotak SMS pasir

1. Verifikasi nomor telepon
   a. Saat Anda Akun AWS berada di SMS kotak pasir, buka SNSkonsol Amazon.
   b. Di panel navigasi, di bawah Seluler, pilih Pesan teks (SMS).
   c. Di bagian nomor telepon tujuan Sandbox, tambahkan dan verifikasi satu atau beberapa nomor telepon tujuan. Verifikasi ini memastikan Anda berhasil mengirim dan menerima pesan.
2. SMSPenerbitan uji
   - Konfirmasikan bahwa Anda dapat mengirim dan menerima pesan ke setidaknya satu nomor telepon terverifikasi. Untuk petunjuk lebih rinci tentang cara mempublikasikan SMS pesan, lihatMenerbitkan SMS pesan ke ponsel menggunakan Amazon SNS.
3. Memulai suntingan kotak pasir
   - Di halaman pesan teks seluler (SMS) SNS konsol Amazon, di bawah Informasi akun, pilih Keluar dari SMS kotak pasir. Tindakan ini mengarahkan Anda ke Pusat Dukungan Amazon dan secara otomatis membuat kasus dukungan dengan opsi peningkatan kuota Layanan yang dipilih.
4. Isi formulir
   - Dalam formulir dukungan di bawah Peningkatan kuota Layanan, lakukan hal berikut:
     i. Pilih Pilih Pesan SNS Teks sebagai layanan.
     ii. Berikan nama situs web URL atau aplikasi tempat Anda ingin mengirim SMS pesan.
     iii. Tentukan jenis pesan yang akan Anda kirim: One Time Password, Promotional, atau Transactional.

iv. Pilih Wilayah AWSdari mana Anda akan mengirim SMS pesan.

v. Buat daftar negara atau wilayah tempat Anda berencana mengirim SMS pesan.

vi. Jelaskan bagaimana pelanggan Anda ikut serta untuk menerima pesan.

vii. Sertakan template pesan apa pun yang ingin Anda gunakan.

5. Tentukan kuota dan Wilayah

   - Di Requests (Permintaan), lakukan hal-hal berikut:

     i. Pilih Wilayah AWStempat yang ingin Anda pindahkan Akun AWS.

     ii. Pilih Batas Umum untuk Jenis Sumber Daya.

     iii. Pilih Exit SMS Sandbox untuk Kuota.

     iv. (Opsional) Untuk meminta kenaikan tambahan atau penyesuaian lainnya, pilih Tambahkan permintaan lain dan tentukan detail yang diperlukan.

     v. Untuk nilai kuota Baru, masukkan batas yang USD Anda minta.

6. Detail tambahan

   a. Dalam deskripsi Kasus, berikan detail tambahan yang relevan dengan permintaan Anda.

   b. Di bawah opsi Kontak, pilih bahasa kontak pilihan Anda.

7. Kirim permintaan

   - Pilih Kirim untuk mengirim permintaan Anda AWS Support.

AWS Support Tim memberikan tanggapan awal atas permintaan Anda dalam waktu 24 jam.

Untuk mencegah sistem kami digunakan untuk mengirim konten yang tidak diinginkan atau berbahaya, kami mempertimbangkan setiap permintaan dengan hati-hati. Jika bisa, kami akan mengabulkan permintaan Anda dalam waktu 24 jam ini. Namun, jika kami memerlukan informasi tambahan dari Anda, mungkin perlu waktu lebih lama untuk menyelesaikan permintaan Anda.

Jika kasus penggunaan Anda tidak sesuai dengan kebijakan kami, kami mungkin tidak dapat mengabulkan permintaan Anda.

Identitas originasi untuk pesan Amazon SNS SMS

> **Important**
>
> Panduan SNS SMS Pengembang Amazon telah diperbarui. Amazon SNS telah terintegrasi dengan AWS Olah Pesan Pengguna Akhir SMSpengiriman SMS pesan. Panduan ini berisi informasi terbaru tentang cara membuat, mengonfigurasi, dan mengelola SNS SMS pesan Amazon Anda.

Identitas originasi untuk SMS pesan adalah pengidentifikasi yang digunakan untuk mewakili pengirim pesan. SMS Anda dapat mengidentifikasi diri Anda kepada penerima menggunakan jenis identitas asal berikut:

- Nomor Originasi — String numerik yang mengidentifikasi nomor telepon pengirim SMS pesan. Ada beberapa jenis nomor originasi, termasuk kode panjang (nomor telepon standar yang biasanya memiliki 10 digit atau lebih), 10 digit kode panjang (10DLC), nomor bebas pulsa (TFN) dan kode pendek (nomor telepon yang berisi antara empat dan tujuh digit). Support untuk nomor originasi tidak tersedia di negara-negara di mana undang-undang setempat mengharuskan penggunaan pengirimIDs. Saat Anda mengirim SMS pesan menggunakan nomor originasi, perangkat penerima menunjukkan nomor originasi sebagai nomor telepon pengirim. Anda dapat menentukan nomor asal yang berbeda berdadsarkan kasus penggunaan.

  > **Tip**
  >
  > Untuk melihat daftar semua nomor originasi yang ada di AWS akun Anda, di panel navigasi SNSkonsol Amazon, pilih Nomor Originasi.

  Support untuk nomor originasi tidak tersedia di negara-negara di mana undang-undang setempat mewajibkan penggunaan pengirim, IDs bukan nomor originasi.

  Untuk informasi tambahan, lihat Nomor telepon di Panduan AWS Olah Pesan Pengguna Akhir SMS Pengguna.

- Pengirim IDs — Nama abjad yang mengidentifikasi pengirim pesan. SMS Saat Anda mengirim SMS pesan menggunakan ID pengirim, dan penerima berada di area di mana autentikasi ID pengirim didukung, ID pengirim akan muncul di perangkat penerima, bukan nomor telepon Anda.

ID pengirim memberi SMS penerima informasi lebih lanjut tentang pengirim daripada nomor telepon, kode panjang, atau kode pendek yang disediakan.

Pengirim IDs didukung di beberapa negara dan wilayah di seluruh dunia. Di beberapa tempat, jika Anda adalah bisnis yang mengirim SMS pesan ke pelanggan individu, Anda harus menggunakan ID pengirim yang telah terdaftar sebelumnya dengan badan pengatur atau grup industri. Untuk daftar lengkap negara dan wilayah yang mendukung atau memerlukan pengirimIDs, lihat Negara dan wilayah yang didukung untuk SMS pengiriman pesan AWS Olah Pesan Pengguna Akhir SMS di Panduan AWS Olah Pesan Pengguna Akhir SMS Pengguna.

Tidak ada biaya tambahan untuk menggunakan pengirimIDs. Namun, dukungan dan persyaratan untuk otentikasi ID pengirim bervariasi menurut negara. Beberapa pasar utama (termasuk Kanada, China, dan Amerika Serikat) tidak mendukung penggunaan pengirimIDs. Beberapa area mengharuskan perusahaan yang mengirim SMS pesan ke pelanggan individu harus menggunakan ID pengirim yang telah terdaftar sebelumnya dengan badan pengatur atau grup industri.

Untuk informasi tambahan, lihat Pengirim IDs di Panduan AWS Olah Pesan Pengguna Akhir SMS Pengguna.

Mengkonfigurasi SMS pesan di Amazon SNS

> **Important**
>
> Panduan SNS SMS Pengembang Amazon telah diperbarui. Amazon SNS telah terintegrasi dengan AWS Olah Pesan Pengguna Akhir SMSpengiriman SMS pesan. Panduan ini berisi informasi terbaru tentang cara membuat, mengonfigurasi, dan mengelola SNS SMS pesan Amazon Anda.

Anda dapat menggunakan konfigurasi di Amazon SNS SMS untuk mengatur SMS preferensi agar sesuai dengan kebutuhan Anda, seperti menyesuaikan kuota pengeluaran dan mengatur pencatatan status pengiriman. Topik ini juga memberikan detail tentang cara mempublikasikan SMS pesan ke topik menggunakan SNS konsol Amazon dan AWS SDK, menangani kuota secara efisien, dan mengambil statistik terperinci tentang SMS aktivitas.

Topik

- Mengirim SMS pesan menggunakan Amazon SNS

- Mengatur preferensi SMS pesan di Amazon SNS
- Mengelola nomor SNS telepon dan langganan Amazon
- Pemantauan SNS SMS aktivitas Amazon
- Meminta dukungan untuk perpesanan Amazon SNS SMS

Mengirim SMS pesan menggunakan Amazon SNS

Bagian ini menjelaskan cara mengirim SMS pesan menggunakan AmazonSNS, termasuk memublikasikan topik, berlangganan nomor telepon ke topik, menyetel atribut pada pesan, dan memublikasikan langsung ke ponsel.

Topik

- Menerbitkan SMS pesan ke SNS topik Amazon
- Menerbitkan SMS pesan ke ponsel menggunakan Amazon SNS

Menerbitkan SMS pesan ke SNS topik Amazon

Anda dapat mempublikasikan satu SMS pesan ke banyak nomor telepon sekaligus dengan berlangganan nomor telepon tersebut ke SNS topik Amazon. SNSTopik adalah saluran komunikasi tempat Anda dapat menambahkan pelanggan dan kemudian mempublikasikan pesan ke semua pelanggan tersebut. Pelanggan menerima semua pesan yang dipublikasikan ke topik sampai Anda membatalkan langganan, atau sampai pelanggan memilih untuk tidak menerima SMS pesan dari akun Anda. AWS

Topik

- Mengirim pesan ke topik menggunakan AWS konsol
- Mengirim pesan ke topik menggunakan AWS SDKs

Mengirim pesan ke topik menggunakan AWS konsol

Cara membuat topik

Selesaikan langkah-langkah berikut jika Anda belum memiliki topik yang ingin Anda kirimi SMS pesan.

1. Masuk ke SNSkonsol Amazon.

2. Di menu konsol, pilih wilayah yang mendukung SMS pengiriman pesan.
3. Di panel navigasi, pilih Topics (Topik).
4. Di halaman Topics (Topik), pilih Create topic (Buat topik).
5. Di halaman Create topic (Buat topik), pada Details (Detail), lakukan hal-hal berikut:

   a. Untuk Type (Jenis), pilih Standard (Standar).
   b. Untuk Name (Nama), masukkan nama topik.
   c. (Opsional) Untuk nama Tampilan, masukkan awalan khusus untuk SMS pesan Anda. Saat Anda mengirim pesan ke topik, Amazon menambahkan SNS nama tampilan diikuti dengan braket sudut kanan (>) dan spasi. Nama tampilan tidak peka huruf besar/kecil, dan Amazon SNS mengonversi nama tampilan menjadi karakter huruf besar. Misalnya, jika nama tampilan topiknya adalah `MyTopic` dan pesannya adalah `Hello World!`, pesan tersebut akan muncul sebagai:

      ```
      MYTOPIC> Hello World!
      ```

6. Pilih Create topic (Buat topik). Nama topik dan Nama Sumber Daya Amazon (ARN) muncul di halaman Topik.

Untuk membuat SMS langganan

Anda dapat menggunakan langganan untuk mengirim SMS pesan ke beberapa penerima dengan menerbitkan pesan hanya sekali ke topik Anda.

> **Note**
>
> Saat Anda mulai menggunakan Amazon SNS untuk mengirim SMS pesan, AWS akun Anda ada di SMSkotak pasir. SMSKotak pasir menyediakan lingkungan yang aman bagi Anda untuk mencoba SNS fitur Amazon tanpa mempertaruhkan reputasi Anda sebagai SMS pengirim. Saat akun Anda berada di SMS kotak pasir, Anda dapat menggunakan semua fitur AmazonSNS, tetapi Anda dapat mengirim SMS pesan hanya ke nomor telepon tujuan yang diverifikasi. Untuk informasi selengkapnya, lihat Menggunakan kotak SNS SMS pasir Amazon.

1. Masuk ke SNSkonsol Amazon.
2. Di panel navigasi, pilih Subscriptions (Langganan).

3. Di halaman Subscriptions (Langganan), pilih Create subscription (Buat langganan).
4. Di halaman Create subscription (Buat langganan), pada Details (Detail), lakukan hal berikut:
   a. Untuk Topik ARN, masukkan atau pilih Nama Sumber Daya Amazon (ARN) dari topik yang ingin Anda kirimi SMS pesan.
   b. Untuk Protokol, pilih SMS.
   c. Untuk Endpoint (Titik akhir), masukkan nomor telepon yang ingin Anda buat berlangganan ke topik Anda.
5. Pilih Create subscription (Buat langganan). Informasi langganan muncul di halaman Subscriptions (Langganan).

   Untuk menambahkan nomor telepon lainnya, ulangi langkah-langkah ini. Anda juga dapat menambahkan jenis langganan lainnya, seperti email.

Cara mengirim pesan

Saat Anda mempublikasikan pesan ke suatu topik, Amazon SNS mencoba mengirimkan pesan itu ke setiap nomor telepon yang berlangganan topik tersebut.

1. Di SNSkonsol Amazon, di halaman Topik, pilih nama topik yang ingin Anda kirimi SMS pesan.
2. Di halaman detail topik, pilih Publish message (Publikasikan pesan).
3. Di halaman Publish message to topic (Publikasikan pesan ke topik), di bawah Message detail (Detail pesan), lakukan hal berikut:
   a. Untuk Subjek, kosongkan bidang kecuali topik Anda berisi langganan email dan Anda ingin mempublikasikan ke email dan SMS langganan. Amazon SNS menggunakan Subjek yang Anda masukkan sebagai baris subjek email.
   b. (Opsional) Untuk Time to Live (TTL), masukkan beberapa detik yang Amazon SNS miliki untuk mengirim SMS pesan Anda ke pelanggan endpoint aplikasi seluler mana pun.
4. Di bawah Message body (Isi pesan), lakukan hal berikut:
   a. Untuk Message structure (Struktur pesan), pilih Identical payload for all delivery protocols (Muatan yang sama untuk semua protokol pengiriman) untuk mengirim pesan yang sama ke semua jenis protokol yang berlangganan topik Anda. Atau, pilih Custom payload for each delivery protocol (Muatan kustom untuk setiap protokol pengiriman) untuk menyesuaikan pesan untuk pelanggan dari jenis protokol yang berbeda. Misalnya, Anda

dapat memasukkan pesan default untuk pelanggan nomor telepon dan pesan kustom untuk pelanggan email.

b. Untuk Message body to send to the endpoint (Badan pesan yang akan dikirim ke titik akhir), masukkan pesan Anda, atau pesan kustom Anda per protokol pengiriman.

   Jika topik Anda memiliki nama tampilan, Amazon SNS menambahkannya ke pesan, yang meningkatkan panjang pesan. Panjang nama tampilan adalah jumlah karakter dalam nama ditambah dua karakter untuk braket sudut kanan (>) dan ruang yang SNS ditambahkan Amazon.

   Untuk informasi tentang kuota ukuran SMS pesan, lihatMenerbitkan SMS pesan ke ponsel menggunakan Amazon SNS.

5. (Opsional) Untuk atribut Pesan, tambahkan metadata pesan seperti cap waktu, tanda tangan, dan. IDs
6. Pilih Publish message (Publikasikan Pesan). Amazon SNS mengirimkan SMS pesan dan menampilkan pesan sukses.

Mengirim pesan ke topik menggunakan AWS SDKs

Untuk menggunakan AWS SDK, Anda harus mengkonfigurasinya dengan kredensional Anda. Untuk informasi selengkapnya, lihat File konfigurasi dan kredensial bersama di Panduan Referensi Alat AWS SDKs dan Alat.

Contoh kode berikut ini menunjukkan cara:

- Buat SNS topik Amazon.
- Berlangganan nomor telepon ke topik.
- Publikasikan SMS pesan ke topik sehingga semua nomor telepon berlangganan menerima pesan sekaligus.

Java

SDKuntuk Java 2.x

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

Buat topik dan kembalikanARN.

```
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 "";
    }
}
```

Berlangganan titik akhir ke suatu topik.

```
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);
        }
    }
}
```

Tetapkan atribut pada pesan, seperti ID pengirim, harga maksimum, dan jenisnya. Atribut pesan bersifat opsional.

```
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);
        }
    }
}
```

Publikasikan pesan ke topik. Pesan dikirim ke setiap pelanggan.

```
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);
        }
    }
}
```

Menerbitkan SMS pesan ke ponsel menggunakan Amazon SNS

Anda dapat menggunakan Amazon SNS untuk mengirim SMS pesan langsung ke ponsel tanpa berlangganan nomor telepon ke SNS topik Amazon.

> **Note**
>
> Membuat nomor telepon berlangganan ke topik berguna jika Anda ingin mengirim satu pesan ke beberapa nomor telepon sekaligus. Untuk petunjuk tentang memublikasikan SMS pesan ke suatu topik, lihatMenerbitkan SMS pesan ke SNS topik Amazon.

Saat mengirim pesan, Anda dapat mengontrol pesan dioptimalkan atau tidak untuk biayanya atau pengiriman andal. Anda juga dapat menentukan ID pengirim atau nomor asal. Jika Anda mengirim pesan secara terprogram menggunakan Amazon SNS API atau AWS SDKs, Anda dapat menentukan harga maksimum untuk pengiriman pesan.

Setiap SMS pesan dapat berisi hingga 140 byte, dan kuota karakter tergantung pada skema pengkodean. Misalnya, SMS pesan dapat berisi:

- 160 GSM karakter
- 140 ASCII karakter
- 70 UCS -2 karakter

Jika Anda memublikasikan pesan yang melebihi kuota ukuran, Amazon SNS mengirimkannya sebagai beberapa pesan, masing-masing sesuai dengan kuota ukuran. Pesan tidak terputus di tengah kata, melainkan di batas seluruh kata. Kuota ukuran total untuk satu tindakan SMS publikasi adalah 1.600 byte.

Saat Anda mengirim SMS pesan, Anda menentukan nomor telepon menggunakan format E.164, struktur penomoran telepon standar yang digunakan untuk telekomunikasi internasional. Nomor telepon yang mengikuti format ini dapat terdiri dari maksimum 15 digit bersama dengan prefiks tanda tambah (+) dan kode negara. Misalnya, nomor telepon AS dalam format E.164 muncul sebagai +1 XXX555 0100.

Topik

- Mengirim pesan (konsol)
- Mengirim pesan (AWS SDKs)

Mengirim pesan (konsol)

1. Masuk ke SNSkonsol Amazon.
2. Di menu konsol, pilih wilayah yang mendukung SMS pengiriman pesan.
3. Di panel navigasi, pilih Pesan teks (SMS).
4. Pada halaman Pesan teks seluler (SMS), pilih Publikasikan pesan teks.
5. Pada halaman Publikasikan SMS pesan, untuk jenis Pesan, pilih salah satu dari berikut ini:

   - Promotional (Promosi) – Pesan tidak penting, seperti pesan pemasaran.
   - Transactional (Transaksional) – Pesan penting yang mendukung transaksi pelanggan, seperti kode sandi satu kali (OTP) untuk autentikasi multi-faktor.

   > **Note**
   >
   > Pengaturan tingkat pesan ini menimpa jenis pesan default tingkat akun Anda. Anda dapat menyetel jenis pesan default tingkat akun dari bagian Preferensi pesan teks pada halaman Pesan teks seluler (SMS).

   Untuk informasi harga untuk pesan promosi dan transaksional, lihat Harga Seluruh Dunia SMS.
6. Untuk Destination phone number (Nomor telepon tujuan), masukkan nomor telepon yang ingin Anda kirimi pesan tersebut.
7. Untuk Message (Pesan), masukkan pesan yang akan dikirim.
8. (Opsional) Di bawah Origination identities (Identitas asal), tentukan cara mengenalkan diri Anda ke penerima:

   - Untuk menentukan Sender ID ID Pengirim, ketik ID kustom yang terdiri dari 3-11 karakter alfanumerik, termasuk setidaknya satu huruf dan tanpa spasi. ID pengirim ditampilkan sebagai pengirim pesan di perangkat penerima. Misalnya, Anda dapat menggunakan merek bisnis Anda untuk membuat sumber pesan lebih mudah dikenali.

     Support untuk pengirim IDs bervariasi menurut negara dan/atau wilayah. Misalnya, pesan yang dikirim ke nomor telepon US tidak akan menampilkan ID pengirim. Untuk negara dan wilayah yang mendukung pengirimIDs, lihat Negara dan wilayah yang didukung untuk SMS pengiriman pesan AWS Olah Pesan Pengguna Akhir SMS di Panduan AWS Olah Pesan Pengguna Akhir SMS Pengguna.

Jika Anda tidak menentukan ID pengirim, salah satu dari berikut ini akan ditampilkan sebagai identitas asal:

- Di negara-negara yang mendukung kode panjang, kode panjang akan ditampilkan.
- Di negara-negara di mana hanya pengirim IDs yang didukung, NOTICEditampilkan.

ID pengirim tingkat pesan ini menimpa ID pengirim default Anda, yang Anda tetapkan di halaman Text messaging preferences (Preferensi olahpesan teks).

- Untuk menentukan Nomor asal, masukkan string yang terdiri dari 5-14 nomor untuk ditampilkan sebagai nomor telepon pengirim di perangkat penerima. String ini harus cocok dengan nomor originasi yang dikonfigurasi di negara Akun AWS tujuan Anda. Nomor originasi dapat berupa nomor 10, DLC nomor bebas pulsa, kode person-to-person panjang, atau kode pendek. Untuk informasi selengkapnya, lihat Identitas originasi untuk pesan Amazon SNS SMS.

  Jika Anda tidak menentukan nomor originasi, Amazon SNS memilih nomor originasi yang akan digunakan untuk pesan SMS teks, berdasarkan konfigurasi Anda. Akun AWS

9. Jika Anda mengirim SMS pesan ke penerima di India, perluas atribut khusus Negara, dan tentukan atribut berikut:

   - ID Entitas — ID entitas atau ID entitas utama (PE) untuk mengirim SMS pesan ke penerima di India. ID ini adalah string unik dari 1—50 karakter yang disediakan oleh Telecom Regulatory Authority of India (TRAI) untuk mengidentifikasi entitas yang Anda daftarkan. TRAI
   - ID Template - ID template untuk mengirim SMS pesan ke penerima di India. ID ini adalah string unik TRAI yang disediakan dari 1-50 karakter yang mengidentifikasi template yang Anda daftarkan. TRAI ID templat harus dikaitkan dengan ID pengirim yang Anda tentukan untuk pesan.

   Untuk informasi selengkapnya tentang pengiriman SMS pesan ke penerima di India, proses pendaftaran ID pengirim India di AWS Olah Pesan Pengguna Akhir SMS Panduan Pengguna.

10. Pilih Publish message (Publikasikan Pesan).

> **Tip**
>
> Untuk mengirim SMS pesan dari nomor originasi, Anda juga dapat memilih Nomor Originasi di panel navigasi SNS konsol Amazon. Pilih nomor originasi yang disertakan SMSdalam kolom Kemampuan, lalu pilih Publikasikan pesan teks.

Mengirim pesan (AWS SDKs)

Untuk mengirim SMS pesan menggunakan salah satu AWS SDKs, gunakan API operasi SDK yang sesuai dengan `Publish` permintaan di Amazon SNSAPI. Dengan permintaan ini, Anda dapat mengirim SMS pesan langsung ke nomor telepon. Anda juga dapat menggunakan parameter `MessageAttributes` untuk menetapkan nilai untuk nama atribut berikut:

`AWS.SNS.SMS.SenderID`

ID kustom yang berisi 3-11 karakter alfanumerik atau karakter tanda hubung (-), termasuk setidaknya satu huruf dan tidak ada spasi. ID pengirim ditampilkan sebagai pengirim pesan di perangkat penerima. Misalnya, Anda dapat menggunakan merek bisnis Anda untuk membuat sumber pesan lebih mudah dikenali.

Support untuk pengirim IDs bervariasi menurut negara atau wilayah. Misalnya, pesan yang dikirim ke nomor telepon US tidak akan menampilkan ID pengirim. Untuk daftar negara atau wilayah yang mendukung pengirimIDs, lihat Negara dan wilayah yang didukung untuk SMS pengiriman pesan AWS Olah Pesan Pengguna Akhir SMS di Panduan AWS Olah Pesan Pengguna Akhir SMS Pengguna.

Jika Anda tidak menentukan ID pengirim, kode panjangakan ditampilkan sebagai ID pengirim di negara atau wilayah yang didukung. Untuk negara atau wilayah yang memerlukan ID pengirim alfabet, NOTICEmuncul sebagai ID pengirim.

Atribut tingkat pesan ini menimpa atribut tingkat akun `DefaultSenderID`, yang dapat Anda atur menggunakan permintaan `SetSMSAttributes`.

`AWS.MM.SMS.OriginationNumber`

Sebuah string kustom yang terdiri dari 5–14 angka, yang dapat mencakup awalan tanda tambah (+) opsional. String angka ini muncul sebagai nomor telepon pengirim di perangkat penerima. String harus cocok dengan nomor originasi yang dikonfigurasi di AWS akun Anda untuk negara tujuan. Nomor originasi dapat berupa angka 10, DLC nomor bebas pulsa, kode panjang person-

to-person (P2P), atau kode pendek. Untuk informasi selengkapnya, lihat Nomor telepon di Panduan AWS Olah Pesan Pengguna Akhir SMS Pengguna.

Jika Anda tidak menentukan nomor originasi, Amazon SNS memilih nomor originasi berdasarkan konfigurasi akun Anda AWS .

`AWS.SNS.SMS.MaxPrice`

Harga maksimum USD yang bersedia Anda keluarkan untuk mengirim SMS pesan. Jika Amazon SNS menentukan bahwa mengirim pesan akan dikenakan biaya yang melebihi harga maksimum Anda, itu tidak mengirim pesan.

Atribut ini tidak berpengaruh jika month-to-date SMS biaya Anda telah melebihi kuota yang ditetapkan untuk atribut tersebut. `MonthlySpendLimit` Anda dapat mengatur atribut `MonthlySpendLimit` menggunakan permintaan `SetSMSAttributes`.

Jika Anda mengirim pesan ke SNS topik Amazon, harga maksimum berlaku untuk setiap pengiriman pesan ke setiap nomor telepon yang berlangganan topik tersebut.

`AWS.SNS.SMS.SMSType`

Jenis pesan yang Anda kirim:

- `Promotional` (default) – Pesan tidak penting, seperti pesan pemasaran.
- `Transactional` (Transaksional) – Pesan penting yang mendukung transaksi pelanggan, seperti kode sandi satu kali (OTP) untuk autentikasi multi-faktor.

Atribut tingkat pesan ini menimpa atribut tingkat akun `DefaultSMSType`, yang dapat Anda atur menggunakan permintaan `SetSMSAttributes`.

`AWS.MM.SMS.EntityId`

Atribut ini hanya diperlukan untuk mengirim SMS pesan ke penerima di India.

Ini adalah ID entitas atau ID entitas utama (PE) Anda untuk mengirim SMS pesan ke penerima di India. ID ini adalah string unik dari 1—50 karakter yang disediakan oleh Telecom Regulatory Authority of India (TRAI) untuk mengidentifikasi entitas yang Anda daftarkan. TRAI

`AWS.MM.SMS.TemplateId`

Atribut ini hanya diperlukan untuk mengirim SMS pesan ke penerima di India.

Ini adalah template Anda untuk mengirim SMS pesan ke penerima di India. ID ini adalah string unik TRAI yang disediakan dari 1-50 karakter yang mengidentifikasi template yang Anda daftarkan. TRAI ID templat harus dikaitkan dengan ID pengirim yang Anda tentukan untuk pesan.

Mengirim pesan

Contoh kode berikut menunjukkan cara mempublikasikan SMS pesan menggunakan AmazonSNS.

.NET

AWS SDK for .NET

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

```
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}");
            }
        }
    }
}
```

- Untuk API detailnya, lihat Publikasikan di AWS SDK for .NET APIReferensi.

C++

SDKuntuk C ++

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

```
/**
 * Publish SMS: use Amazon Simple Notification Service (Amazon SNS) to send an
 SMS text message to a phone number.
 * Note: This requires additional AWS configuration prior to running example.
 *
 *  NOTE: When you start using Amazon SNS to send SMS messages, your AWS account
 is in the SMS sandbox and you can only
 *  use verified destination phone numbers. See https://docs.aws.amazon.com/sns/
latest/dg/sns-sms-sandbox.html.
 *  NOTE: If destination is in the US, you also have an additional restriction
 that you have use a dedicated
 *  origination ID (phone number). You can request an origination number using
 Amazon Pinpoint for a fee.
 *  See https://aws.amazon.com/blogs/compute/provisioning-and-using-10dlc-
origination-numbers-with-amazon-sns/
 *  for more information.
 *
 *  <phone_number_value> input parameter uses E.164 format.
 *  For example, in United States, this input value should be of the form:
 +12223334444
 */

//! Send an SMS text message to a phone number.
/*!
  \param message: The message to publish.
  \param phoneNumber: The phone number of the recipient in E.164 format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::publishSms(const Aws::String &message,
                             const Aws::String &phoneNumber,
```

```
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetPhoneNumber(phoneNumber);

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

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

    return outcome.IsSuccess();
}
```

- Untuk API detailnya, lihat Publikasikan di AWS SDK for C++ APIReferensi.

Java

SDKuntuk Java 2.x

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

```
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);
        }
    }
}
```

- Untuk API detailnya, lihat Publikasikan di AWS SDK for Java 2.x APIReferensi.

Kotlin

SDKuntuk Kotlin

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

```
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.")
    }
}
```

- Untuk API detailnya, lihat Publish in AWS SDKuntuk API referensi Kotlin.

PHP

SDKuntuk PHP

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

```
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());
}
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for PHP.
- Untuk API detailnya, lihat Publikasikan di AWS SDK for PHP APIReferensi.

Python

SDKuntuk Python (Boto3)

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

```
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
```

- Untuk API detailnya, lihat Publikasikan AWS SDKuntuk Referensi Python (Boto3). API

Mengatur preferensi SMS pesan di Amazon SNS

Gunakan Amazon SNS untuk menentukan preferensi SMS pesan. Misalnya, Anda dapat menentukan apakah akan mengoptimalkan pengiriman untuk biaya atau keandalan, batas pengeluaran bulanan Anda, bagaimana pengiriman dicatat, dan apakah akan berlangganan laporan SMS penggunaan harian.

Preferensi ini berlaku untuk setiap SMS pesan yang Anda kirim dari akun Anda, tetapi Anda dapat mengganti beberapa dari mereka ketika Anda mengirim pesan individual. Untuk informasi selengkapnya, lihat Menerbitkan SMS pesan ke ponsel menggunakan Amazon SNS.

Topik

- Menyetel preferensi SMS pesan menggunakan AWS Management Console
- Pengaturan preferensi (AWS SDKs)
- Menyetel preferensi SMS pesan untuk pengiriman khusus negara

Menyetel preferensi SMS pesan menggunakan AWS Management Console

1. Masuk ke SNSkonsol Amazon.
2. Pilih wilayah yang mendukung SMS pengiriman pesan.
3. Pada panel navigasi, pilih Seluler dan kemudian Pesan teks (SMS).

4. Pada halaman Pesan teks seluler (SMS), di bagian Preferensi pesan teks, pilih Edit.
5. Di halaman Edit text messaging preferences (Edit preferensi olahpesan teks), di bagian Details, lakukan hal berikut:

   a. Untuk Default message type (Jenis pesan bawaan), pilih salah satu jenis berikut:

      - Promosi — Pesan non-kritis (misalnya, pemasaran). Amazon SNS mengoptimalkan pengiriman pesan untuk mengeluarkan biaya terendah.
      - Transaksional (default) — Pesan penting yang mendukung transaksi pelanggan, seperti kode sandi satu kali untuk otentikasi multi-faktor. Amazon SNS mengoptimalkan pengiriman pesan untuk mencapai keandalan tertinggi.

      Untuk informasi harga untuk pesan promosi dan transaksional, lihat Harga Global SMS.

   b. (Opsional) Untuk batas pengeluaran Akun, masukkan jumlah (dalamUSD) yang ingin Anda belanjakan untuk SMS pesan setiap bulan kalender.

      > **Important**
      >
      > - Secara default, kuota pembelanjaan diatur ke USD 1,00. Jika Anda ingin meningkatkan kuota layanan, kirimkan permintaan.
      > - Jika jumlah yang ditetapkan di konsol melebihi kuota layanan Anda, Amazon SNS berhenti menerbitkan SMS pesan.
      > - Karena Amazon SNS adalah sistem terdistribusi, Amazon berhenti mengirim SMS pesan dalam beberapa menit setelah kuota pengeluaran terlampaui. Selama interval ini, jika Anda terus mengirim SMS pesan, Anda mungkin dikenakan biaya yang melebihi kuota Anda.

6. (Opsional) Untuk Default sender ID (ID pengirim default), masukkan ID kustom, seperti merek bisnis Anda, yang ditampilkan sebagai pengirim di perangkat penerima.

   > **Note**
   >
   > Support untuk pengirim IDs bervariasi menurut negara.

7. (Opsional) Masukkan nama Nama bucket Amazon S3 untuk laporan penggunaan.

> **Note**
>
> Kebijakan bucket S3 harus memberikan akses tulis ke AmazonSNS.

8. Pilih Simpan perubahan.

Pengaturan preferensi (AWS SDKs)

Untuk mengatur SMS preferensi Anda menggunakan salah satu AWS SDKs, gunakan tindakan SDK yang sesuai dengan `SetSMSAttributes` permintaan di Amazon SNSAPI. Dengan permintaan ini, Anda menetapkan nilai ke SMS atribut yang berbeda, seperti kuota belanja bulanan dan SMS tipe default Anda (promosi atau transaksional). Untuk semua SMS atribut, lihat S etSMSAttributes di APIReferensi Layanan Pemberitahuan Sederhana Amazon.

Contoh kode berikut menunjukkan cara menggunakan`SetSMSAttributes`.

C++

SDKuntuk C ++

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

Cara menggunakan Amazon SNS untuk mengatur efaultSMSType atribut D.

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

- Untuk API detailnya, lihat S etSMSAttributes di AWS SDK for C++ APIReferensi.

CLI

AWS CLI

Untuk mengatur atribut SMS pesan

`set-sms-attributes`Contoh berikut menetapkan ID pengirim default untuk SMS pesan ke`MyName`.

```
aws sns set-sms-attributes \
    --attributes DefaultSenderID=MyName
```

Perintah ini tidak menghasilkan output.

- Untuk API detailnya, lihat S etSMSAttributes di Referensi AWS CLI Perintah.

Java

SDKuntuk Java 2.x

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

```
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);
        }
    }
}
```

- Untuk API detailnya, lihat S etSMSAttributes di AWS SDK for Java 2.x APIReferensi.

JavaScript

SDKuntuk JavaScript (v3)

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

Buat klien dalam modul terpisah dan ekspor klien tersebut.

```
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({});
```

Impor modul SDK dan klien dan panggil fileAPI.

```
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;
};
```

- Untuk informasi selengkapnya, lihat AWS SDK for JavaScript Panduan Developer.
- Untuk API detailnya, lihat S etSMSAttributes di AWS SDK for JavaScript APIReferensi.

PHP

SDKuntuk PHP

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

```
$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());
}
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for PHP.
- Untuk API detailnya, lihat S etSMSAttributes di AWS SDK for PHP APIReferensi.

Menyetel preferensi SMS pesan untuk pengiriman khusus negara

Anda dapat mengelola dan mengontrol SMS lalu lintas Anda dengan mengirim pesan hanya ke negara tujuan tertentu. Ini memastikan bahwa pesan Anda dikirim hanya ke negara yang disetujui, menghindari SMS biaya yang tidak diinginkan. Petunjuk berikut menggunakan konfigurasi Protect Amazon Pinpoint untuk menentukan negara yang ingin Anda izinkan atau blokir.

1. Buka AWS SMS konsol di https://console.aws.amazon.com/sms-voice/.
2. Di panel navigasi, di bawah Ikhtisar, di bagian Mulai cepat, pilih Buat konfigurasi proteksi.
3. Di bawah Lindungi detail konfigurasi, masukkan nama yang ramah bisnis untuk konfigurasi proteksi Anda (misalnya, Allow-Only-AU).
4. Di bawah aturan SMS negara, pilih kotak centang Wilayah/Negara untuk memblokir pengiriman pesan ke semua negara yang didukung.
5. Hapus centang kotak untuk negara tempat Anda ingin mengirim pesan. Misalnya, untuk mengizinkan pesan hanya ke Australia, batalkan centang kotak untuk Australia.
6. Di bagian Lindungi asosiasi konfigurasi, di bawah Jenis asosiasi, pilih Akun default. Ini akan memastikan bahwa konfigurasi AWS Olah Pesan Pengguna Akhir SMS Protect memengaruhi semua pesan yang dikirim melalui AmazonSNS, Amazon Cognito, dan panggilan Amazon `SendMessages`APIPinpoint.
7. Pilih Buat konfigurasi proteksi untuk menyimpan pengaturan Anda.

   Pesan konfirmasi berikut ditampilkan:

   ```
   Success Protect configuration protect-abc0123456789 has been created.
   ```

8. Masuk ke SNSkonsol Amazon.
9. Publikasikan pesan ke salah satu negara yang diblokir, seperti India.

   Pesan tidak akan terkirim. Anda dapat memverifikasi ini di log kegagalan pengiriman menggunakan CloudWatch. Cari grup log sns/region/AccountID/DirectPublishToPhoneNumber/Failureuntuk respons yang mirip dengan contoh berikut:

   ```
   {
   "notification": {
   "messageId": "bd59a509-XXXX-XXXX-82f8-fbdb8cb68217",
   "timestamp": "YYYY-MM-DD XX:XX:XX.XXXX“
   },
   "delivery": {
   "destination": "+91XXXXXXXXXX",
   "smsType": "Transactional",
   "providerResponse": "Cannot deliver message to the specified destination country",
   "dwellTimeMs": 85
   },
   "status": "FAILURE"
   }
   ```

Mengelola nomor SNS telepon dan langganan Amazon

Amazon SNS menyediakan beberapa opsi untuk mengelola siapa yang menerima SMS pesan dari akun Anda. Dengan frekuensi terbatas, Anda dapat memilih nomor telepon yang telah memilih untuk tidak menerima SMS pesan dari akun Anda. Untuk berhenti mengirim pesan ke SMS langganan, Anda dapat menghapus langganan atau topik yang mempublikasikannya.

Topik

Memilih keluar dari menerima pesan SMS

Jika diwajibkan oleh undang-undang dan peraturan setempat (seperti AS dan Kanada), SMS penerima dapat menggunakan perangkat mereka untuk memilih keluar dengan membalas pesan dengan salah satu dari berikut ini:

- ARRET(Perancis)
- CANCEL
- END
- OPT-OUT
- OPTOUT
- QUIT
- REMOVE
- STOP
- TD
- UNSUBSCRIBE

Untuk memilih keluar, penerima harus membalas nomor originasi yang sama dengan yang SNS digunakan Amazon untuk menyampaikan pesan. Setelah memilih keluar, penerima tidak akan lagi menerima SMS pesan yang dikirim dari Anda Akun AWS kecuali Anda memilih nomor telepon.

Jika nomor telepon berlangganan SNS topik Amazon, memilih keluar tidak menghapus langganan, tetapi SMS pesan akan gagal dikirimkan ke langganan itu kecuali Anda memilih nomor telepon.

Mengelola nomor telepon dan langganan menggunakan konsol Amazon SNS

Anda dapat menggunakan SNS konsol Amazon untuk mengontrol nomor telepon mana yang menerima SMS pesan dari akun Anda.

Memilih nomor telepon yang telah memilih keluar dari konsol Amazon SNS

Anda dapat melihat nomor telepon mana yang telah dipilih untuk tidak menerima SMS pesan dari akun Anda, dan Anda dapat memilih nomor telepon ini untuk melanjutkan pengiriman pesan kepada mereka.

Anda dapat memilih nomor telepon hanya sekali setiap 30 hari.

1. Masuk ke SNSkonsol Amazon.
2. Di menu konsol, atur pemilih wilayah ke wilayah yang mendukung SMS pengiriman pesan.
3. Pada panel navigasi, pilih Pesan teks (SMS).
4. Pada halaman Pesan teks seluler (SMS), di bagian Nomor telepon yang dipilih keluar, nomor telepon yang dipilih akan ditampilkan.
5. Pilih kotak centang untuk nomor telepon yang ingin Anda pilih, dan pilih Opt in. Nomor telepon tidak lagi memilih keluar dan akan menerima SMS pesan yang Anda kirim ke sana.

Menghapus SMS langganan konsol Amazon SNS

Hapus SMS langganan untuk berhenti mengirim SMS pesan ke nomor telepon tersebut saat Anda mempublikasikan ke topik Anda.

1. Di panel navigasi, pilih Subscriptions (Langganan).
2. Pilih kotak centang untuk langganan yang ingin Anda hapus. Lalu pilih Actions (Tindakan), dan pilih Delete Subscriptions (Hapus Langganan).
3. Di jendela Delete (Hapus), pilih Delete (Hapus). Amazon SNS menghapus langganan dan menampilkan pesan sukses.

Menghapus topik konsol Amazon SNS

Menghapus topik ketika Anda tidak ingin lagi menerbitkan pesan ke titik akhir berlangganan.

1. Di panel navigasi, pilih Topics (Topik).

2. Pilih kotak centang untuk topik yang ingin Anda hapus. Lalu pilih Actions (Tindakan), dan pilih Delete Topics (Hapus Topik).
3. Di jendela Delete (Hapus), pilih Delete (Hapus). Amazon SNS menghapus topik dan menampilkan pesan sukses.

Mengelola nomor telepon dan langganan menggunakan AWS SDK

Anda dapat menggunakan file AWS SDKs untuk membuat permintaan terprogram ke Amazon SNS dan mengelola nomor telepon mana yang dapat menerima SMS pesan dari akun Anda.

Untuk menggunakan AWS SDK, Anda harus mengkonfigurasinya dengan kredensi Anda. Untuk informasi selengkapnya, lihat File konfigurasi dan kredensial bersama di Panduan Referensi Alat AWS SDKsdan Alat.

Melihat semua nomor telepon opted-out menggunakan AWS SDK

Untuk melihat semua nomor telepon yang dipilih keluar, kirimkan `ListPhoneNumbersOptedOut` permintaan dengan Amazon. SNS API

Contoh kode berikut menunjukkan cara menggunakan`ListPhoneNumbersOptedOut`.

CLI

AWS CLI

Untuk mencantumkan SMS opt-out pesan

`list-phone-numbers-opted-out`Contoh berikut mencantumkan nomor telepon yang dipilih untuk tidak menerima SMS pesan.

```
aws sns list-phone-numbers-opted-out
```

Output:

```
{
    "phoneNumbers": [
        "+15555550100"
    ]
}
```

- Untuk API detailnya, lihat ListPhoneNumbersOptedOutdi Referensi AWS CLI Perintah.

Java

SDKuntuk Java 2.x

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

```
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);
        }
    }
}
```

- Untuk API detailnya, lihat ListPhoneNumbersOptedOutdi AWS SDK for Java 2.x APIReferensi.

PHP

SDKuntuk PHP

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

```
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());
}
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for PHP.
- Untuk API detailnya, lihat ListPhoneNumbersOptedOutdi AWS SDK for PHP APIReferensi.

Memeriksa apakah nomor telepon dipilih keluar menggunakan AWS SDK

Untuk memeriksa apakah nomor telepon dipilih keluar, kirimkan `CheckIfPhoneNumberIsOptedOut` permintaan ke Amazon. SNS API

Contoh kode berikut menunjukkan cara menggunakan`CheckIfPhoneNumberIsOptedOut`.

.NET

AWS SDK for .NET

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

```
    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}");
            }
        }
    }
```

- Untuk API detailnya, lihat CheckIfPhoneNumberIsOptedOutdi AWS SDK for .NET APIReferensi.

CLI

AWS CLI

Untuk memeriksa SMS pesan opt-out untuk nomor telepon

`check-if-phone-number-is-opted-out`Contoh berikut memeriksa apakah nomor telepon yang ditentukan dipilih untuk tidak menerima SMS pesan dari AWS akun saat ini.

```
aws sns check-if-phone-number-is-opted-out \
    --phone-number +1555550100
```

Output:

```
{
    "isOptedOut": false
}
```

- Untuk API detailnya, lihat CheckIfPhoneNumberIsOptedOutdi Referensi AWS CLI Perintah.

Java

SDKuntuk Java 2.x

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

```
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);
        }
    }
}
```

- Untuk API detailnya, lihat CheckIfPhoneNumberIsOptedOutdi AWS SDK for Java 2.x APIReferensi.

JavaScript

SDKuntuk JavaScript (v3)

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

Buat klien dalam modul terpisah dan ekspor klien tersebut.

```
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({});
```

Impor modul SDK dan klien dan panggil fileAPI.

```
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;
};
```

- Untuk informasi selengkapnya, lihat AWS SDK for JavaScript Panduan Developer.
- Untuk API detailnya, lihat CheckIfPhoneNumberIsOptedOutdi AWS SDK for JavaScript APIReferensi.

PHP

SDKuntuk PHP

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

```
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());
}
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for PHP.
- Untuk API detailnya, lihat CheckIfPhoneNumberIsOptedOutdi AWS SDK for PHP APIReferensi.

Memilih nomor telepon yang telah dipilih keluar menggunakan Amazon SNS API

Untuk memilih nomor telepon, kirimkan `OptInPhoneNumber` permintaan dengan Amazon SNSAPI.

Anda dapat memilih nomor telepon hanya sekali setiap 30 hari.

Menghapus SMS langganan menggunakan AWS SDK

Untuk menghapus SMS langganan dari SNS topik Amazon, dapatkan langganan ARN dengan mengirimkan `ListSubscriptions` permintaan ke Amazon SNSAPI, lalu teruskan ARN ke permintaan. `Unsubscribe`

Contoh kode berikut menunjukkan cara menggunakan`Unsubscribe`.

.NET

AWS SDK for .NET

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

Berhenti berlangganan dari topik dengan berlangganan ARN.

```
/// <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;
}
```

- Untuk API detailnya, lihat Berhenti berlangganan di AWS SDK for .NET API Referensi.

C++

SDK untuk C ++

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

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

    Aws::SNS::Model::UnsubscribeRequest request;
    request.SetSubscriptionArn(subscriptionARN);

    const Aws::SNS::Model::UnsubscribeOutcome outcome =
 snsClient.Unsubscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Unsubscribed successfully " << std::endl;
    }
    else {
        std::cerr << "Error while unsubscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Untuk API detailnya, lihat Berhenti berlangganan di AWS SDK for C++ APIReferensi.

CLI

AWS CLI

Untuk berhenti berlangganan dari suatu topik

`unsubscribe`Contoh berikut menghapus langganan yang ditentukan dari suatu topik.

```
aws sns unsubscribe \
    --subscription-arn arn:aws:sns:us-west-2:0123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f
```

Perintah ini tidak menghasilkan output.

- Untuk API detailnya, lihat Berhenti berlangganan di Referensi AWS CLI Perintah.

Java

SDKuntuk Java 2.x

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

```
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);
        }
    }
}
```

- Untuk API detailnya, lihat Berhenti berlangganan di AWS SDK for Java 2.x APIReferensi.

JavaScript

SDKuntuk JavaScript (v3)

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

Buat klien dalam modul terpisah dan ekspor klien tersebut.

```
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({});
```

Impor modul SDK dan klien dan panggil fileAPI.

```
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;
};
```

- Untuk informasi selengkapnya, lihat AWS SDK for JavaScript Panduan Developer.
- Untuk API detailnya, lihat Berhenti berlangganan di AWS SDK for JavaScript APIReferensi.

Kotlin

SDKuntuk Kotlin

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

```
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}")
    }
}
```

- Untuk API detailnya, lihat Berhenti berlangganan AWS SDKuntuk referensi Kotlin API.

PHP

SDKuntuk PHP

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

```
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());
}
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for PHP.
- Untuk API detailnya, lihat Berhenti berlangganan di AWS SDK for PHP APIReferensi.

Python

SDKuntuk Python (Boto3)

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

```
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
```

- Untuk API detailnya, lihat Berhenti berlangganan AWS SDKuntuk Referensi Python (APIBoto3).

SAP ABAP

SDKuntuk SAP ABAP

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

```
    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.
```

- Untuk API detailnya, lihat Berhenti berlangganan AWS SDKuntuk SAP ABAP API referensi.

Menghapus topik menggunakan AWS SDK

Untuk menghapus topik dan semua langganannya, dapatkan topik ARN dengan mengirimkan `ListTopics` permintaan ke Amazon SNSAPI, lalu teruskan ARN ke permintaan. `DeleteTopic`

Contoh kode berikut menunjukkan cara menggunakan`DeleteTopic`.

.NET

AWS SDK for .NET

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

Hapus topik berdasarkan topiknyaARN.

```
/// <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;
}
```

- Untuk API detailnya, lihat DeleteTopicdi AWS SDK for .NET APIReferensi.

C++

SDKuntuk C ++

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

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

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

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

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

    return outcome.IsSuccess();
}
```

- Untuk API detailnya, lihat DeleteTopicdi AWS SDK for C++ APIReferensi.

CLI

AWS CLI

Untuk menghapus SNS topik

`delete-topic`Contoh berikut menghapus SNS topik yang ditentukan.

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

Perintah ini tidak menghasilkan output.

- Untuk API detailnya, lihat DeleteTopicdi Referensi AWS CLI Perintah.

Go

SDKuntuk Go V2

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

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(ctx context.Context, topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(ctx, &sns.DeleteTopicInput{
  TopicArn: aws.String(topicArn)})
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}
```

- Untuk API detailnya, lihat DeleteTopicdi AWS SDK for Go APIReferensi.

Java

SDKuntuk Java 2.x

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

```
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);
        }
    }
}
```

- Untuk API detailnya, lihat DeleteTopicdi AWS SDK for Java 2.x APIReferensi.

JavaScript

SDKuntuk JavaScript (v3)

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

Buat klien dalam modul terpisah dan ekspor klien tersebut.

```
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({});
```

Impor modul SDK dan klien dan panggil fileAPI.

```
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
  //   }
  // }
};
```

- Untuk informasi selengkapnya, lihat AWS SDK for JavaScript Panduan Developer.
- Untuk API detailnya, lihat DeleteTopicdi AWS SDK for JavaScript APIReferensi.

Kotlin

SDKuntuk Kotlin

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

```
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.")
    }
}
```

- Untuk API detailnya, lihat DeleteTopic AWSSDKAPIreferensi Kotlin.

PHP

SDKuntuk PHP

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

```
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());
}
```

- Untuk API detailnya, lihat DeleteTopicdi AWS SDK for PHP APIReferensi.

Python

SDKuntuk Python (Boto3)

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

```
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
```

- Untuk API detailnya, lihat DeleteTopic AWSSDKReferensi Python (Boto3). API

SAP ABAP

SDKuntuk SAP ABAP

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

```
    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.
```

- Untuk API detailnya, lihat DeleteTopic AWSSDKuntuk SAP ABAP API referensi.

Pemantauan SNS SMS aktivitas Amazon

Dengan memantau SMS aktivitas Anda, Anda dapat melacak nomor telepon tujuan, pengiriman yang berhasil atau gagal, alasan kegagalan, biaya, dan informasi lainnya. Amazon SNS membantu dengan meringkas statistik di konsol, mengirim informasi ke Amazon CloudWatch, dan mengirim laporan SMS penggunaan harian ke bucket Amazon S3 yang Anda tentukan.

Topik

- Melihat statistik SNS SMS pengiriman Amazon
- Pemantauan SNS SMS pengiriman Amazon dengan CloudWatch metrik dan log Amazon
- Berlangganan laporan SMS penggunaan SNS harian Amazon

Melihat statistik SNS SMS pengiriman Amazon

Anda dapat menggunakan SNS konsol Amazon untuk melihat statistik tentang SMS pengiriman terbaru Anda.

1. Masuk ke SNSkonsol Amazon.
2. Di menu konsol, atur pemilih wilayah ke wilayah yang mendukung SMS pengiriman pesan.
3. Pada panel navigasi, pilih Pesan teks (SMS).
4. Pada halaman Pesan teks (SMS), di bagian Statistik akun, lihat grafik untuk pengiriman pesan transaksional dan promosi SMS Anda. Setiap grafik menunjukkan data berikut selama 15 hari sebelumnya:
   - Tingkat pengiriman (persentase pengiriman yang berhasil)
   - Terkirim (jumlah upaya pengiriman)
   - Gagal (jumlah kegagalan pengiriman)

Di halaman ini, Anda juga dapat memilih tombol Usage (Penggunaan) untuk membuka bucket Amazon S3 tempat Anda menyimpan laporan penggunaan harian Anda. Untuk informasi selengkapnya, lihat Berlangganan laporan SMS penggunaan SNS harian Amazon.

Pemantauan SNS SMS pengiriman Amazon dengan CloudWatch metrik dan log Amazon

Anda dapat menggunakan Amazon CloudWatch dan Amazon CloudWatch Logs untuk memantau pengiriman SMS pesan Anda.

Topik

Melihat CloudWatch metrik Amazon

Amazon SNS secara otomatis mengumpulkan metrik tentang pengiriman SMS pesan Anda dan mendorongnya ke Amazon. CloudWatch Anda dapat menggunakan CloudWatch untuk memantau metrik ini dan membuat alarm untuk mengingatkan Anda ketika metrik melewati ambang batas. Misalnya, Anda dapat memantau CloudWatch metrik untuk mempelajari tarif SMS pengiriman dan month-to-date SMS biaya Anda.

Untuk informasi tentang CloudWatch metrik pemantauan, pengaturan CloudWatch alarm, dan jenis metrik yang tersedia, lihat. Memantau SNS topik Amazon menggunakan CloudWatch

Melihat CloudWatch Log

Anda dapat mengumpulkan informasi tentang pengiriman SMS pesan yang berhasil dan tidak berhasil dengan memungkinkan Amazon menulis ke SNS Amazon CloudWatch Logs. Untuk setiap SMS pesan yang Anda kirim, Amazon SNS menulis log yang mencakup harga pesan, status keberhasilan atau kegagalan, alasan kegagalan (jika pesan gagal), waktu tinggal pesan, dan informasi lainnya.

Untuk mengaktifkan dan melihat CloudWatch Log untuk SMS pesan Anda

1. Masuk ke SNSkonsol Amazon.
2. Di menu konsol, atur pemilih wilayah ke wilayah yang mendukung SMS pengiriman pesan.
3. Pada panel navigasi, pilih Pesan teks (SMS).
4. Pada halaman Pesan teks seluler (SMS), di bagian Preferensi pesan teks, pilih Edit.
5. Di halaman berikutnya, perluas bagian Delivery status logging (Pencatatan status pengiriman).

6. Untuk tingkat sampel Sukses, tentukan persentase SMS pengiriman yang berhasil di mana Amazon SNS akan menulis CloudWatch log di Log. Sebagai contoh:

   - Untuk menulis log hanya untuk pengiriman yang gagal, atur nilai ini ke 0.
   - Untuk menulis log untuk 10% dari pengiriman yang berhasil, atur nilai ke 10.

   Jika Anda tidak menentukan persentase, Amazon SNS menulis log untuk semua pengiriman yang berhasil.
7. Untuk memberikan izin yang diperlukan, lakukan salah satu hal berikut:

   - Untuk membuat peran layanan baru, pilih Create new service role (Buat peran layanan baru) dan kemudian Create new roles (Buat peran baru). Di halaman berikutnya, pilih Izinkan untuk memberi Amazon akses SNS tulis ke sumber daya akun Anda.
   - Untuk menggunakan peran layanan yang ada, pilih Gunakan peran layanan yang ada, lalu tempelkan ARN nama di kotak IAMperan untuk pengiriman yang berhasil dan gagal.

     Peran layanan yang Anda tentukan harus mengizinkan akses tulis ke sumber daya akun Anda. Untuk informasi selengkapnya tentang membuat IAM peran, lihat Membuat peran untuk AWS layanan di Panduan IAM Pengguna.
8. Pilih Simpan perubahan.
9. Kembali ke halaman Mobile text messaging (SMS), buka bagian Log status pengiriman untuk melihat log yang tersedia.

> **Note**
>
> Bergantung pada operator nomor telepon tujuan, diperlukan waktu hingga 72 jam agar log pengiriman muncul di SNS konsol Amazon.

Contoh log untuk SMS pengiriman yang berhasil

Log status pengiriman untuk SMS pengiriman yang berhasil akan menyerupai contoh berikut:

```
{
    "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"
}
```

Contoh log untuk SMS pengiriman yang gagal

Log status pengiriman untuk SMS pengiriman yang gagal akan menyerupai contoh berikut:

```
{
    "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"
}
```

SMSalasan kegagalan pengiriman

Alasan kegagalan diberikan dengan atribut `providerResponse`. SMSpesan mungkin gagal disampaikan karena alasan berikut:

- Diblokir sebagai spam oleh operator telepon
- Tujuan ada di daftar yang diblokir
- Nomor telepon tidak valid
- Isi pesan tidak valid
- Operator telepon telah memblokir pesan ini
- Operator telepon saat ini tidak dapat dihubungi/tidak tersedia
- Ponsel telah diblokir SMS
- Telepon ada dalam daftar yang diblokir
- Telepon saat ini tidak dapat dihubungi/tidak tersedia
- Nomor telepon memilih tidak menerima pesan
- Pengiriman ini akan melebihi harga maksimum
- Kesalahan tak diketahui yang mencoba menjangkau telepon

Berlangganan laporan SMS penggunaan SNS harian Amazon

Anda dapat memantau SMS pengiriman Anda dengan berlangganan laporan penggunaan harian dari Amazon. SNS Untuk setiap hari Anda mengirim setidaknya satu SMS pesan, Amazon SNS mengirimkan laporan penggunaan sebagai CSV file ke bucket Amazon S3 yang ditentukan. Dibutuhkan 24 jam agar laporan SMS penggunaan tersedia di bucket S3.

Topik

Informasi laporan penggunaan harian

Laporan penggunaan mencakup informasi berikut untuk setiap SMS pesan yang Anda kirim dari akun Anda.

Perhatikan bahwa laporan ini tidak menyertakan pesan yang dikirim ke penerima yang telah memilih untuk tidak menerima pesan.

- Waktu publikasi untuk pesan (inUTC)
- ID Pesan

- Nomor telepon tujuan
- Jenis pesan
- Status pengiriman
- Harga pesan (inUSD)
- Jumlah bagian (pesan dibagi menjadi beberapa bagian jika terlalu panjang untuk satu pesan)
- Jumlah total bagian

> **Note**
>
> Jika Amazon SNS tidak menerima nomor bagian, kami menetapkan nilainya ke nol.

Berlangganan laporan penggunaan harian

Untuk berlangganan laporan penggunaan harian, Anda harus membuat bucket Amazon S3 dengan izin yang sesuai.

Cara membuat bucket Amazon S3 untuk laporan penggunaan harian Anda

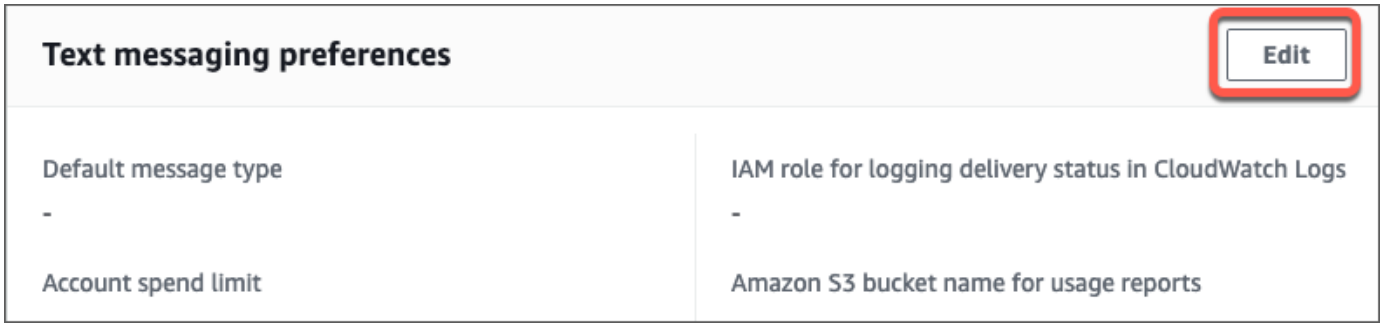
1. Dari Akun AWS yang mengirim SMS pesan, masuk ke konsol Amazon S3.
2. Pilih Create Bucket (Buat Bucket).
3. Untuk Bucket Name (Nama Bucket), sebaiknya masukkan nama yang unik untuk akun dan organisasi Anda. Misalnya, gunakan pola `<my-bucket-prefix>-<account_id>-<org-id>`.

   Untuk informasi tentang konvensi dan batasan untuk nama bucket, lihat Aturan untuk Penamaan Bucket di Panduan Pengguna Layanan Penyimpanan Sederhana Amazon.
4. Pilih Buat.
5. Di tabel All Buckets (Semua Bucket), pilih nama bucket.
6. Di tab Permission (Izin), pilih Bucket policy (Kebijakan bucket).
7. Di jendela Bucket Policy Editor, berikan kebijakan yang memungkinkan prinsipal SNS layanan Amazon menulis ke bucket Anda. Sebagai contoh, lihat Contoh kebijakan bucket.

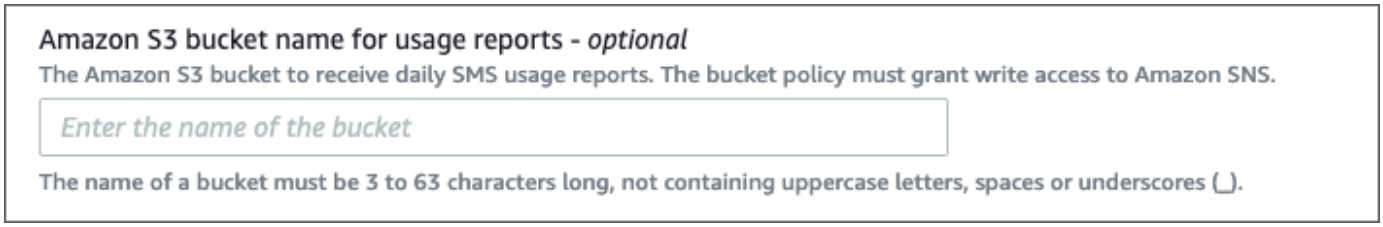
   Jika Anda menggunakan kebijakan contoh, ingatlah untuk mengganti *`my-s3-bucket`* dengan nama bucket yang Anda pilih di Langkah 3.
8. Pilih Simpan.

Cara berlangganan laporan penggunaan harian

1. Masuk ke SNSkonsol Amazon.
2. Pada panel navigasi, pilih Pesan teks (SMS).
3. Pada halaman Pesan teks (SMS), di bagian Preferensi pesan teks, pilih Edit.



4. Di halaman Edit text messaging preferences (Edit preferensi olahpesan teks), di bagian Details (Detail), tentukan Nama bucket Amazon S3 untuk laporan penggunaan.



5. Pilih Simpan perubahan.

Contoh kebijakan bucket

Kebijakan berikut memungkinkan prinsipal SNS layanan Amazon untuk melakukan`s3:PutObject`,`s3:GetBucketLocation`, dan `s3:ListBucket` tindakan.

AWS menyediakan alat untuk semua layanan dengan prinsip layanan yang telah diberikan akses ke sumber daya di akun Anda. Ketika kepala sekolah dalam pernyataan kebijakan bucket Amazon S3 adalah masalah wakil yang membingungkan. Untuk membatasi wilayah dan akun tempat bucket dapat menerima laporan penggunaan harian, gunakan `aws:SourceArn` seperti yang ditunjukkan pada contoh di bawah ini. Jika Anda tidak ingin membatasi wilayah mana yang dapat menghasilkan laporan ini, gunakan `aws:SourceAccount` untuk membatasi berdasarkan akun mana yang menghasilkan laporan. Jika Anda tidak tahu sumber dayanya, gunakan`aws:SourceAccount`. ARN

Gunakan contoh berikut yang menyertakan perlindungan deputi yang membingungkan saat Anda membuat bucket Amazon S3 untuk menerima laporan SMS penggunaan harian dari Amazon. SNS

```
{
 "Version": "2008-10-17",
 "Statement": [{
   "Sid": "AllowPutObject",
   "Effect": "Allow",
   "Principal": {
    "Service": "sns.amazonaws.com"
   },
   "Action": "s3:PutObject",
   "Resource": "arn:aws:s3:::amzn-s3-demo-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:::amzn-s3-demo-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:::amzn-s3-demo-bucket",
   "Condition": {
    "StringEquals": {
     "aws:SourceAccount": "account_id"
    },
    "ArnLike": {
     "aws:SourceArn": "arn:aws:sns:region:account_id:*"
    }
   }
  }
 ]
}
```

> **Note**
>
> Anda dapat menerbitkan laporan penggunaan ke bucket Amazon S3 yang dimiliki oleh Akun AWS yang ditentukan dalam elemen `Condition` di kebijakan Amazon S3. Untuk mempublikasikan laporan penggunaan ke bucket Amazon S3 yang Akun AWS dimiliki orang lain, lihat Bagaimana cara menyalin objek S3 dari yang lain? Akun AWS.

Contoh laporan penggunaan harian

Setelah Anda berlangganan laporan penggunaan harian, setiap hari, Amazon SNS menempatkan CSV file dengan data penggunaan di lokasi berikut:

```
<my-s3-bucket>/SMSUsageReports/<region>/YYYY/MM/DD/00x.csv.gz
```

Setiap file dapat berisi hingga 50.000 catatan. Jika catatan selama satu hari melebihi kuota ini, Amazon SNS akan menambahkan beberapa file. Berikut adalah contoh laporan:

```
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
```

Meminta dukungan untuk perpesanan Amazon SNS SMS

> **Important**
>
> Panduan SNS SMS Pengembang Amazon telah diperbarui. Amazon SNS telah terintegrasi dengan AWS Olah Pesan Pengguna Akhir SMSpengiriman SMS pesan. Panduan ini berisi informasi terbaru tentang cara membuat, mengonfigurasi, dan mengelola SNS SMS pesan Amazon Anda.

SMSOpsi tertentu dengan Amazon SNS tidak tersedia untuk AWS akun Anda sampai Anda menghubungi AWS Support. Buat kasus di Pusat AWS Support untuk meminta hal-hal berikut:

- Peningkatan ambang SMS pengeluaran bulanan Anda

  Secara default, ambang pengeluaran bulanan adalah $1,00 (USD). Ambang batas pengeluaran Anda menentukan volume pesan yang dapat Anda kirim dengan AmazonSNS. Anda dapat meminta ambang batas pengeluaran yang memenuhi volume pesan bulanan yang diharapkan untuk kasus SMS penggunaan Anda.
- Perpindahan dari SMSkotak pasir sehingga Anda dapat mengirim SMS pesan tanpa batasan. Untuk informasi selengkapnya, lihat Pindah dari kotak SNS SMS pasir Amazon.
- Nomor asal khusus
- ID pengirim khusus. ID pengirim adalah ID kustom yang ditampilkan sebagai pengirim di perangkat penerima. Misalnya, Anda dapat menggunakan merek bisnis Anda untuk membuat sumber pesan lebih mudah dikenali. Support untuk pengirim IDs bervariasi menurut negara atau wilayah. Untuk informasi selengkapnya, lihat Negara dan wilayah yang didukung untuk SMS pengiriman pesan AWS Olah Pesan Pengguna Akhir SMS di Panduan AWS Olah Pesan Pengguna Akhir SMS Pengguna.

Topik

- Meminta kenaikan kuota SNS SMS pengeluaran Amazon bulanan Anda

Meminta kenaikan kuota SNS SMS pengeluaran Amazon bulanan Anda

Amazon SNS menyediakan kuota pengeluaran untuk membantu Anda mengelola biaya maksimum per bulan yang dikeluarkan dengan mengirim menggunakan akun Anda. SMS Kuota pengeluaran

membatasi risiko Anda jika terjadi serangan berbahaya, dan mencegah aplikasi hulu Anda mengirim lebih banyak pesan dari yang diharapkan. Anda dapat mengonfigurasi Amazon SNS untuk menghentikan penerbitan SMS pesan ketika menentukan bahwa pengiriman SMS pesan akan dikenakan biaya yang melebihi kuota pengeluaran Anda untuk bulan berjalan.

Untuk memastikan operasi Anda tidak terpengaruh, kami sarankan untuk meminta kuota pengeluaran yang cukup tinggi untuk mendukung beban kerja produksi Anda. Untuk informasi selengkapnya, lihat Langkah 1: Buka SNS SMS kasing Amazon. Setelah Anda menerima kuota, Anda dapat mengelola risiko Anda dengan menerapkan kuota penuh, atau nilai yang lebih kecil, seperti yang dijelaskan pada Langkah 2: Perbarui pengaturan Anda SMS. Dengan menerapkan nilai yang lebih kecil, Anda dapat mengontrol pengeluaran bulanan Anda dengan opsi untuk meningkatkan jika perlu.

> **Important**
>
> Karena Amazon SNS adalah sistem terdistribusi, Amazon berhenti mengirim SMS pesan dalam beberapa menit jika kuota pengeluaran terlampaui. Selama periode ini, jika Anda terus mengirim SMS pesan, Anda mungkin dikenakan biaya yang melebihi kuota Anda.

Kami menetapkan kuota pengeluaran untuk semua akun baru sebesar $1,00 (USD) per bulan. Kuota ini dimaksudkan untuk memungkinkan Anda menguji kemampuan pengiriman pesan Amazon. SNS Untuk meminta peningkatan kuota SMS pengeluaran akun Anda, buka kasus peningkatan kuota di AWS Support Center.

Topik

- Langkah 1: Buka SNS SMS kasing Amazon
- Langkah 2: Perbarui SMS pengaturan Anda di SNS konsol Amazon

Langkah 1: Buka SNS SMS kasing Amazon

Anda dapat meminta kenaikan kuota belanja bulanan Anda dengan membuka kasus peningkatan kuota di AWS Support Center.

> **Note**
>
> Beberapa bidang pada formulir permintaan ditandai sebagai "opsional." Namun, AWS Support memerlukan semua informasi yang disebutkan dalam langkah-langkah berikut untuk

memproses permintaan Anda. Jika Anda tidak memberikan semua informasi yang diperlukan, Anda mungkin mengalami penundaan dalam pemrosesan permintaan Anda.

1. Masuk ke AWS Management Console at https://console.aws.amazon.com/.
2. Di menu Dukungan, pilih Pusat Dukungan.
3. Pada panel Kasus dukungan Anda, pilih Buat kasus.
4. Pilih peningkatan batas Mencari layanan? link, lalu lengkapi yang berikut ini:
   - Untuk tipe Limit, pilih SNS Pesan Teks.
   - (Opsional) Untuk Menyediakan tautan ke situs atau aplikasi yang akan mengirim SMS pesan, memberikan informasi tentang situs web, aplikasi, atau layanan yang akan mengirim SMS pesan.
   - (Opsional) Untuk jenis pesan apa yang ingin Anda kirim, pilih jenis pesan yang akan dikirim menggunakan kode panjang Anda:
     - One Time Password (Kata Sandi Satu Kali) – Pesan yang menyediakan kata sandi yang digunakan pelanggan Anda untuk melakukan autentikasi dengan situs web atau aplikasi Anda.
     - Promotional (Promosi) – Pesan tidak penting yang mempromosikan bisnis atau layanan Anda, seperti penawaran atau pengumuman khusus.
     - Transactional (Transaksional) – Pesan informasi penting yang mendukung transaksi pelanggan, seperti konfirmasi pesanan atau pemberitahuan akun. Pesan transaksional tidak boleh berisi konten promosi atau pemasaran.
   - (Opsional) Untuk AWS Wilayah mana Anda akan mengirim pesan, pilih wilayah tempat Anda akan mengirim pesan.
   - (Opsional) Untuk negara mana Anda berencana untuk mengirim pesan, masukkan negara atau wilayah tempat Anda ingin membeli kode pendek.
   - (Opsional) Dalam Bagaimana pelanggan Anda memilih untuk menerima pesan dari Anda, berikan detail tentang proses keikutsertaan Anda.
   - (Opsional) Di kolom Harap berikan templat pesan yang Anda rencanakan untuk digunakan untuk mengirim pesan ke pelanggan Anda, sertakan templat yang akan Anda gunakan.
5. Di bawah Permintaan, lengkapi bagian berikut:
   - Untuk Wilayah, pilih Wilayah tempat Anda akan mengirim pesan.

> **Note**
>
> Wilayah diperlukan di bagian Permintaan. Bahkan jika Anda memberikan informasi ini di bagian Rincian kasus, Anda juga harus memasukkannya di sini.

- Untuk Resource Type (Jenis Sumber Daya), pilih General Limits (Batas Umum).
- Untuk Limit, pilih Kenaikan Ambang Batas Pengeluaran Akun.

6. Untuk nilai batas baru, masukkan jumlah maksimum (dalamUSD) yang dapat Anda belanjakan pada SMS setiap bulan kalender.
7. Di bawah deskripsi Kasus, untuk deskripsi kasus Penggunaan, berikan rincian berikut:

   - Situs web atau aplikasi perusahaan atau layanan yang mengirim SMS pesan.
   - Layanan yang disediakan oleh situs web atau aplikasi Anda, dan bagaimana SMS pesan Anda berkontribusi pada layanan tersebut.
   - Bagaimana pengguna mendaftar untuk secara sukarela menerima SMS pesan Anda di situs web, aplikasi, atau lokasi lain Anda.

   Jika kuota pengeluaran yang Anda minta (nilai yang Anda tentukan untuk nilai kuota Baru) melebihi $10.000 (USD), berikan detail tambahan berikut untuk setiap negara yang Anda kirimi pesan:

   - Apakah Anda menggunakan ID pengirim atau kode pendek. Jika Anda menggunakan ID pengirim, berikan:
     - ID pengirim.
     - Apakah ID pengirim terdaftar dengan operator nirkabel di negara tersebut.
   - Maksimum yang diharapkan transactions-per-second (TPS) untuk pesan Anda.
   - Ukuran pesan rata-rata.
   - Template untuk pesan yang Anda kirim ke negara tersebut.
   - (Opsional) Kebutuhan pengkodean karakter, jika ada.
8. (Opsional) Jika Anda ingin mengirimkan permintaan lebih lanjut, pilih Tambahkan permintaan lain. Jika Anda menyertakan beberapa permintaan, berikan informasi yang diperlukan untuk masing-masing permintaan. Untuk informasi yang diperlukan, lihat bagian lain di dalamnyaMeminta dukungan untuk perpesanan Amazon SNS SMS.

9. Di bawah Opsi kontak, untuk Bahasa kontak pilihan, pilih bahasa yang Anda inginkan untuk menerima komunikasi untuk kasus ini.
10. Setelah selesai, spilih Kirim.

AWS Support Tim memberikan tanggapan awal atas permintaan Anda dalam waktu 24 jam.

Untuk mencegah sistem kami digunakan untuk mengirim konten yang tidak diinginkan atau berbahaya, kami mempertimbangkan setiap permintaan dengan hati-hati. Jika bisa, kami akan mengabulkan permintaan Anda dalam waktu 24 jam ini. Namun, jika kami memerlukan informasi tambahan dari Anda, mungkin perlu waktu lebih lama untuk menyelesaikan permintaan Anda.

Jika kasus penggunaan Anda tidak sesuai dengan kebijakan kami, kami mungkin tidak dapat mengabulkan permintaan Anda.

Langkah 2: Perbarui SMS pengaturan Anda di SNS konsol Amazon

Setelah kami memberi tahu Anda bahwa kuota pengeluaran bulanan Anda telah meningkat, Anda harus menyesuaikan kuota pengeluaran untuk akun Anda di konsol Amazon. SNS

> **Important**
>
> Anda harus menyelesaikan langkah-langkah berikut atau batas SMS pengeluaran Anda tidak akan meningkat.

Cara menyesuaikan kuota pengeluaran Anda di konsol tersebut

1. Masuk ke SNSkonsol Amazon.
2. Buka menu navigasi kiri, perluas Seluler, lalu pilih Pesan teks (SMS).
3. Pada halaman Pesan teks seluler (SMS), di bagian Preferensi pesan teks, pilih Edit.
4. Pada halaman Edit preferensi pesan teks, di bagian Detail, masukkan batas SMS pengeluaran baru Anda di bidang Batas pengeluaran akun.

> **Note**
>
> Anda mungkin akan menerima peringatan bahwa nilai yang dimasukkan lebih besar dari batas pengeluaran default. Anda dapat mengabaikan peringatan ini.

5. Pilih Simpan perubahan.

> **Note**
>
> Jika Anda mendapatkan kesalahan “Parameter Tidak Valid”, periksa kontak dari AWS Support dan konfirmasikan bahwa Anda memasukkan batas SMS pengeluaran baru yang benar. Jika Anda masih mengalami masalah, buka kasing di AWS Support Center.

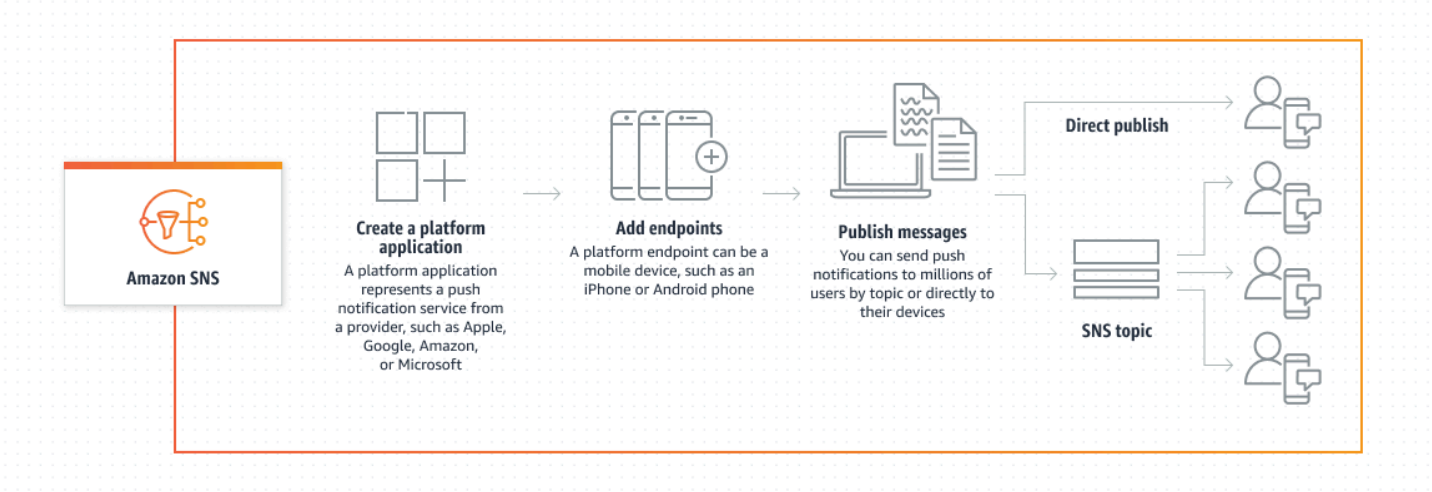
Saat Anda membuat kasus Anda di AWS Support Pusat, pastikan untuk menyertakan semua informasi yang diperlukan untuk jenis permintaan yang Anda kirimkan. Jika tidak, AWS Support harus menghubungi Anda untuk mendapatkan informasi ini sebelum melanjutkan. Dengan mengirimkan kasus yang detail, Anda membantu memastikan bahwa kasus Anda terpenuhi tanpa penundaan. Untuk detail yang diperlukan untuk jenis SMS permintaan tertentu, lihat topik berikut.

Untuk informasi selengkapnya tentang pengirimIDs, lihat dokumentasi berikut di Panduan AWS Olah Pesan Pengguna Akhir SMS Pengguna:

AWS Olah Pesan Pengguna Akhir SMS Topik	Deskripsi
Meminta peningkatan kuota pengeluaran	Kuota pengeluaran Anda menentukan berapa banyak uang yang dapat Anda keluarkan untuk mengirim SMS pesan AWS Olah Pesan Pengguna Akhir SMS setiap bulannya.
Buka kasus di pusat dukungan untuk ID pengirim	Jika Anda berencana untuk mengirim pesan ke penerima negara di mana pengirim IDs diperlukan, Anda dapat meminta ID pengirim dengan membuat kasus baru di Pusat. AWS Support

Mengirim notifikasi push seluler dengan Amazon SNS

Anda dapat menggunakan Amazon SNS untuk mengirim pesan pemberitahuan push langsung ke aplikasi di perangkat seluler. Pesan pemberitahuan push yang dikirim ke titik akhir seluler dapat muncul di aplikasi seluler sebagai peringatan pesan, pembaruan lencana, atau peringatan suara.



Topik

Cara kerja notifikasi SNS pengguna Amazon

Anda mengirim pesan notifikasi push ke kedua perangkat seluler dan desktop menggunakan salah satu layanan notifikasi push yang didukung berikut:

- Pesan Perangkat Amazon (ADM)
- Layanan Pemberitahuan Push Apple (APNs) untuk iOS dan Mac OS X
- Baidu Cloud Push (Baidu)
- Firebase Cloud Messaging () FCM

- Layanan Pemberitahuan Push Microsoft untuk Windows Phone (MPNS)
- Layanan Pemberitahuan Push Windows (WNS)

Layanan pemberitahuan push, seperti APNs danFCM, menjaga koneksi dengan setiap aplikasi dan perangkat seluler terkait yang terdaftar untuk menggunakan layanan mereka. Ketika aplikasi dan perangkat seluler terdaftar, layanan notifikasi push mengembalikan token perangkat. Amazon SNS menggunakan token perangkat untuk membuat titik akhir seluler, yang dapat mengirim pesan pemberitahuan push langsung. Agar Amazon dapat SNS berkomunikasi dengan berbagai layanan pemberitahuan push, Anda mengirimkan kredenal layanan pemberitahuan push Anda SNS ke Amazon untuk digunakan atas nama Anda. Untuk informasi selengkapnya, lihat Menyiapkan pemberitahuan push dengan Amazon SNS.

Selain mengirim pesan notifikasi push langsung, Anda juga dapat menggunakan Amazon SNS untuk mengirim pesan ke titik akhir seluler yang berlangganan suatu topik. Konsepnya sama dengan berlangganan jenis endpoint lainnya, seperti Amazon, HTTP /SSQS, email, dan, ke topikSMS, seperti yang dijelaskan dalam. Apa yang dimaksud dengan Amazon SNS? Perbedaannya adalah Amazon SNS berkomunikasi menggunakan layanan pemberitahuan push agar titik akhir seluler berlangganan menerima pesan notifikasi push yang dikirim ke topik.

Menyiapkan pemberitahuan push dengan Amazon SNS

1. Dapatkan kredensial dan token perangkat untuk platform seluler yang ingin Anda dukung.
2. Gunakan kredensil untuk membuat objek aplikasi platform (`PlatformApplicationArn`) menggunakan Amazon. SNS Untuk informasi selengkapnya, lihat Membuat aplikasi SNS platform Amazon.
3. Gunakan kredenal yang dikembalikan untuk meminta token perangkat untuk aplikasi seluler dan perangkat Anda dari layanan pemberitahuan push. Token yang Anda terima mewakili aplikasi dan perangkat seluler Anda.
4. Gunakan token perangkat dan `PlatformApplicationArn` untuk membuat objek endpoint platform (`EndpointArn`) menggunakan AmazonSNS. Untuk informasi selengkapnya, lihat Menyiapkan titik akhir SNS platform Amazon untuk notifikasi seluler.
5. Gunakan `EndpointArn` untuk memublikasikan pesan ke aplikasi di perangkat seluler. Untuk informasi selengkapnya, lihat Pesan perangkat SNS seluler Amazon langsung dan Publikasikan API di API Referensi Layanan Pemberitahuan Sederhana Amazon.

Menyiapkan aplikasi seluler di Amazon SNS

Topik ini menjelaskan cara mengatur aplikasi seluler dalam AWS Management Console menggunakan informasi yang dijelaskan dalamPrasyarat untuk pemberitahuan pengguna Amazon SNS.

Topik

Prasyarat untuk pemberitahuan pengguna Amazon SNS

Untuk mulai menggunakan notifikasi push SNS seluler Amazon, Anda memerlukan yang berikut ini:

- Satu set kredensi untuk menghubungkan ke salah satu layanan pemberitahuan push yang didukung:ADM,,, BaiduAPNs,FCM, MPNS atau. WNS
- Token perangkat atau ID registrasi untuk aplikasi dan perangkat seluler.
- Amazon SNS dikonfigurasi untuk mengirim pesan pemberitahuan push ke titik akhir seluler.
- Aplikasi seluler yang terdaftar dan dikonfigurasi untuk menggunakan salah satu layanan notifikasi push yang didukung.

Mendaftarkan aplikasi Anda dengan layanan notifikasi push memerlukan beberapa langkah. Amazon SNS memerlukan beberapa informasi yang Anda berikan ke layanan pemberitahuan push untuk mengirim pesan pemberitahuan push langsung ke titik akhir seluler. Secara umum, Anda memerlukan kredensial yang diperlukan untuk menghubungkan ke layanan notifikasi push, token perangkat atau ID pendaftaran (mewakili perangkat seluler dan aplikasi seluler Anda) yang diterima dari layanan notifikasi push, dan aplikasi seluler yang terdaftar dengan layanan notifikasi push.

Bentuk yang tepat dari kredensial berbeda antara platform seluler, tetapi dalam setiap kasus, kredensial ini harus diserahkan saat membuat koneksi ke platform. Satu set kredensial dikeluarkan

untuk setiap aplikasi seluler, dan itu harus digunakan untuk mengirim pesan ke setiap instans dari aplikasi itu.

Nama spesifik akan bervariasi tergantung pada layanan notifikasi push yang digunakan. Misalnya, saat menggunakan APNs sebagai layanan pemberitahuan push, Anda memerlukan token perangkat. Atau, saat menggunakanFCM, token perangkat setara disebut ID registrasi. Token perangkat atau ID pendaftaran adalah string yang dikirim ke aplikasi oleh sistem operasi perangkat seluler. Ini secara unik mengidentifikasi instans aplikasi seluler yang berjalan pada perangkat seluler tertentu dan dapat dianggap sebagai pengidentifikasi unik dari pasangan aplikasi-perangkat ini.

Amazon SNS menyimpan kredensialnya (ditambah beberapa pengaturan lainnya) sebagai sumber daya aplikasi platform. Token perangkat (sekali lagi dengan beberapa pengaturan tambahan) direpresentasikan sebagai objek yang disebut titik akhir platform. Setiap endpoint platform milik satu aplikasi platform tertentu, dan setiap endpoint platform dapat dikomunikasikan dengan menggunakan kredensial yang disimpan dalam aplikasi platform yang sesuai.

Bagian berikut mencakup prasyarat untuk setiap layanan notifikasi push yang didukung. Setelah Anda memperoleh informasi prasyarat, Anda dapat mengirim pesan pemberitahuan push menggunakan atau push AWS Management Console seluler AmazonSNS. APIs Untuk informasi selengkapnya, lihat Menyiapkan pemberitahuan push dengan Amazon SNS.

Membuat aplikasi SNS platform Amazon

Agar Amazon SNS dapat mengirim pesan notifikasi ke titik akhir seluler, baik secara langsung atau melalui langganan ke suatu topik, Anda harus terlebih dahulu membuat aplikasi platform. Setelah mendaftarkan aplikasi AWS, Anda perlu membuat titik akhir untuk aplikasi dan perangkat seluler. Amazon SNS menggunakan endpoint ini untuk mengirim pesan notifikasi ke aplikasi dan perangkat.

Untuk membuat aplikasi platform

1. Masuk ke SNSkonsol Amazon.
2. Di panel navigasi, pilih Pemberitahuan push.
3. Di bagian Aplikasi platform, pilih Buat aplikasi platform.

   Untuk daftar AWS Wilayah tempat Anda dapat membuat aplikasi seluler, lihatAplikasi SNS seluler Amazon didukung Wilayah.
4. Masukkan nama untuk mewakili aplikasi Anda. Nama aplikasi harus terdiri dari hanya huruf besar dan kecil, angka, garis bawah, tanda hubung, dan ASCII titik. Nama juga harus 1—256 karakter.

5. Untuk platform pemberitahuan Push, pilih platform tempat aplikasi terdaftar, lalu masukkan kredenal yang sesuai.

   > **Note**
   >
   > Jika Anda menggunakan salah satu platform Apple Push Notification Service (APNs), Anda dapat memilih antara otentikasi berbasis token atau sertifikat, lalu pilih Pilih file untuk mengunggah file.p8 atau.p12 (diekspor dari Keychain Access) ke Amazon. SNS

6. Pilih Buat aplikasi platform.

   Ini mendaftarkan aplikasi dengan AmazonSNS, yang membuat objek aplikasi platform untuk platform yang dipilih dan kemudian mengembalikan yang sesuai PlatformApplicationArn.

Menyiapkan titik akhir SNS platform Amazon untuk notifikasi seluler

Saat aplikasi dan perangkat seluler terdaftar dengan layanan notifikasi push, layanan notifikasi push mengembalikan token perangkat. Amazon SNS menggunakan token perangkat untuk membuat titik akhir seluler, yang dapat mengirim pesan pemberitahuan push langsung. Untuk informasi selengkapnya, silakan lihat Prasyarat untuk pemberitahuan pengguna Amazon SNS dan Menyiapkan pemberitahuan push dengan Amazon SNS.

Bagian ini menjelaskan pendekatan yang disarankan untuk membuat endpoint platform.

Topik

- Membuat endpoint platform
- Kode semu
- AWS SDKcontoh
- Pemecahan Masalah

Membuat endpoint platform

Untuk mendorong notifikasi ke aplikasi dengan AmazonSNS, token perangkat aplikasi tersebut harus terlebih dahulu terdaftar di Amazon SNS dengan memanggil tindakan titik akhir platform buat. Tindakan ini mengambil Amazon Resource Name (ARN) dari aplikasi platform dan token perangkat sebagai parameter dan mengembalikan titik akhir platform yang dibuat. ARN

`CreatePlatformEndpoint`Tindakan tersebut melakukan hal berikut:

- Jika endpoint platform sudah ada, jangan membuatnya lagi. Kembali ke pemanggil titik ARN akhir platform yang ada.
- Jika titik akhir platform dengan token perangkat yang sama tetapi pengaturan yang berbeda sudah ada, jangan membuatnya lagi. Lempar pengecualian ke pemanggil.
- Jika titik akhir platform tidak ada, buatlah. Kembali ke pemanggil titik ARN akhir platform yang baru dibuat.

Anda tidak boleh langsung memanggil tindakan buat endpoint platform setiap kali aplikasi dimulai, karena pendekatan ini tidak selalu menyediakan endpoint yang berfungsi. Hal ini dapat terjadi, misalnya, saat aplikasi dihapus dan diinstal ulang pada perangkat yang sama dan endpoint untuk itu sudah ada tetapi dinonaktifkan. Proses pendaftaran yang berhasil harus memenuhi hal-hal berikut:

1. Pastikan endpoint platform ada untuk kombinasi aplikasi-perangkat ini.
2. Pastikan token perangkat di endpoint platform adalah token perangkat valid terbaru.
3. Pastikan endpoint platform diaktifkan dan siap digunakan.

Kode semu

Kode semu berikut menjelaskan praktik yang disarankan untuk membuat endpoint platform yang berfungsi, saat ini, dan diaktifkan dalam berbagai kondisi awal. Pendekatan ini berfungsi baik ini pertama kali aplikasi terdaftar atau tidak, apakah endpoint platform untuk aplikasi ini sudah ada, dan apakah endpoint platform diaktifkan, memiliki token perangkat yang benar, dan seterusnya. Aman untuk memanggilnya beberapa kali berturut-turut, karena tidak akan membuat endpoint platform duplikat atau mengubah endpoint platform yang ada jika sudah diperbarui dan diaktifkan.

```
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
      (GetEndpointAttributes shows the endpoint as disabled)
    call set endpoint attributes to set the latest device token and then enable the
 platform endpoint
  endif
endif
```

Pendekatan ini dapat digunakan kapan saja aplikasi ingin mendaftar atau mendaftar ulang sendiri. Ini juga dapat digunakan saat memberi tahu Amazon SNS tentang perubahan token perangkat. Dalam hal ini, Anda cukup memanggil tindakan dengan nilai token perangkat terbaru. Beberapa hal yang perlu diperhatikan tentang pendekatan ini adalah:

- Ada dua kasus di mana ia dapat memanggil tindakan buat endpoint platform. Ini dapat disebut di awal, di mana aplikasi tidak mengetahui titik akhir platformnya sendiriARN, seperti yang terjadi selama pendaftaran pertama kali. Ini juga disebut jika panggilan `GetEndpointAttributes` tindakan awal gagal dengan pengecualian yang tidak ditemukan, seperti yang akan terjadi jika aplikasi mengetahui titik akhirnya ARN tetapi telah dihapus.
- `GetEndpointAttributes`Tindakan ini dipanggil untuk memverifikasi status titik akhir platform bahkan jika titik akhir platform baru saja dibuat. Ini terjadi ketika endpoint platform sudah ada tetapi dinonaktifkan. Dalam hal ini, tindakan buat endpoint platform berhasil tetapi tidak mengaktifkan endpoint platform, jadi Anda harus memeriksa ulang status endpoint platform sebelum mengembalikan kesuksesan.

AWS SDKcontoh

Kode berikut menunjukkan bagaimana menerapkan kode semu sebelumnya menggunakan SNS klien Amazon yang disediakan oleh. AWS SDKs

Untuk menggunakan AWS SDK, Anda harus mengkonfigurasinya dengan kredensi Anda. Untuk informasi selengkapnya, lihat File konfigurasi dan kredensial bersama di Panduan Referensi Alat AWS SDKs dan Alat.

CLI

AWS CLI

Untuk membuat endpoint aplikasi platform

`create-platform-endpoint`Contoh berikut membuat titik akhir untuk aplikasi platform tertentu menggunakan token yang ditentukan.

```
aws sns create-platform-endpoint \
    --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/GCM/
MyApplication \
    --token EXAMPLE12345...
```

Output:

```
{
      "EndpointArn": "arn:aws:sns:us-west-2:1234567890:endpoint/GCM/
MyApplication/12345678-abcd-9012-efgh-345678901234"
}
```

Java

SDKuntuk Java 2.x

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

```
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 device token or registration ID of the mobile device.
 This is a unique
               identifier provided by the device platform (e.g., Apple Push
 Notification Service (APNS) for iOS devices, Firebase Cloud Messaging (FCM)
               for Android devices) when the mobile app is registered to receive
 push notifications.

               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);
            return;
        }

        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());
        }
    }
}
```

Untuk informasi selengkapnya, lihat APITindakan push seluler.

Pemecahan Masalah

Panggilan berulang kali membuat endpoint platform dengan token perangkat yang sudah usang

Khusus untuk FCM titik akhir, Anda mungkin berpikir yang terbaik adalah menyimpan token perangkat pertama yang dikeluarkan aplikasi dan kemudian memanggil titik akhir platform buat dengan token perangkat itu setiap kali saat aplikasi dimulai. Ini mungkin tampak benar karena membebaskan aplikasi dari keharusan mengelola status token perangkat dan Amazon SNS akan secara otomatis memperbarui token perangkat ke nilai terbarunya. Namun, solusi ini memiliki sejumlah masalah serius:

- Amazon SNS mengandalkan umpan balik dari FCM untuk memperbarui token perangkat yang kedaluwarsa ke token perangkat baru. FCMmenyimpan informasi tentang token perangkat lama untuk beberapa waktu, tetapi tidak tanpa batas waktu. Setelah FCM lupa tentang koneksi antara token perangkat lama dan token perangkat baru, Amazon tidak SNS akan lagi dapat memperbarui token perangkat yang disimpan di titik akhir platform ke nilai yang benar; itu hanya akan menonaktifkan titik akhir platform sebagai gantinya.
- Aplikasi platform akan berisi beberapa endpoint platform yang sesuai dengan token perangkat yang sama.

- Amazon SNS memberlakukan kuota pada jumlah titik akhir platform yang dapat dibuat dimulai dengan token perangkat yang sama. Pada akhirnya, pembuatan endpoint baru akan gagal dengan pengecualian parameter yang tidak valid dan pesan kesalahan berikut: "Endpoint ini sudah terdaftar dengan token yang berbeda."

Untuk informasi selengkapnya tentang mengelola FCM titik akhir, lihatSNSManajemen Amazon untuk titik akhir Firebase Cloud Messaging.

Mengaktifkan kembali endpoint platform yang terkait dengan token perangkat yang tidak valid

Ketika platform seluler (seperti APNs atauFCM) memberi tahu Amazon SNS bahwa token perangkat yang digunakan dalam permintaan publikasi tidak valid, Amazon SNS menonaktifkan titik akhir platform yang terkait dengan token perangkat tersebut. Amazon kemudian SNS akan menolak penerbitan berikutnya ke token perangkat itu. Meskipun Anda mungkin berpikir bahwa yang terbaik adalah mengaktifkan kembali endpoint platform dan terus memublikasikan, dalam sebagian besar situasi, hal ini tidak akan berhasil: pesan yang diterbitkan tidak terkirim dan endpoint platform menjadi dinonaktifkan lagi segera setelahnya.

Ini karena token perangkat yang terkait dengan endpoint platform benar-benar tidak valid. Pengiriman tidak dapat berhasil karena tidak lagi sesuai dengan aplikasi yang diinstal. Lain kali diterbitkan ke, platform seluler akan kembali memberi tahu Amazon SNS bahwa token perangkat tidak valid, dan Amazon SNS akan kembali menonaktifkan titik akhir platform.

Untuk mengaktifkan kembali endpoint platform yang dinonaktifkan, endpoint tersebut harus dikaitkan dengan token perangkat yang valid (dengan panggilan tindakan atribut endpoint yang ditetapkan) lalu diaktifkan. Hanya dengan demikian pengiriman ke endpoint platform tersebut akan berhasil. Satu-satunya waktu mengaktifkan kembali endpoint platform tanpa memperbarui token perangkatnya akan berfungsi adalah ketika token perangkat yang terkait dengan endpoint itu dulu tidak valid tetapi kemudian menjadi valid lagi. Hal ini dapat terjadi, misalnya, saat aplikasi dihapus penginstalannya lalu diinstal kembali di perangkat seluler yang sama dan menerima token perangkat yang sama. Pendekatan yang disajikan di atas melakukan ini, memastikan untuk hanya mengaktifkan kembali endpoint platform setelah memverifikasi bahwa token perangkat yang terkait dengannya adalah yang terbaru yang tersedia.

Mengintegrasikan token perangkat dengan Amazon SNS untuk notifikasi seluler

Saat pertama kali mendaftarkan aplikasi dan perangkat seluler dengan layanan notifikasi, seperti Apple Push Notification Service (APNs) dan Firebase Cloud Messaging (FCM), token perangkat

atau registrasi IDs dikembalikan dari layanan notifikasi. Saat Anda menambahkan token perangkat atau pendaftaran IDs ke AmazonSNS, token tersebut digunakan `PlatformApplicationArn` API untuk membuat titik akhir untuk aplikasi dan perangkat. Saat Amazon SNS membuat titik akhir, an `EndpointArn` dikembalikan. `EndpointArn`Begitulah cara Amazon SNS mengetahui aplikasi dan perangkat seluler mana yang akan mengirim pesan notifikasi.

Anda dapat menambahkan token perangkat dan pendaftaran IDs ke Amazon SNS menggunakan metode berikut:

- Tambahkan satu token secara manual untuk AWS menggunakan AWS Management Console
- Unggah beberapa token menggunakan `CreatePlatformEndpoint` API
- Daftarkan token dari perangkat yang akan menginstal aplikasi Anda di masa mendatang

Untuk menambahkan token perangkat atau ID pendaftaran secara manual

1. Masuk ke SNSkonsol Amazon.
2. Di panel navigasi, pilih Pemberitahuan Push.
3. Di bagian Aplikasi Platform, pilih aplikasi Anda, lalu pilih Edit. Jika Anda belum membuat aplikasi platform, buat sekarang. Untuk petunjuk tentang cara melakukannya, lihat Membuat aplikasi SNS platform Amazon.
4. Pilih Tambahkan Titik Akhir.
5. Di kotak Token Endpoint, masukkan ID token atau ID pendaftaran, tergantung pada layanan notifikasi yang mana. Misalnya, dengan ADM dan FCM Anda memasukkan ID pendaftaran.
6. (Opsional) Di kotak Data Pengguna, masukkan informasi arbitrer untuk dikaitkan dengan endpoint. Amazon SNS tidak menggunakan data ini. Data harus dalam format UTF -8 dan kurang dari 2KB.
7. Pilih Tambahkan Titik Akhir.

   Dengan titik akhir yang dibuat, Anda dapat mengirim pesan langsung ke perangkat seluler atau mengirim pesan ke perangkat seluler yang berlangganan topik.

Untuk mengunggah beberapa token menggunakan **`CreatePlatformEndpoint`** API

Langkah-langkah berikut menunjukkan cara menggunakan contoh aplikasi Java (`bulkuploadpaket`) yang disediakan oleh AWS untuk mengunggah beberapa token (token perangkat atau

pendaftaranIDs) ke AmazonSNS. Anda dapat menggunakan aplikasi contoh ini untuk membantu Anda memulai mengunggah token yang ada.

> **Note**
>
> Langkah-langkah berikut menggunakan Eclipse Java. IDE Langkah-langkah mengasumsikan Anda telah menginstal AWS SDK for Java dan Anda memiliki kredensi AWS keamanan untuk Anda. Akun AWS Untuk informasi selengkapnya, lihat AWS SDK for Java. Untuk informasi selengkapnya tentang kredensial, lihat Bagaimana Cara Mendapatkan Kredensial Keamanan? di Referensi Umum AWS.

1. Unduh dan unzip file snsmobilepush.zip.
2. Buat Proyek Java baru di Eclipse.
3. Impor folder `SNSSamples` ke direktori tingkat atas Proyek Java yang baru dibuat. Di Eclipse, pilih nama Proyek Java yang tepat lalu pilih Impor, perluas Umum, pilih Sistem File, pilih Berikutnya, telusuri ke folder `SNSSamples`, pilih OK, lalu pilih Selesai.
4. Unduh salinan Open CSV library dan tambahkan ke Build Path `bulkupload` paket.
5. Buka file `BulkUpload.properties` yang terdapat dalam paket `bulkupload`.
6. Tambahkan berikut ini ke `BulkUpload.properties`:

   - `ApplicationArn` yang ingin Anda tambahkan endpoint.
   - Jalur absolut untuk lokasi CSV file Anda yang berisi token.
   - Nama-nama untuk CSV file (seperti `goodTokens.csv` dan`badTokens.csv`) yang akan dibuat untuk mencatat token yang SNS diurai Amazon dengan benar dan yang gagal.
   - (Opsional) Karakter untuk menentukan pembatas dan kutipan dalam CSV file yang berisi token.
   - (Opsional) Jumlah utas yang digunakan untuk membuat endpoint secara bersamaan. Defaultnya adalah 1 utas.

   `BulkUpload.properties` Anda yang selesai akan terlihat seperti berikut:

```
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. Jalankan BatchCreatePlatformEndpointSample aplikasi.java untuk mengunggah token ke AmazonSNS.

   Dalam contoh ini, titik akhir yang dibuat untuk token yang berhasil diunggah ke Amazon SNS akan dicatatgoodTokens.csv, sedangkan token yang salah bentuk akan dicatat. badTokens.csv Selain itu, Anda akan melihat STD OUT log yang ditulis ke konsol Eclipse, berisi konten yang mirip dengan berikut ini:

   ```
   <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
   ```

Untuk mendaftarkan token dari perangkat yang akan menginstal aplikasi Anda di masa mendatang

Anda dapat menggunakan salah satu dari dua opsi berikut:

- Gunakan layanan Amazon Cognito: Aplikasi seluler Anda akan memerlukan kredensi untuk membuat titik akhir yang terkait dengan aplikasi platform Amazon Anda. SNS Kami menyarankan Anda menggunakan kredensial sementara yang kedaluwarsa setelah jangka waktu tertentu. Untuk sebagian besar skenario, kami menyarankan Anda menggunakan Amazon Cognito untuk membuat kredensial keamanan sementara. Untuk informasi selengkapnya, lihat Panduan Developer Amazon Cognito . Jika Anda ingin diberi tahu saat aplikasi mendaftar dengan AmazonSNS, Anda dapat mendaftar untuk menerima SNS acara Amazon yang akan menyediakan titik akhir baru. ARN Anda juga dapat menggunakan ListEndpointByPlatformApplication API untuk mendapatkan daftar lengkap titik akhir yang terdaftar di AmazonSNS.
- Gunakan server proksi: Jika infrastruktur aplikasi Anda sudah disiapkan untuk aplikasi seluler Anda untuk menelepon dan mendaftar di setiap penginstalan, Anda dapat terus menggunakan penyiapan ini. Server Anda akan bertindak sebagai proxy dan meneruskan token perangkat ke notifikasi push SNS seluler Amazon, bersama dengan data pengguna apa pun yang ingin Anda simpan. Untuk tujuan ini, server proxy akan terhubung ke Amazon SNS menggunakan AWS kredensi Anda dan menggunakan CreatePlatformEndpoint API panggilan untuk mengunggah informasi token. Endpoint Amazon Resource Name (ARN) yang baru dibuat akan

dikembalikan, yang dapat disimpan server Anda untuk melakukan panggilan publikasi berikutnya ke AmazonSNS.

Metode otentikasi pemberitahuan push Amazon SNS Apple

Anda dapat mengizinkan Amazon SNS untuk mengirim pemberitahuan push ke aplikasi iOS atau macOS Anda dengan memberikan informasi yang mengidentifikasi Anda sebagai pengembang aplikasi. Untuk mengautentikasi, berikan kunci atau sertifikat saat membuat aplikasi platform, yang keduanya bisa Anda dapatkan dari akun Pengembang Apple Anda.

Kunci penandatanganan token

Kunci penandatanganan pribadi yang SNS digunakan Amazon untuk menandatangani token otentikasi Layanan Pemberitahuan Push Apple (APNs).

Jika Anda memberikan kunci penandatanganan, Amazon SNS menggunakan token untuk mengautentikasi setiap pemberitahuan push yang Anda kirim. APNs Dengan kunci penandatanganan, Anda dapat mengirim pemberitahuan push ke lingkungan APNs produksi dan kotak pasir.

Kunci penandatanganan tidak kedaluwarsa, dan Anda dapat menggunakan kunci penandatanganan yang sama untuk beberapa aplikasi. Untuk informasi selengkapnya, lihat Berkomunikasi dengan APNs menggunakan token autentikasi di bagian Bantuan Akun Pengembang di situs web Apple.

Sertifikat

TLSSertifikat yang SNS digunakan Amazon untuk mengautentikasi APNs saat Anda mengirim pemberitahuan push. Anda mendapatkan sertifikat dari akun Pengembang Apple Anda.

Sertifikat kedaluwarsa setelah satu tahun. Ketika ini terjadi, Anda harus membuat sertifikat baru dan memberikannya ke AmazonSNS. Untuk informasi selengkapnya, lihat Membuat Koneksi Berbasis Sertifikat ke APNs situs web Pengembang Apple.

Untuk mengelola APNs setelan menggunakan AWS Management Console

1. Masuk ke SNSkonsol Amazon.
2. Di Seluler, pilih Pemberitahuan push.
3. Pilih Aplikasi yang ingin Anda edit APNs pengaturannya, lalu pilih Edit.

4. Pada halaman Edit, untuk jenis Otentikasi, pilih Token atau Sertifikat.
5. Muat kredensi yang sesuai untuk sertifikat atau kunci penandatanganan token. Anda bisa mendapatkan informasi ini dari akun Pengembang Apple Anda.
6. Bergantung pada jenis otentikasi yang Anda pilih, lakukan salah satu hal berikut:
   - Jika Anda memilih Token, berikan informasi berikut dari akun Pengembang Apple Anda. Amazon SNS memerlukan informasi ini untuk membuat token otentikasi.
     - Kunci penandatanganan — Kunci penandatanganan token otentikasi dari akun Pengembang Apple Anda, yang Anda unduh sebagai file.p8. Apple memungkinkan Anda mengunduh kunci penandatanganan hanya sekali.
     - ID kunci penandatanganan — ID yang ditetapkan ke kunci penandatanganan Anda. Amazon SNS memerlukan informasi ini untuk membuat token otentikasi. Untuk menemukan nilai ini di akun Pengembang Apple Anda, pilih Sertifikat, IDs & Profil, lalu pilih kunci Anda di bagian Kunci.
     - Pengenal tim — ID yang ditetapkan ke tim akun Pengembang Apple Anda. Anda dapat menemukan nilai ini di halaman Keanggotaan.
     - Bundle identifier — ID yang ditetapkan ke aplikasi Anda. Untuk menemukan nilai ini, pilih Sertifikat, IDs & Profil, pilih Aplikasi IDs di bagian Pengenal, lalu pilih aplikasi Anda.
   - Jika Anda memilih Sertifikat, berikan informasi berikut:
     - SSLsertifikat - Berkas.p12 untuk sertifikat AndaTLS. Anda dapat mengekspor file ini dari Keychain Access setelah mengunduh dan menginstal sertifikat dari akun Pengembang Apple Anda.
     - Kata sandi sertifikat — Jika Anda menetapkan kata sandi ke sertifikat Anda, tentukan di sini.
     - Load certificate — Pilih Load certificate untuk mengunggah sertifikat Anda.
7. Setelah selesai, pilih Simpan perubahan.

SNSIntegrasi Amazon dengan penyiapan autentikasi Firebase Cloud Messaging

Topik ini menjelaskan cara mendapatkan kredensi yang diperlukan FCM API (HTTPv1) dari Google untuk digunakan dengan AWS API, AWS CLI dan. AWS Management Console

Topik

- Mengelola FCM pengaturan menggunakan CLI
- Mengelola FCM pengaturan menggunakan konsol
- Mengelola FCM pengaturan (konsol)

> **Important**
>
> 20 Juni 2023 — Google menghentikan warisan Firebase Cloud Messaging () mereka. FCM HTTP API Amazon SNS sekarang mendukung pengiriman ke semua jenis perangkat menggunakan FCM HTTP v1API. Kami menyarankan Anda memigrasikan aplikasi push seluler yang ada ke FCM HTTP v1 terbaru API pada atau sebelum 1 Juni 2024 untuk menghindari gangguan.
> 18 Januari 2024 — Amazon SNS memperkenalkan dukungan untuk FCM HTTP v1 API untuk pengiriman notifikasi push seluler ke perangkat Android.
> 26 Maret 2024 - Amazon SNS mendukung FCM HTTP v1 API untuk perangkat Apple dan tujuan Webpush. Kami menyarankan Anda memigrasikan aplikasi push seluler yang ada ke FCM HTTP v1 terbaru API pada atau sebelum 1 Juni 2024 untuk menghindari gangguan aplikasi.

Anda dapat mengizinkan Amazon SNS untuk mengirim pemberitahuan push ke aplikasi Anda dengan memberikan informasi yang mengidentifikasi Anda sebagai pengembang aplikasi. Untuk mengautentikasi, berikan APIkunci atau token saat membuat aplikasi platform. Anda bisa mendapatkan informasi berikut dari konsol aplikasi Firebase:

APIKunci

APIKuncinya adalah kredensi yang digunakan saat memanggil Firebase Legacy. API The FCM Legacy APIs akan dihapus oleh Google pada 20 Juni 2024. Jika saat ini Anda menggunakan API kunci sebagai kredensi platform, Anda dapat memperbarui kredensi platform dengan memilih Token sebagai opsi, dan mengunggah JSON file terkait untuk aplikasi Firebase Anda.

Token

Token akses berumur pendek digunakan saat memanggil HTTP v1API. Ini adalah saran Firebase API untuk mengirim pemberitahuan push. Untuk menghasilkan token akses, Firebase menyediakan satu set kredensional kepada developer dalam bentuk file kunci pribadi (juga disebut sebagai file service.json).

Prasyarat

Anda harus mendapatkan kredensi FCM service.json Anda sebelum Anda dapat mulai mengelola pengaturan di Amazon. FCM SNS Untuk mendapatkan kredensial service.json, lihat Memigrasi dari lama FCM APIs ke HTTP v1 di dokumentasi Google Firebase.

Mengelola FCM pengaturan menggunakan CLI

Anda dapat membuat notifikasi FCM push menggunakan file AWS API. Jumlah dan ukuran sumber SNS daya Amazon dalam suatu AWS akun terbatas. Untuk informasi selengkapnya, lihat titik akhir dan kuota Amazon Simple Notification Service di Panduan.Referensi Umum AWS

Untuk membuat pemberitahuan FCM push bersama dengan SNS topik Amazon (AWS API)

Saat menggunakan kredensil kunci, adalah. `PlatformCredential` `API key` Saat menggunakan kredensil token, file kunci pribadi yang `PlatformCredential` JSON diformat:

- `CreatePlatformApplication`

Untuk mengambil jenis FCM kredensi untuk SNS topik Amazon yang ada ()AWS API

Mengambil jenis kredensi`"AuthenticationMethod": "Token"`, atau: `"AuthenticationMethod": "Key"`

- GetPlatformApplicationAttributes

Untuk menyetel FCM atribut untuk SNS topik Amazon yang ada (AWS API)

Menetapkan FCM atribut:

- SetPlatformApplicationAttributes

Mengelola FCM pengaturan menggunakan konsol

Anda dapat membuat pemberitahuan FCM push menggunakan AWS Command Line Interface (CLI). Jumlah dan ukuran sumber SNS daya Amazon dalam suatu AWS akun terbatas. Untuk informasi selengkapnya, lihat titik akhir dan kuota Amazon Simple Notification Service.

Untuk membuat pemberitahuan FCM push bersama dengan SNS topik Amazon (AWS CLI)

Saat menggunakan kredensil kunci, adalah. `PlatformCredential` `API key` Saat menggunakan kredensil token, file kunci pribadi yang `PlatformCredential` JSON diformat. Saat menggunakan AWS CLI, file harus dalam format string dan karakter khusus harus diabaikan. Untuk memformat file dengan benar, Amazon SNS merekomendasikan untuk menggunakan perintah berikut`SERVICE_JSON=`jq @json <<< cat service.json``:

- create-platform-application

Untuk mengambil jenis FCM kredensi untuk SNS topik Amazon yang ada ()AWS CLI

Mengambil jenis kredensi`"AuthenticationMethod": "Token"`, atau: `"AuthenticationMethod": "Key"`

- get-platform-application-attributes

Untuk menyetel FCM atribut untuk SNS topik Amazon yang ada (AWS CLI)

Menetapkan FCM atribut:

- set-platform-application-attributes

Mengelola FCM pengaturan (konsol)

Gunakan langkah-langkah berikut untuk memasukkan kredensil yang digunakan aplikasi Anda untuk terhubung. FCM

1. Masuk ke SNSkonsol Amazon.
2. Di Seluler, pilih Pemberitahuan push.
3. Pilih FCMaplikasi yang ada dan pilih Edit. Jika Anda belum membuat aplikasi platform, lihatMembuat aplikasi SNS platform Amazon.
4. Pada halaman Edit, untuk Firebase Cloud Messaging Credentials, pilih Token atau Key. Anda bisa mendapatkan informasi berikut dari konsol aplikasi Firebase.
   - Jika Anda memilih Token, unggah file kunci pribadi yang valid. Isi file ini digunakan untuk menghasilkan token akses berumur pendek saat mengirim notifikasi.
   - Jika Anda memilih Kunci, masukkan API kunci Google.
5. Setelah selesai, pilih Simpan perubahan.

Topik terkait

- Menggunakan payload Google Firebase Cloud Messaging v1 di Amazon SNS

SNSManajemen Amazon untuk titik akhir Firebase Cloud Messaging

Topik

- Mengelola dan memelihara token perangkat
- Mendeteksi token yang tidak valid
- Menghapus token basi

Mengelola dan memelihara token perangkat

Anda dapat memastikan pengiriman pemberitahuan push aplikasi seluler Anda dengan mengikuti langkah-langkah berikut:

1. Simpan semua token perangkat, SNS titik akhir Amazon yang sesuaiARNs, dan stempel waktu di server aplikasi Anda.
2. Hapus semua token basi dan hapus titik SNS akhir ARNs Amazon yang sesuai.

Setelah start-up awal aplikasi Anda, Anda akan menerima token perangkat (juga disebut sebagai token pendaftaran) untuk perangkat. Token perangkat ini dicetak oleh sistem operasi perangkat, dan terkait dengan FCM aplikasi Anda. Setelah Anda menerima token perangkat ini, Anda dapat mendaftarkannya ke Amazon SNS sebagai titik akhir platform. Kami menyarankan Anda menyimpan token perangkat, titik akhir SNS platform AmazonARN, dan stempel waktu dengan menyimpannya ke server aplikasi Anda, atau toko persisten lainnya. Untuk menyiapkan FCM aplikasi Anda untuk mengambil dan menyimpan token perangkat, lihat Mengambil dan menyimpan token pendaftaran di dokumentasi Firebase Google.

Penting bagi Anda untuk mempertahankan up-to-date token. Token perangkat pengguna Anda dapat berubah dalam kondisi berikut:

1. Aplikasi seluler dipulihkan pada perangkat baru.
2. Pengguna menghapus instalasi atau memperbarui aplikasi.
3. Pengguna menghapus data aplikasi.

Saat token perangkat Anda berubah, kami sarankan Anda memperbarui SNS titik akhir Amazon yang sesuai dengan token baru. Ini memungkinkan Amazon SNS untuk melanjutkan komunikasi ke perangkat terdaftar. Anda dapat melakukan ini dengan menerapkan kode semu berikut dalam aplikasi seluler Anda. Ini menjelaskan praktik yang direkomendasikan untuk membuat dan memelihara titik akhir platform yang diaktifkan. Pendekatan ini dapat dijalankan setiap kali aplikasi seluler dimulai, atau sebagai pekerjaan terjadwal di latar belakang.

Kode semu

Gunakan kode FCM semu berikut untuk mengelola dan memelihara token perangkat.

```
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
```

Untuk mempelajari lebih lanjut tentang persyaratan pembaruan token, lihat Memperbarui Token secara Reguler di dokumentasi Firebase Google.

Mendeteksi token yang tidak valid

Saat pesan dikirim ke titik akhir FCM v1 dengan token perangkat yang tidak valid, SNS Amazon akan menerima salah satu pengecualian berikut:

- `UNREGISTERED`(HTTP404) — Saat Amazon SNS menerima pengecualian ini, Anda akan menerima peristiwa kegagalan pengiriman dengan `FailureType` of`InvalidPlatformToken`,

dan token Platform yang *`FailureMessage`* terkait dengan titik akhir tidak valid. Amazon SNS akan menonaktifkan titik akhir platform Anda saat pengiriman gagal dengan pengecualian ini.

- `INVALID_ARGUMENT`(HTTP400) — Saat Amazon SNS menerima pengecualian ini, itu berarti token perangkat atau muatan pesan tidak valid. Untuk informasi selengkapnya, lihat ErrorCodedi dokumentasi Firebase Google.

Karena `INVALID_ARGUMENT` dapat dikembalikan dalam salah satu kasus ini, Amazon SNS akan mengembalikan `FailureType` dari`InvalidNotification`, dan badan *`FailureMessage`* Pemberitahuan tidak valid. Ketika Anda menerima kesalahan ini, verifikasi bahwa payload Anda sudah benar. Jika benar, verifikasi bahwa token perangkat tersebut up-to-date. Amazon tidak SNS akan menonaktifkan titik akhir platform Anda saat pengiriman gagal dengan pengecualian ini.

Kasus lain di mana Anda akan mengalami peristiwa kegagalan `InvalidPlatformToken` pengiriman adalah ketika token perangkat terdaftar bukan milik aplikasi yang mencoba mengirim pesan itu. Dalam hal ini, Google akan mengembalikan kesalahan SENDER_ID_ MISMATCH. Amazon SNS akan menonaktifkan titik akhir platform Anda saat pengiriman gagal dengan pengecualian ini.

Semua kode kesalahan yang diamati yang diterima dari FCM v1 API tersedia untuk Anda CloudWatch saat Anda mengatur pencatatan status pengiriman untuk aplikasi Anda.

Untuk menerima acara pengiriman untuk aplikasi Anda, lihatPeristiwa aplikasi yang tersedia.

Menghapus token basi

Token dianggap basi setelah pengiriman pesan ke perangkat titik akhir mulai gagal. Amazon SNS menetapkan token basi ini sebagai titik akhir yang dinonaktifkan untuk aplikasi platform Anda. Saat Anda memublikasikan ke titik akhir yang dinonaktifkan, Amazon SNS akan menampilkan `EventDeliveryFailure` peristiwa dengan `FailureType` dari`EndpointDisabled`, dan Endpoint dinonaktifkan. `FailureMessage` Untuk menerima acara pengiriman untuk aplikasi Anda, lihatPeristiwa aplikasi yang tersedia.

Saat Anda menerima kesalahan ini dari AmazonSNS, Anda perlu menghapus atau memperbarui token basi di aplikasi platform Anda.

Menggunakan Amazon SNS untuk pemberitahuan push seluler

Bagian ini menjelaskan cara mengirim notifikasi push seluler.

Topik

- Menerbitkan ke topik
- Pesan perangkat SNS seluler Amazon langsung
- Menerbitkan SNS notifikasi Amazon dengan muatan khusus platform

Menerbitkan ke topik

Anda juga dapat menggunakan Amazon SNS untuk mengirim pesan ke titik akhir seluler yang berlangganan suatu topik. Konsepnya sama dengan berlangganan jenis endpoint lainnya, seperti Amazon, HTTP /SSQS, email, dan, ke topikSMS, seperti yang dijelaskan dalam. Apa yang dimaksud dengan Amazon SNS? Perbedaannya adalah Amazon SNS berkomunikasi melalui layanan notifikasi seperti Apple Push Notification Service (APNS) dan Google Firebase Cloud Messaging (FCM). Melalui layanan notifikasi, endpoint seluler yang berlangganan menerima notifikasi yang dikirim ke topik.

Pesan perangkat SNS seluler Amazon langsung

Anda dapat mengirim pesan pemberitahuan SNS push Amazon langsung ke titik akhir yang mewakili aplikasi di perangkat seluler.

Untuk mengirim pesan langsung

1. Masuk ke SNSkonsol Amazon.
2. Pada panel navigasi, pilih Pemberitahuan push.
3. Pada halaman Pemberitahuan push seluler, di bagian Aplikasi Platform, pilih nama aplikasi, misalnya ***MyApp***.
4. Pada ***MyApp*** halaman, di bagian Endpoints, pilih endpoint dan kemudian pilih Publish message.
5. Pada halaman Publikasikan pesan ke endpoint, masukkan pesan yang akan muncul di aplikasi pada perangkat seluler, lalu pilih Publikasikan pesan.

   Amazon SNS mengirimkan pesan notifikasi ke layanan notifikasi platform yang, pada gilirannya, mengirim pesan ke aplikasi.

Menerbitkan SNS notifikasi Amazon dengan muatan khusus platform

Anda dapat menggunakan AWS Management Console atau Amazon SNS APIs untuk mengirim pesan khusus dengan muatan khusus platform ke perangkat seluler. Untuk informasi tentang

menggunakan Amazon SNSAPIs, lihat APITindakan push seluler dan `SNSMobilePush.java` file di`snsmobilepush.zip`.

Topik

- Mengirim pesan JSON yang diformat
- Mengirim pesan khusus platform
- Mengirim pesan ke aplikasi di berbagai platform
- Mengirim pesan ke APNs sebagai peringatan atau pemberitahuan latar belakang
- Menggunakan payload Google Firebase Cloud Messaging v1 di Amazon SNS

Mengirim pesan JSON yang diformat

Saat Anda mengirim muatan khusus platform, data harus diformat sebagai string pasangan JSON nilai kunci, dengan tanda kutip lolos.

Contoh berikut menunjukkan pesan khusus untuk FCM platform.

```
{
"GCM": "{\"fcmV1Message\": {\"message\": {\"notification\": {\"title\": \"Hello\",
 \"body\": \"This is a test.\"}, \"data\": {\"dataKey\": \"example\"}}}}"
}
```

Mengirim pesan khusus platform

Selain mengirim data khusus sebagai pasangan nilai kunci, Anda dapat mengirim pasangan nilai kunci khusus platform.

Contoh berikut menunjukkan penyertaan FCM parameter `time_to_live` dan `collapse_key` setelah pasangan nilai kunci data kustom dalam parameter. FCM `data`

```
{
"GCM": "{\"fcmV1Message\": {\"message\": {\"notification\": {\"title\": \"TitleTest\",
 \"body\": \"Sample message for Android or iOS endpoints.\"}, \"data\":{\"time_to_live
\": 3600,\"collapse_key\":\"deals\"}}}}"
}
```

Untuk daftar pasangan nilai kunci yang didukung oleh setiap layanan pemberitahuan push yang didukung di AmazonSNS, lihat berikut ini:

> **Important**
>
> Amazon SNS sekarang mendukung Firebase Cloud Messaging (FCM) HTTP v1 API untuk mengirim notifikasi push seluler ke perangkat Android.
> 26 Maret 2024 - Amazon SNS mendukung FCM HTTP v1 API untuk perangkat Apple dan tujuan Webpush. Kami menyarankan Anda memigrasikan aplikasi push seluler yang ada ke FCM HTTP v1 terbaru API pada atau sebelum 1 Juni 2024 untuk menghindari gangguan aplikasi.

- Referensi Kunci Payload dalam dokumentasi APNs
- Firebase Cloud Messaging HTTP Protocol dalam dokumentasi FCM
- Kirim Pesan dalam ADM dokumentasi

Mengirim pesan ke aplikasi di berbagai platform

Untuk mengirim pesan ke aplikasi yang diinstal pada perangkat untuk beberapa platform, seperti FCM danAPNs, Anda harus terlebih dahulu berlangganan titik akhir seluler ke topik di Amazon SNS dan kemudian mempublikasikan pesan ke topik tersebut.

Contoh berikut menunjukkan pesan untuk dikirim ke endpoint seluler berlangganAPNs,FCM, dan: ADM

```
{
  "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\"}}"
}
```

Mengirim pesan ke APNs sebagai peringatan atau pemberitahuan latar belakang

Amazon SNS dapat mengirim pesan ke APNs as `alert` atau `background` notifikasi (untuk informasi selengkapnya, lihat Mendorong Pembaruan Latar Belakang ke Aplikasi Anda dalam APNs dokumentasi).

- `alert`APNsPemberitahuan menginformasikan pengguna dengan menampilkan pesan peringatan, memutar suara, atau menambahkan lencana ke ikon aplikasi Anda.
- `background`APNsPemberitahuan bangun atau menginstruksikan aplikasi Anda untuk bertindak atas konten pemberitahuan, tanpa memberi tahu pengguna.

Menentukan nilai APNs header kustom

Sebaiknya tentukan nilai kustom untuk atribut pesan yang `AWS.SNS.MOBILE.APNS.PUSH_TYPE` dicadangkan menggunakan SNS `Publish` API tindakan Amazon AWS SDKs, atau. AWS CLI CLIContoh berikut menetapkan `content-available` ke 1 dan `apns-push-type` ke `background` untuk topik yang ditentukan.

```
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
```

> **Note**
>
> Pastikan JSON strukturnya valid. Tambahkan koma setelah setiap pasangan kunci-nilai, kecuali yang terakhir.

Menyimpulkan header tipe APNs push dari payload

Jika Anda tidak menyetel `apns-push-type` APNs header, Amazon akan SNS menyetel header ke `alert` atau `background` bergantung pada `content-available` kunci dalam `aps` kamus konfigurasi APNs payload JSON yang diformat.

> **Note**
>
> Amazon SNS hanya dapat menyimpulkan `alert` atau `background` header, meskipun `apns-push-type` header dapat diatur ke nilai lain.

- `apns-push-type` diatur ke `alert`
  - Jika kamus `aps` berisi `content-available` yang diatur ke `1` dan satu atau beberapa kunci yang memicu interaksi pengguna.
  - Jika kamus `aps` berisi `content-available` yang diatur ke `0` atau jika kunci `content-available` tidak ada.
  - Jika nilai kunci `content-available` bukan bilangan bulat atau Boolean.
- `apns-push-type` diatur ke `background`
  - Jika kamus `aps` hanya berisi `content-available` yang diatur ke `1` dan tidak ada kunci lain yang memicu interaksi pengguna.

> **Important**
>
> Jika Amazon SNS mengirimkan objek konfigurasi mentah APNs sebagai notifikasi khusus latar belakang, Anda harus menyertakan `content-available` set ke `1` dalam kamus. `aps` Meskipun Anda dapat menyertakan kunci kustom, kamus `aps` tidak boleh berisi kunci apa pun yang memicu interaksi pengguna (misalnya, peringatan, lencana, atau suara).

Berikut ini adalah contoh objek konfigurasi mentah.

```
{
  "APNS": "{\"aps\":{\"content-available\":1},\"Foo1\":\"Bar\",\"Foo2\":123}"
}
```

Dalam contoh ini, Amazon SNS menyetel `apns-push-type` APNs header untuk pesan tersebut`background`. Ketika Amazon SNS mendeteksi bahwa `apn` kamus berisi `content-available` kunci yang disetel ke `1` —dan tidak berisi kunci lain yang dapat memicu interaksi pengguna—ia akan menyetel header ke. `background`

Menggunakan payload Google Firebase Cloud Messaging v1 di Amazon SNS

Amazon SNS mendukung penggunaan FCM HTTP v1 API untuk mengirim notifikasi ke tujuan Android, iOS, dan Webpush. Topik ini memberikan contoh struktur payload saat menerbitkan notifikasi push seluler menggunakanCLI, atau Amazon SNSAPI.

Anda dapat menyertakan jenis pesan berikut di payload saat mengirim FCM notifikasi:

- Pesan data — Pesan data ditangani oleh aplikasi klien Anda dan berisi pasangan nilai kunci khusus. Saat membuat pesan data, Anda harus menyertakan `data` kunci dengan JSON objek sebagai nilainya, lalu masukkan pasangan nilai kunci kustom Anda.
- Pesan pemberitahuan atau pesan tampilan - Pesan notifikasi berisi sekumpulan kunci yang telah ditentukan sebelumnya yang ditangani oleh. FCM SDK Tombol-tombol ini bervariasi tergantung pada jenis perangkat yang Anda kirimkan. Untuk informasi selengkapnya tentang kunci notifikasi khusus platform, lihat berikut ini:
  - Tombol notifikasi Android
  - APNStombol notifikasi
  - Tombol notifikasi Webpush

Untuk informasi selengkapnya tentang jenis FCM pesan, lihat Jenis pesan di dokumentasi Firebase Google.

Daftar Isi

Menggunakan struktur payload FCM v1 untuk mengirim pesan

Jika Anda membuat FCM aplikasi untuk pertama kalinya, atau ingin memanfaatkan fitur FCM v1, Anda dapat memilih untuk mengirim muatan berformat FCM v1. Untuk melakukan ini, Anda harus menyertakan kunci `fcmV1Message` tingkat atas. Untuk informasi selengkapnya tentang membuat

payload FCM v1, lihat Memigrasi dari lama FCM APIs ke HTTP v1 dan Menyesuaikan pesan di seluruh platform dalam dokumentasi Firebase Google.

FCMv1 contoh payload dikirim ke Amazon: SNS

> **Note**
>
> Nilai GCM kunci yang digunakan dalam contoh berikut harus dikodekan sebagai String saat menerbitkan notifikasi menggunakan Amazon. SNS

```
{
  "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\"
          }
        }
      }
    }
  }"
}
```

Saat mengirim JSON payload, pastikan untuk menyertakan `message-structure` atribut dalam permintaan Anda, dan atur ke`json`.

CLIcontoh:

```
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
```

Untuk informasi selengkapnya tentang mengirim payload berformat FCM v1, lihat hal berikut di dokumentasi Firebase Google:

- Migrasi dari warisan FCM APIs ke v1 HTTP
- Tentang FCM pesan
- RESTSumber daya: projects.messages

Menggunakan struktur payload lama untuk mengirim pesan ke v1 FCM API

Saat bermigrasi ke FCM v1, Anda tidak perlu mengubah struktur payload yang Anda gunakan untuk kredensi lama Anda. Amazon SNS mengubah payload Anda menjadi struktur payload FCM v1 baru, dan mengirimkannya ke Google.

Format payload pesan masukan:

```
{
  "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\"}}"
}
```

Pesan dikirim ke Google:

```
{
  "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"
    }
  }
}
```

Potensi risiko

- Pemetaan lama ke v1 tidak mendukung Layanan Pemberitahuan Push Apple (APNS) `headers` atau tombol. `fcm_options` Jika Anda ingin menggunakan bidang ini, kirim payload FCM v1.
- Dalam beberapa kasus, header pesan diperlukan oleh FCM v1 untuk mengirim notifikasi senyap ke perangkat AndaAPNs. Jika saat ini Anda mengirim notifikasi senyap ke APNs perangkat Anda, mereka tidak akan berfungsi dengan pendekatan lama. Sebagai gantinya, kami sarankan menggunakan payload FCM v1 untuk menghindari masalah yang tidak terduga. Untuk menemukan daftar APNs header dan tujuan penggunaannya, lihat Berkomunikasi dengan APNs di Panduan Pengembang Apple.
- Jika Anda menggunakan SNS atribut TTL Amazon saat mengirim notifikasi, itu hanya akan diperbarui di `android` bidang. Jika Anda ingin menyetel TTL APNS atribut, gunakan payload FCM v1.
- `webpush`Kunci`android`,`apns`, dan akan dipetakan dan diisi dengan semua kunci yang relevan yang disediakan. Misalnya, jika Anda menyediakan`title`, yang merupakan kunci yang dibagikan di antara ketiga platform, pemetaan FCM v1 akan mengisi ketiga platform dengan judul yang Anda berikan.
- Beberapa kunci bersama di antara platform mengharapkan jenis nilai yang berbeda. Misalnya, `badge` kunci yang diteruskan untuk `apns` mengharapkan nilai integer, sedangkan `badge` kunci dilewatkan untuk `webpush` mengharapkan nilai String. Dalam kasus di mana Anda memberikan `badge` kunci, pemetaan FCM v1 hanya akan mengisi kunci yang Anda berikan nilai valid.

FCMperistiwa kegagalan pengiriman

Tabel berikut menyediakan jenis SNS kegagalan Amazon yang sesuai dengan kode kesalahan/status yang diterima dari Google untuk permintaan notifikasi FCM v1. Semua kode kesalahan yang diamati

yang diterima dari FCM v1 API tersedia untuk Anda CloudWatch saat Anda mengatur pencatatan status pengiriman untuk aplikasi Anda.

FCMkode kesalahan/ status	Jenis SNS kegagalan Amazon	Pesan kegagalan	Penyebab dan mitigasi
`UNREGISTERED`	`InvalidPlatformToken`	Token platform yang terkait dengan titik akhir tidak valid.	Token perangkat yang dilampirkan ke titik akhir Anda sudah basi atau tidak valid. Amazon SNS menonaktifkan titik akhir Anda. Perbarui SNS titik akhir Amazon ke token perangkat terbaru.
`INVALID_ARGUMENT`	`InvalidNotification`	Badan notifikasi tidak valid.	Token perangkat atau payload pesan mungkin tidak valid. Verifikasi bahwa payload pesan Anda valid. Jika payload pesan valid, perbarui SNS titik akhir Amazon ke token perangkat terbaru.
`SENDER_ID_MISMATCH`	`InvalidPlatformToken`	Token platform yang terkait dengan titik akhir tidak valid.	Aplikasi platform yang terkait dengan token perangkat tidak memiliki izin untuk mengirim ke token perangkat . Verifikasi bahwa Anda menggunak

FCMkode kesalahan/ status	Jenis SNS kegagalan Amazon	Pesan kegagalan	Penyebab dan mitigasi
			an FCM kredensil yang benar di aplikasi SNS platform Amazon Anda.
UNAVAILABLE	DependencyUnavailable	Ketergantungan tidak tersedia.	FCMtidak dapat memproses permintaan tepat waktu. Semua percobaan ulang yang dijalankan oleh Amazon SNS telah gagal. Anda dapat menyimpan pesan-pesan ini dalam antrean huruf mati (DLQ) dan mengaktif kannya kembali nanti.
INTERNAL	UnexpectedFailure	Kegagalan tak terduga; silakan hubungi Amazon. Frasa kegagalan [Kesalahan Internal].	FCMServer mengalami kesalahan saat mencoba memproses permintaan Anda. Semua percobaan ulang yang dijalankan oleh Amazon SNS telah gagal. Anda dapat menyimpan pesan-pesan ini dalam antrean huruf mati (DLQ) dan mengaktif kannya kembali nanti.

FCMkode kesalahan/ status	Jenis SNS kegagalan Amazon	Pesan kegagalan	Penyebab dan mitigasi
`THIRD_PAR TY_AUTH_ERROR`	`InvalidCr edentials`	Kredensi aplikasi platform tidak valid.	Pesan yang ditargetk an ke perangkat iOS atau perangkat Webpush tidak dapat dikirim. Verifikas i bahwa kredenal pengembangan dan produksi Anda valid.
`QUOTA_EXCEEDED`	`Throttled`	Permintaan dibatasi oleh [gcm].	Kuota rasio pesan, kuota tarif pesan perangkat, atau kuota tarif pesan topik telah terlampaui. Untuk informasi tentang cara mengatasi masalah ini, lihat ErrorCodedi dokumentasi Firebase Google.

FCMkode kesalahan/ status	Jenis SNS kegagalan Amazon	Pesan kegagalan	Penyebab dan mitigasi
`PERMISSIO N_DENIED`	`InvalidNo tification`	Badan notifikasi tidak valid.	Dalam kasus `PERMISSIO N_DENIED` pengecualian, pemanggil (FCMaplik asi Anda) tidak memiliki izin untuk menjalankan operasi yang ditentukan dalam payload. Arahkan ke FCM konsol Anda, dan verifikasi bahwa kredensil Anda mengaktifkan API tindakan yang diperlukan.

Atribut aplikasi SNS seluler Amazon

Amazon Simple Notification Service (AmazonSNS) menyediakan dukungan untuk mencatat status pengiriman pesan pemberitahuan push. Setelah Anda mengonfigurasi atribut aplikasi, entri log akan dikirim ke CloudWatch Log untuk pesan yang dikirim dari Amazon SNS ke titik akhir seluler. Mencatat status pengiriman pesan membantu memberikan wawasan operasional yang lebih baik, seperti berikut ini:

- Ketahui apakah pesan pemberitahuan push dikirim dari Amazon SNS ke layanan pemberitahuan push.
- Identifikasi respons yang dikirim dari layanan pemberitahuan push ke AmazonSNS.
- Tentukan waktu tunggu pesan (waktu antara stempel waktu publikasi dan sesaat sebelum diserahkan ke layanan notifikasi push).

Untuk mengonfigurasi atribut aplikasi untuk status pengiriman pesan, Anda dapat menggunakan AWS Management Console, kit pengembangan AWS perangkat lunak (SDKs), atau kueriAPI.

Topik

Mengkonfigurasi atribut status pengiriman pesan menggunakan AWS Management Console

1. Masuk ke SNSkonsol Amazon.
2. Pada panel navigasi, arahkan ke Seluler, lalu pilih Notifikasi push.
3. Dari bagian Aplikasi Platform, pilih aplikasi yang berisi titik akhir yang Anda inginkan menerima CloudWatch Log.
4. Pilih Tindakan Aplikasi lalu pilih Status Pengiriman.
5. Pada kotak dialog Status Pengiriman, pilih Buat IAM Peran.

   Anda kemudian akan diarahkan ke IAM konsol.
6. Pilih Izinkan untuk memberi Amazon akses SNS tulis untuk menggunakan CloudWatch Log atas nama Anda.
7. Sekarang, kembali ke kotak dialog Status Pengiriman, masukkan nomor di bidang Persentase Sukses ke Sampel (0-100) untuk persentase pesan yang berhasil dikirim yang ingin Anda terima CloudWatch Log.

   > **Note**
   >
   > Setelah Anda mengonfigurasi atribut aplikasi untuk status pengiriman pesan, semua pengiriman pesan yang gagal menghasilkan CloudWatch Log.
8. Terakhir, pilih Simpan Konfigurasi. Anda sekarang akan dapat melihat dan mengurai CloudWatch Log yang berisi status pengiriman pesan. Untuk informasi selengkapnya tentang penggunaan CloudWatch, lihat CloudWatchDokumentasi.

Contoh CloudWatch log status pengiriman SNS pesan Amazon

Setelah Anda mengonfigurasi atribut status pengiriman pesan untuk titik akhir aplikasi, CloudWatch Log akan dihasilkan. Contoh log, dalam JSON format, ditampilkan sebagai berikut:

SUCCESS


```
{
  "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"
  }
}
```


FAILURE


```
{
  "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"
  }
}
```

Untuk daftar kode respons layanan notifikasi push, lihat Kode respons platform.

Mengkonfigurasi atribut status pengiriman pesan dengan AWS SDKs

AWS SDKsMenyediakan APIs dalam beberapa bahasa untuk menggunakan atribut status pengiriman pesan dengan AmazonSNS.

Contoh Java berikut menunjukkan cara menggunakan `SetPlatformApplicationAttributes` API untuk mengkonfigurasi atribut aplikasi untuk status pengiriman pesan pesan pemberitahuan push. Anda dapat menggunakan atribut berikut untuk status pengiriman pesan: `SuccessFeedbackRoleArn`, `FailureFeedbackRoleArn`, dan `SuccessFeedbackSampleRate`. `FailureFeedbackRoleArn`Atribut `SuccessFeedbackRoleArn` dan digunakan untuk memberi Amazon akses SNS tulis untuk menggunakan CloudWatch Log atas nama Anda. Atribut `SuccessFeedbackSampleRate` adalah untuk menentukan persentase tingkat sampel (0-100) dari pesan yang berhasil terkirim. Setelah Anda mengonfigurasi `FailureFeedbackRoleArn` atribut, maka semua pengiriman pesan yang gagal menghasilkan CloudWatch Log.

```
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);
```

Untuk informasi selengkapnya tentang Java, lihat Memulai dengan AWS SDK for Java. SDK

Kode respons platform

Berikut ini adalah daftar tautan untuk kode respons layanan notifikasi push:

Layanan notifikasi push	Kode respons
Pesan Perangkat Amazon (ADM)	Lihat Format Respons dalam ADM dokumenta si.
Layanan Pemberitahuan Push Apple (APNs)	Lihat HTTP/2 Respons dari APNs dalam Berkomunikasi dengan APNs di Panduan Pemrograman Pemberitahuan Lokal dan Jarak Jauh.
Firebase Cloud Messaging () FCM	Lihat Kode Respons Kesalahan Pesan Hilir di dokumentasi Firebase Cloud Messaging.
Layanan Pemberitahuan Push Microsoft untuk Windows Phone (MPNS)	Lihat Kode Respons Layanan Notifikasi Push untuk Ponsel Windows 8 dalam dokumentasi Pengembangan Windows 8.
Layanan Pemberitahuan Push Windows (WNS)	Lihat "Kode respons" di Permintaan Layanan Notifikasi Push dan Header Respons (Aplikasi Windows Runtime) di dokumentasi Pengemban gan Windows 8.

Pemberitahuan acara SNS aplikasi Amazon untuk aplikasi seluler

Amazon SNS menyediakan dukungan untuk memicu pemberitahuan ketika peristiwa aplikasi tertentu terjadi. Anda kemudian dapat mengambil beberapa tindakan terprogram pada peristiwa itu. Aplikasi Anda harus menyertakan dukungan untuk layanan pemberitahuan push seperti Apple Push Notification Service (APNs), Firebase Cloud Messaging (FCM), dan Windows Push Notification Services (WNS). Anda mengatur notifikasi acara aplikasi menggunakan SNS konsol Amazon, AWS CLI, atau file AWS SDKs.

Topik

- Peristiwa aplikasi yang tersedia
- Mengirim notifikasi push seluler

Peristiwa aplikasi yang tersedia

Notifikasi peristiwa aplikasi melacak kapan endpoint platform individual dibuat, dihapus, dan diperbarui, serta kegagalan pengiriman. Berikut ini adalah nama atribut untuk peristiwa aplikasi.

Nama atribut	Pemicu notifikasi
`EventEndpointCreated`	Endpoint platform baru ditambahkan ke aplikasi Anda.
`EventEndpointDeleted`	Endpoint platform apa pun yang terkait dengan aplikasi Anda akan dihapus.
`EventEndpointUpdated`	Atribut endpoint platform apa pun yang terkait dengan aplikasi Anda diubah.
`EventDeliveryFailure`	Pengiriman ke salah satu endpoint platform yang terkait dengan aplikasi Anda mengalami kegagalan permanen. **Note** Untuk melacak kegagalan pengiriman di sisi aplikasi platform, berlangganan peristiwa status pengiriman pesan untuk aplikasi. Untuk informasi selengkapnya, lihat Menggunakan Atribut SNS Aplikasi Amazon untuk Status Pengiriman Pesan.

Anda dapat mengaitkan atribut apa pun dengan aplikasi yang kemudian dapat menerima notifikasi peristiwa ini.

Mengirim notifikasi push seluler

Untuk mengirim notifikasi peristiwa aplikasi, Anda menentukan topik untuk menerima notifikasi untuk setiap jenis peristiwa. Saat Amazon SNS mengirimkan notifikasi, topik dapat meruteknya ke titik akhir yang akan mengambil tindakan terprogram.

Important

Aplikasi volume tinggi akan membuat sejumlah besar notifikasi peristiwa aplikasi (misalnya, puluhan ribu), yang akan membanjiri endpoint yang dimaksudkan untuk digunakan manusia, seperti alamat email, nomor telepon, dan aplikasi seluler. Pertimbangkan panduan berikut saat Anda mengirim notifikasi peristiwa aplikasi ke suatu topik:

- Setiap topik yang menerima notifikasi hanya boleh berisi langganan untuk titik akhir terprogram, seperti HTTP atau HTTPS titik akhir, antrian SQS Amazon, atau fungsi. AWS Lambda
- Untuk mengurangi jumlah pemrosesan yang dipicu oleh notifikasi, batasi langganan setiap topik ke sejumlah kecil (misalnya, lima atau lebih sedikit).

Anda dapat mengirim pemberitahuan acara aplikasi menggunakan SNS konsol Amazon, AWS Command Line Interface (AWS CLI), atau AWS SDKs.

AWS Management Console

1. Masuk ke SNSkonsol Amazon.
2. Pada panel navigasi, pilih Seluler, Notifikasi push.
3. Pada halaman pemberitahuan push seluler, di bagian Aplikasi Platform, pilih aplikasi lalu pilih Edit.
4. Perluas bagian Notifikasi peristiwa.
5. Pilih Tindakan, Konfigurasikan peristiwa.
6. Masukkan topik ARNs for yang akan digunakan untuk acara berikut:
   - Endpoint Dibuat
   - Endpoint Dihapus
   - Endpoint Diperbarui
   - Kegagalan Pengiriman
7. Pilih Simpan perubahan.

AWS CLI

Jalankan perintah set-platform-application-attributes.

Contoh berikut menetapkan SNS topik Amazon yang sama untuk keempat acara aplikasi:

```
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 SDKs

Atur pemberitahuan acara aplikasi dengan mengirimkan `SetPlatformApplicationAttributes` permintaan ke Amazon SNS API menggunakan file. AWS SDK

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, termasuk bantuan memulai dan informasi tentang versi sebelumnya, lihatMenggunakan Amazon SNS dengan AWS SDK.

APITindakan push seluler

Untuk menggunakan push SNS seluler AmazonAPIs, Anda harus terlebih dahulu memenuhi prasyarat untuk layanan pemberitahuan push, seperti Apple Push Notification Service (APNs) dan Firebase Cloud Messaging (). FCM Untuk informasi selengkapnya tentang prasyarat, lihat Prasyarat untuk pemberitahuan pengguna Amazon SNS.

Untuk mengirim pesan pemberitahuan push ke aplikasi seluler dan perangkat menggunakanAPIs, Anda harus terlebih dahulu menggunakan `CreatePlatformApplication` tindakan, yang mengembalikan `PlatformApplicationArn` atribut. Atribut `PlatformApplicationArn` kemudian digunakan oleh `CreatePlatformEndpoint`, yang mengembalikan atribut `EndpointArn`. Anda kemudian dapat menggunakan atribut `EndpointArn` dengan tindakan `Publish` untuk mengirim pesan notifikasi ke aplikasi dan perangkat seluler, atau Anda dapat menggunakan atribut `EndpointArn` dengan tindakan `Subscribe` untuk berlangganan suatu topik. Untuk informasi selengkapnya, lihat Menyiapkan pemberitahuan push dengan Amazon SNS.

Dorongan SNS seluler Amazon APIs adalah sebagai berikut:

`CreatePlatformApplication`

Membuat objek aplikasi platform untuk salah satu layanan pemberitahuan push yang didukung, seperti APNs danFCM, yang dapat didaftarkan oleh perangkat dan aplikasi seluler. Mengembalikan atribut `PlatformApplicationArn`, yang digunakan oleh tindakan `CreatePlatformEndpoint`.

`CreatePlatformEndpoint`

Membuat endpoint untuk perangkat dan aplikasi seluler di salah satu layanan notifikasi push yang didukung. `CreatePlatformEndpoint` menggunakan atribut `PlatformApplicationArn` yang dikembalikan dari tindakan `CreatePlatformApplication`. Atribut `EndpointArn`, yang dikembalikan saat menggunakan `CreatePlatformEndpoint`, kemudian digunakan dengan tindakan `Publish` untuk mengirim pesan notifikasi ke aplikasi dan perangkat seluler.

`CreateTopic`

Membuat topik yang pesannya dapat dipublikasikan.

`DeleteEndpoint`

Menghapus endpoint untuk perangkat dan aplikasi seluler di salah satu layanan notifikasi push yang didukung.

`DeletePlatformApplication`

Menghapus objek aplikasi platform.

`DeleteTopic`

Menghapus topik dan semua langganannya.

`GetEndpointAttributes`

Mengambil atribut endpoint untuk perangkat dan aplikasi seluler.

`GetPlatformApplicationAttributes`

Mengambil atribut objek aplikasi platform.

`ListEndpointsByPlatformApplication`

Mencantumkan endpoint dan atribut endpoint untuk perangkat dan aplikasi seluler dalam layanan notifikasi push yang didukung.

`ListPlatformApplications`

Mencantumkan objek aplikasi platform untuk layanan notifikasi push yang didukung.

`Publish`

Mengirim pesan notifikasi ke semua endpoint langganan topik.

`SetEndpointAttributes`

Menetapkan atribut untuk endpoint untuk perangkat dan aplikasi seluler.

`SetPlatformApplicationAttributes`

Menetapkan atribut objek aplikasi platform.

`Subscribe`

Bersiap untuk berlangganan endpoint dengan mengirimkan endpoint pesan konfirmasi. Untuk benar-benar membuat langganan, pemilik endpoint harus memanggil ConfirmSubscription tindakan dengan token dari pesan konfirmasi.

`Unsubscribe`

Menghapus langganan.

APIKesalahan push SNS seluler Amazon yang umum

Kesalahan yang dikembalikan oleh Amazon SNS APIs untuk push seluler tercantum dalam tabel berikut. Untuk informasi selengkapnya tentang Amazon SNS APIs untuk push seluler, lihatAPITindakan push seluler.

Kesalahan	Deskripsi	HTTPSkode status	APIAksi
Nama Aplikasi adalah string null	Nama aplikasi yang diperlukan diatur ke null.	400	`CreatePlatformApplication`
Nama Platform adalah string null	Nama platform yang diperlukan diatur ke null.	400	`CreatePlatformApplication`
Nama Platform tidak valid	out-of-rangeNilai atau tidak valid diberikan untuk nama platform.	400	`CreatePlatformApplication`

Kesalahan	Deskripsi	HTTPSkode status	APIAksi
APNs— Principal bukan sertifikat yang valid	Sertifikat yang tidak valid diberikan untuk APNs kepala sekolah, yang merupakan sertifikat. SSL Untuk informasi selengkapnya, lihat CreatePlatformApplicationdi API Referensi Layanan Pemberitahuan Sederhana Amazon.	400	`CreatePlatformApplication`
APNs— Principal adalah sertifikat yang valid tetapi tidak dalam format.pem	Sertifikat yang valid yang tidak dalam format.pem diberikan untuk APNs kepala sekolah, yaitu sertifikat. SSL	400	`CreatePlatformApplication`
APNs— Principal adalah sertifikat yang kedaluwarsa	Sertifikat kedaluwarsa diberikan untuk kepala APNs sekolah, yang merupakan SSL sertifikat.	400	`CreatePlatformApplication`
APNs— Principal bukan sertifikat yang dikeluarkan Apple	Sertifikat yang dikeluarkan non-Apple diberikan untuk kepala APNs sekolah, yang merupakan SSL sertifikat.	400	`CreatePlatformApplication`

Kesalahan	Deskripsi	HTTPSkode status	APIAksi
APNs— Prinsipal tidak disediakan	APNsKepala sekolah, yang merupakan SSL sertifikat, tidak disediakan.	400	`CreatePlatformApplication`
APNs— Kredensi tidak disediakan	APNsKredensi, yang merupakan kunci pribadi, tidak disediakan. Untuk informasi selengkapnya, lihat CreatePlatformApplicationdi API Referensi Layanan Pemberitahuan Sederhana Amazon.	400	`CreatePlatformApplication`
APNs— Kredensi tidak dalam format.pem yang valid	APNsKredensialnya, yang merupakan kunci pribadi, tidak dalam format.pem yang valid.	400	`CreatePlatformApplication`
FCM— serverAPIKey tidak disediakan	FCMKredensi, yang merupakan API kuncinya, tidak disediakan. Untuk informasi selengkapnya, lihat CreatePlatformApplicationdi API Referensi Layanan Pemberitahuan Sederhana Amazon.	400	`CreatePlatformApplication`

Kesalahan	Deskripsi	HTTPSkode status	APIAksi
FCM— serverAPIKey kosong	FCMKredensi, yang merupakan API kuncinya, kosong.	400	`CreatePlatformApplication`
FCM- serverAPIKey adalah string nol	FCMKredensialnya, yang merupakan API kuncinya, adalah nol.	400	`CreatePlatformApplication`
FCM— serverAPIKey tidak valid	FCMKredensi, yang merupakan API kuncinya, tidak valid.	400	`CreatePlatformApplication`
ADM— clientsecret tidak disediakan	Rahasia klien yang diperlukan tidak disediakan.	400	`CreatePlatformApplication`
ADM- clientsecret adalah string null	String yang diperlukan untuk rahasia klien adalah null.	400	`CreatePlatformApplication`
ADM- client_secret adalah string kosong	String yang diperlukan untuk rahasia klien kosong.	400	`CreatePlatformApplication`
ADM- client_secret tidak valid	String yang diperlukan untuk rahasia klien tidak valid.	400	`CreatePlatformApplication`
ADM- client_id adalah string kosong	String yang diperlukan untuk ID klien kosong.	400	`CreatePlatformApplication`
ADM— clientId tidak disediakan	String yang diperlukan untuk ID klien tidak disediakan.	400	`CreatePlatformApplication`

Kesalahan	Deskripsi	HTTPSkode status	APIAksi
ADM- clientid adalah string nol	String yang diperlukan untuk ID klien adalah null.	400	`CreatePlatformApplication`
ADM— client_id tidak valid	String yang diperlukan untuk ID klien tidak valid.	400	`CreatePlatformApplication`
EventEndpointCreated memiliki format yang tidak valid ARN	EventEndpointCreated memiliki format yang tidak validARN.	400	`CreatePlatformApplication`
EventEndpointDeleted memiliki format yang tidak valid ARN	EventEndpointDeleted memiliki format yang tidak validARN.	400	`CreatePlatformApplication`
EventEndpointUpdated memiliki format yang tidak valid ARN	EventEndpointUpdated memiliki format yang tidak validARN.	400	`CreatePlatformApplication`
EventDeliveryAttemptFailure memiliki format yang tidak valid ARN	EventDeliveryAttemptFailure memiliki format yang tidak validARN.	400	`CreatePlatformApplication`
EventDeliveryFailure memiliki format yang tidak valid ARN	EventDeliveryFailure memiliki format yang tidak validARN.	400	`CreatePlatformApplication`
EventEndpointCreated bukan Topik yang ada	EventEndpointCreated bukan topik yang ada.	400	`CreatePlatformApplication`
EventEndpointDeleted bukan Topik yang ada	EventEndpointDeleted bukan topik yang ada.	400	`CreatePlatformApplication`

Kesalahan	Deskripsi	HTTPSkode status	APIAksi
EventEndpointUpdated bukan Topik yang ada	EventEndpointUpdated bukan topik yang ada.	400	`CreatePlatformApplication`
EventDeliveryAttemptFailure bukan Topik yang ada	EventDeliveryAttemptFailure bukan topik yang ada.	400	`CreatePlatformApplication`
EventDeliveryFailure bukan Topik yang ada	EventDeliveryFailure bukan topik yang ada.	400	`CreatePlatformApplication`
Platform ARN tidak valid	Platform ARN tidak valid.	400	`SetPlatformAttributes`
Platform ARN valid tetapi bukan milik pengguna	Platform ARN valid tetapi bukan milik pengguna.	400	`SetPlatformAttributes`
APNs— Principal bukan sertifikat yang valid	Sertifikat yang tidak valid diberikan untuk APNs kepala sekolah, yang merupakan sertifikat. SSL Untuk informasi selengkapnya, lihat CreatePlatformApplicationdi API Referensi Layanan Pemberitahuan Sederhana Amazon.	400	`SetPlatformAttributes`

Kesalahan	Deskripsi	HTTPSkode status	APIAksi
APNs— Principal adalah sertifikat yang valid tetapi tidak dalam format.pem	Sertifikat yang valid yang tidak dalam format.pem diberikan untuk APNs kepala sekolah, yaitu sertifika t. SSL	400	`SetPlatfo rmAttributes`
APNs— Principal adalah sertifikat yang kedaluwarsa	Sertifikat kedaluwar sa diberikan untuk kepala APNs sekolah, yang merupakan SSL sertifikat.	400	`SetPlatfo rmAttributes`
APNs— Principal bukan sertifikat yang dikeluarkan Apple	Sertifikat yang dikeluarkan non-Apple diberikan untuk kepala APNs sekolah, yang merupakan SSL sertifikat.	400	`SetPlatfo rmAttributes`
APNs— Prinsipal tidak disediakan	APNsKepala sekolah, yang merupakan SSL sertifikat, tidak disediakan.	400	`SetPlatfo rmAttributes`
APNs— Kredensi tidak disediakan	APNsKredensi, yang merupakan kunci pribadi, tidak disediakan. Untuk informasi selengkap nya, lihat CreatePla tformApplicationdi API Referensi Layanan Pemberitahuan Sederhana Amazon.	400	`SetPlatfo rmAttributes`

Kesalahan	Deskripsi	HTTPSkode status	APIAksi
APNs— Kredensi tidak dalam format.pem yang valid	APNsKredensialnya, yang merupakan kunci pribadi, tidak dalam format.pem yang valid.	400	`SetPlatformAttributes`
FCM— serverAPIKey tidak disediakan	FCMKredensi, yang merupakan API kuncinya, tidak disediakan. Untuk informasi selengkapnya, lihat CreatePlatformApplicationdi API Referensi Layanan Pemberitahuan Sederhana Amazon.	400	`SetPlatformAttributes`
FCM- serverAPIKey adalah string nol	FCMKredensialnya, yang merupakan API kuncinya, adalah nol.	400	`SetPlatformAttributes`
ADM— clientId tidak disediakan	String yang diperlukan untuk ID klien tidak disediakan.	400	`SetPlatformAttributes`
ADM- clientid adalah string nol	String yang diperlukan untuk ID klien adalah null.	400	`SetPlatformAttributes`
ADM— clientsecret tidak disediakan	Rahasia klien yang diperlukan tidak disediakan.	400	`SetPlatformAttributes`
ADM- clientsecret adalah string null	String yang diperlukan untuk rahasia klien adalah null.	400	`SetPlatformAttributes`

Kesalahan	Deskripsi	HTTPSkode status	APIAksi
EventEndpointUpdated memiliki format yang tidak valid ARN	EventEndpointUpdated memiliki format yang tidak validARN.	400	SetPlatformAttributes
EventEndpointDeleted memiliki format yang tidak valid ARN	EventEndpointDeleted memiliki format yang tidak validARN.	400	SetPlatformAttributes
EventEndpointUpdated memiliki format yang tidak valid ARN	EventEndpointUpdated memiliki format yang tidak validARN.	400	SetPlatformAttributes
EventDeliveryAttemptFailure memiliki format yang tidak valid ARN	EventDeliveryAttemptFailure memiliki format yang tidak validARN.	400	SetPlatformAttributes
EventDeliveryFailure memiliki format yang tidak valid ARN	EventDeliveryFailure memiliki format yang tidak validARN.	400	SetPlatformAttributes
EventEndpointCreated bukan Topik yang ada	EventEndpointCreated bukan topik yang ada.	400	SetPlatformAttributes
EventEndpointDeleted bukan Topik yang ada	EventEndpointDeleted bukan topik yang ada.	400	SetPlatformAttributes
EventEndpointUpdated bukan Topik yang ada	EventEndpointUpdated bukan topik yang ada.	400	SetPlatformAttributes
EventDeliveryAttemptFailure bukan Topik yang ada	EventDeliveryAttemptFailure bukan topik yang ada.	400	SetPlatformAttributes

Kesalahan	Deskripsi	HTTPSkode status	APIAksi
EventDeliveryFailure bukan Topik yang ada	EventDeliveryFailure bukan topik yang ada.	400	`SetPlatformAttributes`
Platform ARN tidak valid	Platform ARN tidak valid.	400	`GetPlatformApplicationAttributes`
Platform ARN valid tetapi bukan milik pengguna	Platform ARN ini valid, tetapi bukan milik pengguna.	403	`GetPlatformApplicationAttributes`
Token yang ditentukan tidak valid	Token yang ditentukan tidak valid.	400	`ListPlatformApplications`
Platform ARN tidak valid	Platform ARN tidak valid.	400	`ListEndpointsByPlatformApplication`
Platform ARN valid tetapi bukan milik pengguna	Platform ARN ini valid, tetapi bukan milik pengguna.	404	`ListEndpointsByPlatformApplication`
Token yang ditentukan tidak valid	Token yang ditentukan tidak valid.	400	`ListEndpointsByPlatformApplication`
Platform ARN tidak valid	Platform ARN tidak valid.	400	`DeletePlatformApplication`

Kesalahan	Deskripsi	HTTPSkode status	APIAksi
Platform ARN valid tetapi bukan milik pengguna	Platform ARN ini valid, tetapi bukan milik pengguna.	403	`DeletePlatformApplication`
Platform ARN tidak valid	Platform ARN tidak valid.	400	`CreatePlatformEndpoint`
Platform ARN valid tetapi bukan milik pengguna	Platform ARN ini valid, tetapi bukan milik pengguna.	404	`CreatePlatformEndpoint`
Token tidak ditentukan	Token tidak ditentukan.	400	`CreatePlatformEndpoint`
Token tidak memiliki panjang yang benar	Token tidak memiliki panjang yang benar.	400	`CreatePlatformEndpoint`
Data Pengguna Pelanggan terlalu besar	Data pengguna pelanggan tidak boleh lebih dari 2048 byte dalam pengkodean UTF -8.	400	`CreatePlatformEndpoint`
Endpoint ARN tidak valid	Endpoint tidak ARN valid.	400	`DeleteEndpoint`
Endpoint ARN valid tetapi bukan milik pengguna	Titik akhir ARN valid, tetapi bukan milik pengguna.	403	`DeleteEndpoint`
Endpoint ARN tidak valid	Endpoint tidak ARN valid.	400	`SetEndpointAttributes`
Endpoint ARN valid tetapi bukan milik pengguna	Titik akhir ARN valid, tetapi bukan milik pengguna.	403	`SetEndpointAttributes`

Kesalahan	Deskripsi	HTTPSkode status	APIAksi
Token tidak ditentuka n	Token tidak ditentuka n.	400	`SetEndpoi ntAttributes`
Token tidak memiliki panjang yang benar	Token tidak memiliki panjang yang benar.	400	`SetEndpoi ntAttributes`
Data Pengguna Pelanggan terlalu besar	Data pengguna pelanggan tidak boleh lebih dari 2048 byte dalam pengkodean UTF -8.	400	`SetEndpoi ntAttributes`
Endpoint ARN tidak valid	Endpoint tidak ARN valid.	400	`GetEndpoi ntAttributes`
Endpoint ARN valid tetapi bukan milik pengguna	Titik akhir ARN valid, tetapi bukan milik pengguna.	403	`GetEndpoi ntAttributes`
Target ARN tidak valid	ARNTargetnya tidak valid.	400	`Publish`
Target ARN valid tetapi bukan milik pengguna	ARNTargetnya valid, tetapi bukan milik pengguna.	403	`Publish`
Format pesan tidak valid	Format pesan tidak valid.	400	`Publish`
Ukuran pesan lebih besar daripada yang didukung oleh protokol/layanan akhir	Ukuran pesan lebih besar daripada yang didukung oleh protokol/layanan akhir.	400	`Publish`

Menggunakan atribut pesan langsung SNS waktu Amazon untuk pemberitahuan push seluler

Amazon Simple Notification Service (AmazonSNS) menyediakan dukungan untuk menyetel atribut pesan Time To Live (TTL) untuk pesan pemberitahuan push seluler. Ini merupakan tambahan dari kemampuan pengaturan yang ada TTL dalam badan SNS pesan Amazon untuk layanan notifikasi push seluler yang mendukung hal ini, seperti Amazon Device Messaging (ADM) dan Firebase Cloud Messaging (FCM) saat mengirim ke Android.

Atribut TTL pesan digunakan untuk menentukan metadata kedaluwarsa tentang pesan. Ini memungkinkan Anda menentukan jumlah waktu yang dimiliki layanan pemberitahuan push, seperti Apple Push Notification Service (APNs) atauFCM, untuk mengirimkan pesan ke titik akhir. Jika karena alasan tertentu (seperti perangkat seluler telah dimatikan) pesan tidak dapat dikirim dalam yang ditentukanTTL, maka pesan akan dihapus dan tidak ada upaya lebih lanjut untuk mengirimkannya akan dilakukan. Untuk menentukan TTL dalam atribut pesan, Anda dapat menggunakan AWS Management Console, kit pengembangan AWS perangkat lunak (SDKs), atau kueriAPI.

Topik

TTLatribut pesan untuk layanan pemberitahuan push

Berikut ini adalah daftar atribut TTL pesan untuk layanan pemberitahuan push yang dapat Anda gunakan untuk mengatur saat menggunakan AWS SDKs atau kueriAPI:

Layanan notifikasi push	TTLatribut pesan
Pesan Perangkat Amazon (ADM)	`AWS.SNS.MOBILE.ADM.TTL`
Layanan Pemberitahuan Push Apple (APNs)	`AWS.SNS.MOBILE.APNS.TTL`
Sandbox Layanan Pemberitahuan Push Apple (APNs_SANDBOX)	`AWS.SNS.MOBILE.APNS_SANDBOX.TTL`

Layanan notifikasi push	TTLatribut pesan
Baidu Cloud Push (Baidu)	`AWS.SNS.MOBILE.BAIDU.TTL`
Firebase Cloud Messaging (FCMsaat mengirim ke Android)	`AWS.SNS.MOBILE.FCM.TTL`
Layanan Pemberitahuan Push Windows (WNS)	`AWS.SNS.MOBILE.WNS.TTL`

Setiap layanan pemberitahuan push menangani TTL secara berbeda. Amazon SNS memberikan tampilan abstrak TTL atas semua layanan pemberitahuan push, yang membuatnya lebih mudah untuk ditentukanTTL. Ketika Anda menggunakan AWS Management Console untuk menentukan TTL (dalam detik), Anda hanya perlu memasukkan TTL nilai sekali dan Amazon kemudian SNS akan menghitung TTL untuk setiap layanan pemberitahuan push yang dipilih saat menerbitkan pesan.

TTLrelatif terhadap waktu publikasi. Sebelum menyerahkan pesan pemberitahuan push ke layanan pemberitahuan push tertentu, Amazon SNS menghitung waktu tinggal (waktu antara stempel waktu publikasi dan sebelum menyerahkan ke layanan pemberitahuan push) untuk pemberitahuan push dan meneruskan sisanya TTL ke layanan pemberitahuan push tertentu. Jika TTL lebih pendek dari waktu tinggal, Amazon tidak SNS akan mencoba mempublikasikannya.

Jika Anda menetapkan TTL untuk pesan notifikasi push, maka TTL nilainya harus berupa bilangan bulat positif, kecuali nilai `0` memiliki arti khusus untuk layanan notifikasi push—seperti with APNs dan FCM (saat mengirim ke Android). Jika TTL nilai disetel ke `0` dan layanan pemberitahuan push tidak memiliki arti khusus untuk`0`, maka Amazon SNS akan menghapus pesan. Untuk informasi selengkapnya tentang TTL parameter yang disetel ke `0` saat menggunakanAPNs, lihat Tabel A-3 Pengidentifikasi item untuk pemberitahuan jarak jauh dalam dokumentasi Penyedia API Biner.

Urutan prioritas untuk menentukan TTL

Prioritas yang SNS digunakan Amazon untuk menentukan pesan notifikasi push didasarkan pada urutan berikut, di mana angka terendah memiliki prioritas tertinggi: TTL

1. Atribut pesan TTL
2. Badan pesan TTL
3. Layanan pemberitahuan push default TTL (bervariasi per layanan)
4. Amazon SNS default TTL (4 minggu)

Jika Anda menetapkan TTL nilai yang berbeda (satu di atribut pesan dan satu lagi di badan pesan) untuk pesan yang sama, Amazon SNS akan memodifikasi TTL di badan pesan agar sesuai dengan TTL yang ditentukan dalam atribut pesan.

Menentukan TTL menggunakan AWS Management Console

1. Masuk ke SNSkonsol Amazon.
2. Pada panel navigasi, pilih Seluler, Notifikasi push.
3. Pada halaman Notifikasi push seluler, di bagian Aplikasi platform, pilih aplikasi.
4. Pada ***MyApplication*** halaman, di bagian Endpoints, pilih endpoint aplikasi dan kemudian pilih Publish message.
5. Di bagian Rincian pesan, masukkan TTL (jumlah detik yang dimiliki layanan pemberitahuan push untuk mengirimkan pesan ke titik akhir).
6. Pilih Publish message (Publikasikan Pesan).

Aplikasi SNS seluler Amazon didukung Wilayah

Saat ini, Anda dapat membuat aplikasi seluler di Wilayah berikut:

- AS Timur (Ohio)
- AS Timur (Virginia Utara)
- AS Barat (California Utara)
- AS Barat (Oregon)
- Afrika (Cape Town)
- Asia Pasifik (Hong Kong)
- Asia Pasifik (Jakarta)
- Asia Pasifik (Mumbai)
- Asia Pasifik (Osaka)
- Asia Pasifik (Seoul)
- Asia Pasifik (Singapura)
- Asia Pasifik (Sydney)
- Asia Pasifik (Tokyo)
- Kanada (Pusat)

- Eropa (Frankfurt)
- Eropa (Irlandia)
- Eropa (London)
- Eropa (Milan)
- Eropa (Paris)
- Eropa (Stockholm)
- Timur Tengah (Bahrain)
- Timur Tengah (UAE)
- Amerika Selatan (Sao Paulo)
- AWS GovCloud (AS-Barat)

Praktik terbaik untuk mengelola pemberitahuan push SNS seluler Amazon

Bagian ini menjelaskan praktik terbaik yang dapat membantu Anda meningkatkan keterlibatan pelanggan Anda.

Manajemen titik akhir

Masalah pengiriman mungkin terjadi jika token perangkat berubah karena tindakan pengguna pada perangkat (misalnya, aplikasi diinstal ulang pada perangkat), atau pembaruan sertifikat yang memengaruhi perangkat yang berjalan pada versi iOS tertentu. Ini adalah praktik terbaik yang disarankan oleh Apple untuk mendaftar APNs setiap kali aplikasi Anda diluncurkan.

Karena token perangkat tidak akan berubah setiap kali aplikasi dibuka oleh pengguna, idempoten `CreatePlatformEndpoint`APIdapat digunakan. Namun, ini dapat memperkenalkan duplikat untuk perangkat yang sama dalam kasus di mana token itu sendiri tidak valid, atau jika titik akhir valid tetapi dinonaktifkan (misalnya, ketidakcocokan lingkungan produksi dan kotak pasir).

Mekanisme manajemen token perangkat seperti yang ada di kode semu dapat digunakan.

Untuk informasi tentang mengelola dan memelihara token perangkat FCM v1, lihatSNSManajemen Amazon untuk titik akhir Firebase Cloud Messaging.

Pencatatan status pengiriman

Untuk memantau status pengiriman pemberitahuan push, kami sarankan Anda mengaktifkan pencatatan status pengiriman untuk aplikasi SNS platform Amazon Anda. Ini membantu

Anda memecahkan masalah kegagalan pengiriman karena log berisi kode respons penyedia yang dikembalikan dari layanan platform push. Untuk detail tentang mengaktifkan pencatatan status pengiriman, lihat Bagaimana cara mengakses log pengiriman SNS topik Amazon untuk pemberitahuan push? .

Pemberitahuan peristiwa

Untuk mengelola titik akhir dengan cara yang didorong oleh acara, Anda dapat menggunakan fungsionalitas pemberitahuan acara. Hal ini memungkinkan SNS topik Amazon yang dikonfigurasi untuk mengobarkan peristiwa ke pelanggan seperti fungsi Lambda, untuk peristiwa aplikasi platform pembuatan titik akhir, penghapusan, pembaruan, dan kegagalan pengiriman.

Pengaturan dan manajemen langganan SNS email Amazon

Anda dapat berlangganan alamat email ke SNS topik Amazon menggunakan AWS Management Console, AWS SDK for Java, atau AWS SDK for .NET.

> **Catatan**
>
> - Kustomisasi badan pesan email tidak didukung. Fitur pengiriman email ditujukan untuk memberikan pemberitahuan sistem internal, bukan pesan pemasaran.
> - Titik akhir email berlangganan langsung hanya didukung untuk topik standar.
> - Throughput pengiriman email dibatasi. Untuk informasi selengkapnya, lihat SNSKuota Amazon.

> **Important**
>
> Untuk mencegah penerima milis berhenti berlangganan semua penerima dari email SNS topik Amazon, lihat Mengatur langganan email yang memerlukan autentikasi untuk berhenti berlangganan dari Support. AWS

Berlangganan alamat email ke SNS topik Amazon menggunakan AWS Management Console

1. Masuk ke SNSkonsol Amazon.

2. Di panel navigasi kiri, pilih Subscriptions (Langganan).
3. Di halaman Subscriptions (Langganan), pilih Create subscription (Buat langganan).
4. Di halaman Create subscription (Buat langganan), di bagian Details (Detail), lakukan:
   a. Untuk Topik ARN, pilih Amazon Resource Name (ARN) dari suatu topik.
   b. Untuk Protokol, pilih Email.
   c. Untuk Titik Akhir, masukkan email address (alamat email).
   d. (Opsional) Untuk mengkonfigurasi kebijakan filter, perluas bagian Subscription filter policy (Kebijakan filter berlangganan). Untuk informasi selengkapnya, lihat Kebijakan filter SNS langganan Amazon.
   e. (Opsional) Untuk mengaktifkan pemfilteran berbasis muatan, konfigurasikan ke. `Filter Policy Scope` `MessageBody` Untuk informasi selengkapnya, lihat Lingkup kebijakan filter SNS langganan Amazon.
   f. (Opsional) Untuk mengonfigurasi antrean surat mati untuk berlanggana, perluas bagian Redrive policy (dead-letter queue) (Kebijakan redrive (antrean surat mati)). Untuk informasi selengkapnya, lihat Antrian SNS surat mati Amazon.
   g. Pilih Create subscription (Buat langganan).

      Konsol membuat langganan dan membuka halaman Rincian.

Anda harus mengonfirmasi berlangganan sebelum alamat email dapat mulai menerima pesan.

Untuk mengkonfirmasi berlangganan

1. Periksa kotak masuk email Anda dan pilih Konfirmasi langganan di email dari AmazonSNS.
2. Amazon SNS membuka browser web Anda dan menampilkan konfirmasi berlangganan dengan ID langganan Anda.

Berlangganan alamat email ke SNS topik Amazon menggunakan AWS SDK

Untuk menggunakan AWS SDK, Anda harus mengkonfigurasinya dengan kredensi Anda. Untuk informasi selengkapnya, lihat File konfigurasi dan kredensial bersama di Panduan Referensi Alat AWS SDKs dan Alat.

Contoh kode berikut menunjukkan cara menggunakan`Subscribe`.

.NET

AWS SDK for .NET

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

Berlangganan alamat email ke suatu topik.

```
        /// <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;
        }
```

Berlangganan antrian ke topik dengan filter opsional.

```
    /// <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;
    }
```

- Untuk API detailnya, lihat Berlangganan di AWS SDK for .NET APIReferensi.

C++

SDKuntuk C ++

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

Berlangganan alamat email ke suatu topik.

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

Berlangganan aplikasi seluler ke suatu topik.

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

Berlangganan fungsi Lambda ke suatu topik.

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

Berlangganan SQS antrian ke suatu topik.

```
        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;
           }
```

Berlangganan dengan filter ke topik.

```
       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;
}
```

- Untuk API detailnya, lihat Berlangganan di AWS SDK for C++ APIReferensi.

CLI

AWS CLI

Untuk berlangganan topik

`subscribe`Perintah berikut berlangganan alamat email ke topik yang ditentukan.

```
aws sns subscribe \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --protocol email \
    --notification-endpoint my-email@example.com
```

Output:

```
{
    "SubscriptionArn": "pending confirmation"
}
```

- Untuk API detailnya, lihat Berlangganan di Referensi AWS CLI Perintah.

Go

SDKuntuk Go V2

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

Berlangganan antrian ke topik dengan filter opsional.

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(ctx context.Context, topicArn string,
 queueArn string, filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(ctx, &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
```

```
}
```

- Untuk API detailnya, lihat Berlangganan di AWS SDK for Go APIReferensi.

Java

SDKuntuk Java 2.x

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

Berlangganan alamat email ke suatu topik.

```
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);
        }
    }
}
```

Berlangganan HTTP titik akhir ke suatu topik.

```
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);
        }
    }
}
```

Berlangganan fungsi Lambda ke suatu topik.

```
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 "";
    }
}
```

- Untuk API detailnya, lihat Berlangganan di AWS SDK for Java 2.x APIReferensi.

JavaScript

SDKuntuk JavaScript (v3)

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

Buat klien dalam modul terpisah dan ekspor klien tersebut.

```
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({});
```

Impor modul SDK dan klien dan panggil fileAPI.

```
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'
  // }
};
```

Berlangganan aplikasi seluler ke suatu topik.

```
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;
};
```

Berlangganan fungsi Lambda ke suatu topik.

```
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;
};
```

Berlangganan SQS antrian ke suatu topik.

```
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;
};
```

Berlangganan dengan filter ke topik.

```
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;
};
```

- Untuk informasi selengkapnya, lihat AWS SDK for JavaScript Panduan Developer.

- Untuk API detailnya, lihat Berlangganan di AWS SDK for JavaScript APIReferensi.

Kotlin

SDKuntuk Kotlin

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

Berlangganan alamat email ke suatu topik.

```
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()
    }
}
```

Berlangganan fungsi Lambda ke suatu topik.

```
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}")
    }
}
```

- Untuk API detailnya, lihat Berlangganan AWS SDKuntuk API referensi Kotlin.

PHP

SDKuntuk PHP

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

Berlangganan alamat email ke suatu topik.

```
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());
}
```

Berlangganan HTTP titik akhir ke suatu topik.

```
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());
}
```

- Untuk API detailnya, lihat Berlangganan di AWS SDK for PHP APIReferensi.

Python

SDKuntuk Python (Boto3)

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

Berlangganan alamat email ke suatu topik.

```
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
```

- Untuk API detailnya, lihat Berlangganan AWS SDKuntuk Referensi Python (Boto3). API

Ruby

SDKuntuk Ruby

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

Berlangganan alamat email ke suatu topik.

```
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
```

- Untuk informasi selengkapnya, lihat AWS SDK for Ruby Panduan Developer.
- Untuk API detailnya, lihat Berlangganan di AWS SDK for Ruby APIReferensi.

Rust

SDKuntuk Rust

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

Berlangganan alamat email ke suatu topik.

```
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(())
}
```

- Untuk API detailnya, lihat Berlangganan AWS SDKuntuk API referensi Rust.

SAP ABAP

SDKuntuk SAP ABAP

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

Berlangganan alamat email ke suatu topik.

```
    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.
```

- Untuk API detailnya, lihat Berlangganan AWS SDKuntuk SAP ABAP API referensi.

Praktik SNS terbaik Amazon

Berikut ini adalah praktik terbaik yang disarankan untuk menggunakan AmazonSNS:

Topik

Praktik terbaik SNS keamanan Amazon

AWS menyediakan banyak fitur keamanan untuk AmazonSNS. Tinjau fitur keamanan ini dalam konteks kebijakan keamanan Anda.

> **Note**
>
> Panduan untuk fitur keamanan ini berlaku untuk kasus penggunaan dan implementasi umum. Sebaiknya Anda meninjau praktik terbaik ini dalam konteks kasus penggunaan, arsitektur, dan model ancaman tertentu Anda.

Praktik terbaik pencegahan

Berikut ini adalah praktik terbaik keamanan preventif untuk Amazon. SNS

Topik

Memastikan topik tidak dapat diakses secara publik

Kecuali Anda secara eksplisit meminta siapa pun di internet untuk dapat membaca atau menulis ke SNS topik Amazon Anda, Anda harus memastikan bahwa topik Anda tidak dapat diakses publik (dapat diakses oleh semua orang di dunia atau oleh pengguna yang diautentikasi AWS).

- Hindari pembuatan kebijakan dengan `Principal` diatur ke `""`.
- Hindari penggunaan wildcard (`*`). Sebagai gantinya, beri nama pengguna tertentu.

Menerapkan akses hak istimewa yang paling rendah

Saat Anda memberikan izin, Anda memutuskan siapa yang menerimanya, topik yang menjadi tujuan izin, dan API tindakan spesifik yang ingin Anda izinkan untuk topik ini. Menerapkan prinsip hak istimewa paling rendah penting untuk mengurangi risiko keamanan. Tindakan ini juga membantu mengurangi efek negatif dari kesalahan atau niat jahat.

Ikuti saran keamanan standar pemberian hak istimewa paling rendah. Artinya, hanya berikan izin yang diperlukan untuk melakukan tugas tertentu. Anda dapat menerapkan hak istimewa paling rendah dengan menggunakan kombinasi kebijakan keamanan yang berkaitan dengan akses pengguna.

Amazon SNS menggunakan model penerbit-pelanggan, yang membutuhkan tiga jenis akses akun pengguna:

- Administrator – Akses untuk membuat, memodifikasi, dan menghapus topik. Administrator juga mengontrol kebijakan topik.
- Penerbit – Akses untuk mengirim pesan ke topik.
- Pelanggan – Akses untuk berlangganan topik.

Untuk informasi selengkapnya, lihat bagian berikut:

- Manajemen identitas dan akses di Amazon SNS
- SNSAPIIzin Amazon: Referensi tindakan dan sumber daya

Gunakan IAM peran untuk aplikasi dan AWS layanan yang memerlukan SNS akses Amazon

Untuk aplikasi atau AWS layanan, seperti AmazonEC2, untuk mengakses SNS topik Amazon, mereka harus menggunakan AWS kredensi yang valid dalam permintaan mereka AWS API. Karena kredensil ini tidak diputar secara otomatis, Anda tidak boleh menyimpan AWS kredensil secara langsung di aplikasi atau instance. EC2

Anda harus menggunakan IAM peran untuk mengelola kredensi sementara untuk aplikasi atau layanan yang perlu mengakses Amazon. SNS Saat Anda menggunakan peran, Anda tidak perlu mendistribusikan kredensi jangka panjang (seperti nama pengguna, kata sandi, dan kunci akses) ke EC2 instans atau AWS layanan, seperti. AWS Lambda Sebagai gantinya, peran menyediakan izin sementara yang dapat digunakan aplikasi saat mereka melakukan panggilan ke AWS sumber daya lain.

Untuk informasi selengkapnya, lihat IAMPeran dan Skenario Umum untuk Peran: Pengguna, Aplikasi, dan Layanan di Panduan IAM Pengguna.

Menerapkan enkripsi sisi server

Untuk mitigasi masalah kebocoran data, gunakan enkripsi saat istirahat untuk mengenkripsi pesan menggunakan kunci yang disimpan di lokasi berbeda dari lokasi yang menyimpan pesan Anda. Enkripsi sisi server (SSE) menyediakan enkripsi data saat istirahat. Amazon SNS mengenkripsi data Anda pada tingkat pesan saat menyimpannya, dan mendekripsi pesan untuk Anda saat Anda mengaksesnya. SSEmenggunakan kunci yang dikelola di AWS Key Management Service. Saat Anda mengautentikasi permintaan dan memiliki izin akses, tidak ada perbedaan dalam mengakses topik terenkripsi atau tidak terenkripsi.

Untuk informasi selengkapnya, lihat Mengamankan SNS data Amazon dengan enkripsi sisi server dan Mengelola kunci dan biaya SNS enkripsi Amazon.

Menegakkan enkripsi data saat transit

Dimungkinkan, tetapi tidak disarankan, untuk mempublikasikan pesan yang tidak dienkripsi selama transit dengan menggunakan. HTTP Namun, ketika suatu topik dienkripsi saat istirahat menggunakan AWS KMS, itu diperlukan untuk digunakan untuk menerbitkan pesan HTTPS untuk memastikan enkripsi baik saat istirahat maupun dalam perjalanan. Meskipun topik tidak secara otomatis menolak HTTP pesan, penggunaan HTTPS diperlukan untuk menjaga standar keamanan.

AWS merekomendasikan agar Anda menggunakan HTTPS alih-alihHTTP. Saat Anda menggunakanHTTPS, pesan secara otomatis dienkripsi selama transit, meskipun SNS topik itu sendiri tidak dienkripsi. TanpaHTTPS, penyerang berbasis jaringan dapat menguping lalu lintas jaringan atau memanipulasinya menggunakan serangan seperti. man-in-the-middle

Untuk menerapkan hanya koneksi terenkripsiHTTPS, tambahkan `aws:SecureTransport`kondisi dalam IAM kebijakan yang dilampirkan ke topik yang tidak terenkripsi. SNS Ini memaksa penerbit pesan untuk menggunakan HTTPS alih-alih. HTTP Anda dapat menggunakan kebijakan contoh berikut sebagai panduan:

```
{
  "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": "*"
    }
  ]
}
```

Pertimbangkan untuk menggunakan VPC titik akhir untuk mengakses Amazon SNS

Jika Anda memiliki topik yang harus dapat berinteraksi dengan Anda, tetapi topik ini sama sekali tidak boleh diekspos ke internet, gunakan VPC titik akhir untuk membatasi akses topik hanya ke host tertentuVPC. Anda dapat menggunakan kebijakan topik untuk mengontrol akses ke topik dari VPC titik akhir Amazon tertentu atau dari yang spesifikVPCs.

SNSVPCEndpoint Amazon menyediakan dua cara untuk mengontrol akses ke pesan Anda:

- Anda dapat mengontrol permintaan, pengguna, atau grup yang diizinkan melalui VPC titik akhir tertentu.
- Anda dapat mengontrol VPC titik akhir VPCs atau mana yang memiliki akses ke topik Anda menggunakan kebijakan topik.

Untuk informasi selengkapnya, silakan lihat Membuat titik akhir dan Membuat kebijakan VPC endpoint Amazon untuk Amazon SNS.

Pastikan langganan tidak dikonfigurasi untuk dikirim ke titik akhir http mentah

Hindari mengonfigurasi langganan untuk dikirim ke titik akhir http mentah. Selalu memiliki langganan yang dikirimkan ke nama domain endpoint. Misalnya, langganan yang dikonfigurasi untuk dikirim ke titik akhir`http://1.2.3.4/my-path`, harus diubah menjadi`http://my.domain.name/my-path`.

Praktik SNS SMS terbaik Amazon

> **Important**
>
> Panduan SNS SMS Pengembang Amazon telah diperbarui. Amazon SNS telah terintegrasi dengan AWS Olah Pesan Pengguna Akhir SMSpengiriman SMS pesan. Panduan ini berisi informasi terbaru tentang cara membuat, mengonfigurasi, dan mengelola SNS SMS pesan Amazon Anda.

Pengguna ponsel cenderung memiliki toleransi yang sangat rendah terhadap pesan yang tidak dimintaSMS. Tingkat respons untuk SMS kampanye yang tidak diminta hampir selalu rendah, dan oleh karena itu laba atas investasi Anda akan buruk.

Selain itu, operator ponsel terus mengaudit SMS pengirim massal. Mereka mencegah atau memblokir pesan dari nomor yang mereka tentukan untuk mengirim pesan yang tidak diminta.

Mengirim konten yang tidak diminta juga merupakan pelanggaran Kebijakan penggunaan AWS yang dapat diterima. SNSTim Amazon secara rutin mengaudit SMS kampanye, dan mungkin membatasi atau memblokir kemampuan Anda untuk mengirim pesan jika tampaknya Anda mengirim pesan yang tidak diminta.

Akhirnya, di banyak negara, wilayah, dan yurisdiksi, ada hukuman berat untuk mengirim pesan yang tidak SMS diminta. Misalnya, di Amerika Serikat, Undang-Undang Perlindungan Konsumen Telepon

(TCPA) menyatakan bahwa konsumen berhak atas kerugian $500—$1.500 (dibayar oleh pengirim) untuk setiap pesan yang tidak diminta yang mereka terima.

Bagian ini menjelaskan beberapa praktik terbaik yang dapat membantu Anda meningkatkan keterlibatan pelanggan dan menghindari hukuman yang mahal. Namun, perhatikan bahwa bagian ini tidak berisi nasihat hukum. Selalu konsultasikan dengan pengacara untuk mendapatkan nasihat hukum.

Topik

Mematuhi hukum, peraturan, dan persyaratan operator

Anda dapat menghadapi denda dan hukuman yang cukup berat jika Anda melanggar hukum dan peraturan tempat tinggal pelanggan Anda. Untuk alasan ini, penting untuk memahami undang-

undang yang terkait dengan SMS pengiriman pesan di setiap negara atau wilayah tempat Anda berbisnis.

Daftar berikut mencakup tautan ke undang-undang utama yang berlaku untuk SMS komunikasi di pasar utama di seluruh dunia.

- Amerika Serikat: Undang-Undang Perlindungan Konsumen Telepon tahun 1991, juga dikenal sebagaiTCPA, berlaku untuk jenis SMS pesan tertentu. Untuk informasi selengkapnya, lihat aturan dan regulasi di situs Federal Communications Commission.
- Inggris Raya: Peraturan Privasi dan Komunikasi Elektronik (EC Directive) 2003, juga dikenal sebagaiPECR, berlaku untuk jenis SMS pesan tertentu. Untuk informasi lebih lanjut, lihat Apa ituPECR? di situs web Kantor Komisaris Informasi Inggris.
- Uni Eropa: Petunjuk Privasi dan Komunikasi Elektronik 2002, kadang-kadang dikenal sebagai ePrivacy Petunjuk, berlaku untuk beberapa jenis SMS pesan. Untuk informasi selengkapnya, lihat dokumen hukum lengkap di situs Europa.eu.
- Kanada: Undang-Undang Memerangi Internet dan Spam Nirkabel, lebih dikenal sebagai Undang-Undang Anti-Spam Kanada atauCASL, berlaku untuk jenis SMS pesan tertentu. Untuk informasi selengkapnya, lihat dokumen hukum lengkap di situs Parliament of Canada.
- Jepang: Undang-Undang tentang Regulasi Transmisi Surat Elektronik Tertentu dapat berlaku untuk jenis SMS pesan tertentu. Untuk informasi lebih lanjut, lihat penanggulangan Jepang terhadap spam di situs web Kementerian Dalam Negeri dan Komunikasi Jepang.

Sebagai pengirim, undang-undang ini mungkin berlaku untuk Anda bahkan jika perusahaan atau organisasi Anda tidak berbasis di salah satu negara ini. Beberapa undang-undang dalam daftar ini awalnya dibuat untuk menangani email atau panggilan telepon yang tidak diminta, tetapi telah ditafsirkan atau diperluas untuk SMS diterapkan pada pesan juga. Negara dan wilayah lain mungkin memiliki undang-undang mereka sendiri terkait dengan transmisi SMS pesan. Konsultasikan dengan pengacara di setiap negara atau wilayah tempat pelanggan Anda berada untuk mendapatkan nasihat hukum.

Di banyak negara, operator lokal pada akhirnya memiliki wewenang untuk menentukan jenis arus lalu lintas melalui jaringan mereka. Ini berarti bahwa operator dapat memberlakukan pembatasan pada SMS konten yang melebihi persyaratan minimum undang-undang setempat.

Mendapatkan izin

Jangan pernah mengirim pesan ke penerima yang belum secara eksplisit meminta untuk menerima jenis pesan tertentu yang ingin Anda kirim. Jangan berbagi daftar opt-in, bahkan di antara organisasi dalam perusahaan yang sama.

Jika penerima dapat mendaftar untuk menerima pesan Anda dengan menggunakan formulir online, tambahkan sistem yang mencegah skrip otomatis berlangganan orang tanpa sepengetahuan mereka. Anda juga harus membatasi berapa kali pengguna dapat mengirimkan nomor telepon dalam satu sesi.

Saat Anda menerima permintaan SMS keikutsertaan, kirimkan pesan kepada penerima yang meminta mereka mengonfirmasi bahwa mereka ingin menerima pesan dari Anda. Jangan mengirim pesan tambahan kepada penerima itu sampai mereka mengonfirmasi langganan mereka. Pesan konfirmasi langganan mungkin menyerupai contoh berikut:

```
Text YES to join ExampleCorp alerts. 2 msgs/month. Msg & data rates may
apply. Reply HELP for help, STOP to cancel.
```

Pertahankan catatan yang mencakup tanggal, waktu, dan sumber setiap permintaan dan konfirmasi keikutsertaan menerima pesan. Hal ini mungkin berguna jika operator atau badan pengawas memintanya, dan juga dapat membantu Anda melakukan audit rutin terhadap daftar pelanggan Anda.

Alur kerja ikut serta

Dalam beberapa kasus (seperti pendaftaran Bebas Pulsa atau Kode Singkat AS) operator seluler mengharuskan Anda untuk memberikan maket atau tangkapan layar dari seluruh alur kerja keikutsertaan Anda. Maket atau tangkapan layar harus sangat mirip dengan alur kerja keikutsertaan yang akan diselesaikan penerima Anda.

Maket atau tangkapan layar Anda harus mencakup semua pengungkapan yang diperlukan yang tercantum di bawah ini untuk mempertahankan tingkat kepatuhan tertinggi.

Pengungkapan yang diperlukan

- Deskripsi kasus penggunaan pesan yang akan Anda kirim melalui program Anda.
- Ungkapan “Pesan dan tarif data mungkin berlaku.”
- Indikasi seberapa sering penerima akan menerima pesan dari Anda. Misalnya, program perpesanan berulang mungkin mengatakan “satu pesan per minggu.” Kata sandi satu kali atau

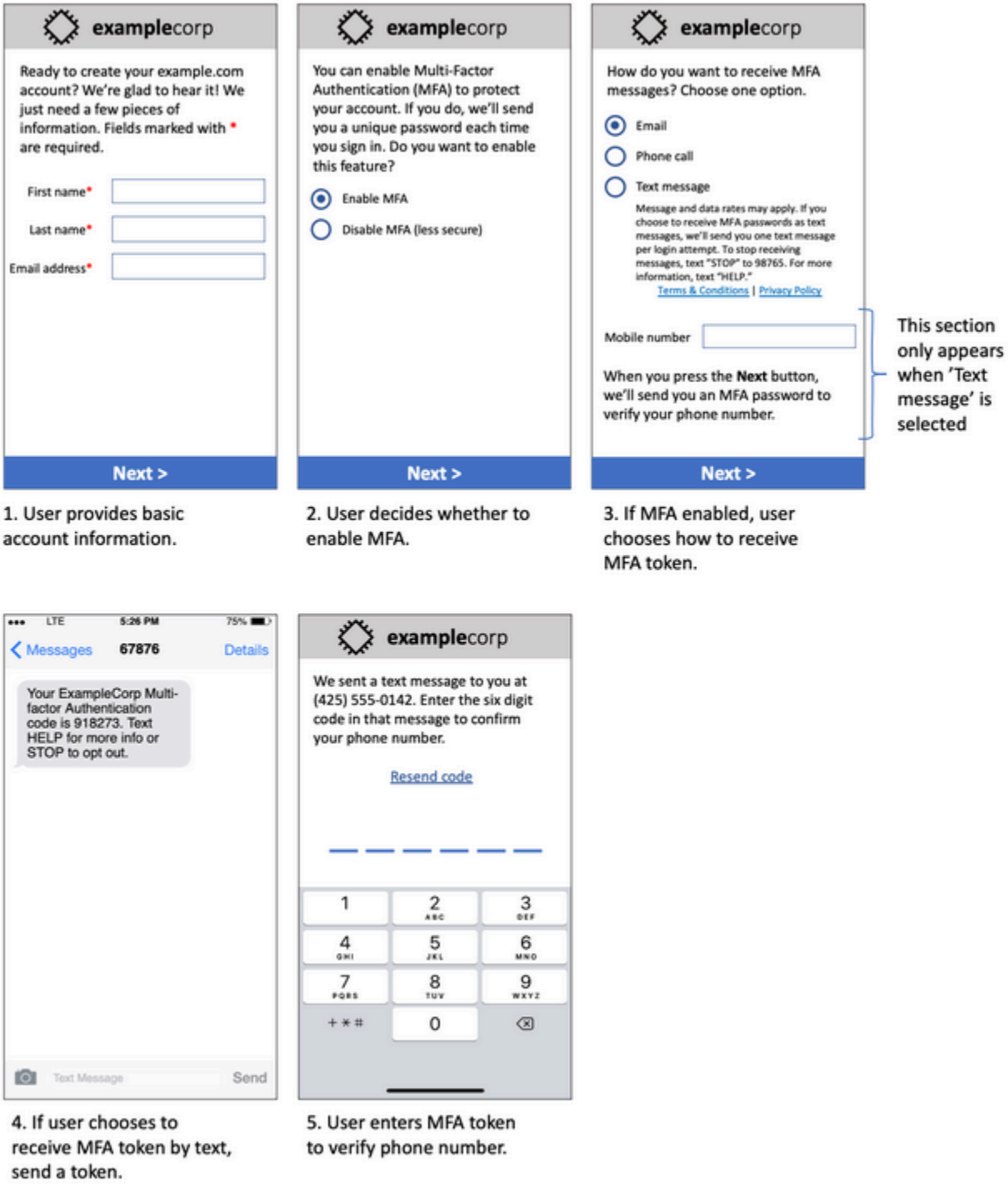
kasus penggunaan otentikasi multi-faktor mungkin mengatakan “frekuensi pesan bervariasi” atau “satu pesan per upaya login.”

- Tautan ke Syarat dan Ketentuan serta dokumen Kebijakan Privasi Anda.

Alasan penolakan umum untuk opt-in yang tidak sesuai

- Jika nama perusahaan yang diberikan tidak sesuai dengan apa yang disediakan dalam mockup atau screen shot. Setiap hubungan yang tidak jelas harus dijelaskan dalam deskripsi alur kerja opt-in.
- Jika tampaknya pesan akan dikirim ke penerima, tetapi tidak ada persetujuan yang dikumpulkan secara eksplisit sebelum melakukannya. Persetujuan eksplisit adalah persyaratan dari semua pesan.
- Jika tampaknya menerima pesan teks diperlukan untuk mendaftar ke layanan. Ini tidak sesuai jika alur kerja tidak memberikan alternatif apa pun selain menerima pesan keikutsertaan dalam bentuk lain seperti email atau panggilan suara.
- Jika bahasa opt-in disajikan sepenuhnya dalam Ketentuan Layanan. Pengungkapan harus selalu disajikan kepada penerima pada saat keikutsertaan daripada disimpan di dalam dokumen kebijakan yang ditautkan.
- Jika pelanggan memberikan persetujuan untuk menerima satu jenis pesan dari Anda dan Anda mengirimi mereka jenis pesan teks lainnya. Misalnya mereka setuju untuk menerima kata sandi satu kali tetapi juga dikirim polling dan pesan survei.
- Jika pengungkapan yang diperlukan (tercantum di atas) tidak disajikan kepada penerima.

Contoh berikut sesuai dengan persyaratan operator seluler untuk kasus penggunaan otentikasi multi-faktor.



Mockup kasus penggunaan otentikasi multi-faktor

Ini berisi teks dan gambar yang telah diselesaikan, dan ini menunjukkan seluruh alur keikutsertaan, lengkap dengan anotasi. Dalam alur keikutsertaan, pelanggan harus mengambil tindakan yang

berbeda dan disengaja untuk memberikan persetujuan mereka untuk menerima pesan teks dan berisi semua pengungkapan yang diperlukan.

Jenis alur kerja opt-in lainnya

Operator seluler juga akan menerima alur kerja opt-in di luar aplikasi dan situs web seperti opt-in verbal atau tertulis jika sesuai dengan apa yang diuraikan di atas. Alur kerja keikutsertaan yang sesuai dan skrip lisan atau tertulis akan mengumpulkan persetujuan eksplisit dari penerima untuk menerima jenis pesan tertentu. Contohnya termasuk skrip verbal yang digunakan agen dukungan untuk mengumpulkan persetujuan sebelum merekam ke dalam database layanan atau nomor telepon yang tercantum pada selebaran promosi. Untuk memberikan mockup dari jenis alur kerja opt-in ini, Anda dapat memberikan tangkapan layar skrip opt-in, materi pemasaran, atau database tempat nomor dikumpulkan. Operator seluler mungkin memiliki pertanyaan tambahan seputar kasus penggunaan ini jika keikutsertaan tidak jelas atau kasus penggunaan melebihi volume tertentu.

Jangan kirim ke daftar lama

Orang sering mengganti nomor telepon. Nomor telepon yang Anda kumpulkan persetujuan untuk dihubungi dua tahun lalu mungkin milik orang lain hari ini. Jangan gunakan daftar nomor telepon lama untuk program pesan baru; jika Anda melakukannya, Anda mungkin memiliki beberapa pesan gagal karena nomor tersebut tidak lagi dalam layanan, dan beberapa orang yang memilih keluar karena mereka tidak ingat memberi Anda persetujuan mereka sejak awal.

Audit daftar pelanggan Anda

Jika Anda mengirim SMS kampanye berulang, audit daftar pelanggan Anda secara teratur. Mengaudit daftar pelanggan Anda memastikan bahwa pelanggan yang menerima pesan Anda hanyalah mereka yang tertarik untuk menerimanya.

Saat Anda mengaudit daftar, kirim pesan kepada setiap pelanggan yang mengingatkan mereka bahwa mereka berlangganan, dan memberi mereka informasi tentang bagaimana cara berhenti berlangganan. Pesan pengingat mungkin menyerupai contoh berikut ini:

```
You're subscribed to ExampleCorp alerts. Msg & data rates may apply. Reply HELP for help, STOP to unsubscribe.
```

Simpan catatan

Simpan catatan yang menunjukkan kapan setiap pelanggan meminta untuk menerima SMS pesan dari Anda, dan pesan mana yang Anda kirim ke setiap pelanggan. Banyak negara dan wilayah di

seluruh dunia mengharuskan SMS pengirim untuk menyimpan catatan ini dengan cara yang dapat dengan mudah diambil. Operator seluler juga dapat meminta informasi ini dari Anda kapan saja. Informasi yang harus Anda berikan bervariasi berdasarkan negara atau wilayah. Untuk informasi lebih lanjut tentang persyaratan pencatatan, tinjau peraturan tentang SMS pesan komersial di setiap negara atau wilayah tempat pelanggan Anda berada.

Kadang-kadang, operator atau badan pengawas meminta kita untuk memberikan bukti bahwa pelanggan memilih untuk menerima pesan dari Anda. Dalam situasi ini, AWS Support hubungi Anda dengan daftar informasi yang dibutuhkan operator atau agensi. Jika Anda tidak dapat memberikan informasi yang diperlukan, kami dapat menghentikan sementara kemampuan Anda untuk mengirim SMS pesan tambahan.

Buat pesan Anda jelas, jujur, dan ringkas

SMSadalah media yang unik. character-per-message Batas 160 berarti pesan Anda harus ringkas. Teknik yang mungkin Anda gunakan di saluran komunikasi lain, seperti email, mungkin tidak berlaku untuk SMS saluran, dan bahkan mungkin tampak tidak jujur atau menipu ketika digunakan dengan pesan. SMS Jika konten dalam pesan Anda tidak selaras dengan praktik terbaik, penerima mungkin mengabaikan pesan Anda; dalam kasus terburuk, operator seluler mungkin mengidentifikasi pesan Anda sebagai spam dan memblokir pesan future dari nomor telepon Anda.

Bagian ini memberikan beberapa tips dan ide untuk membuat badan SMS pesan yang efektif.

Identifikasi diri Anda sebagai pengirim

Penerima Anda harus dapat segera memberi tahu bahwa pesan berasal dari Anda. Pengirim yang mengikuti praktik terbaik ini menyertakan nama pengenal (“nama program”) di awal setiap pesan.

Jangan lakukan ini:

```
Your account has been accessed from a new device. Reply Y to confirm.
```

Coba ini sebagai gantinya:

```
ExampleCorp Financial Alerts: You have logged in to your account from a
new device. Reply Y to confirm, or STOP to opt-out.
```

Jangan mencoba membuat pesan Anda terlihat seperti person-to-person pesan

Beberapa pemasar tergoda untuk menambahkan sentuhan pribadi ke SMS pesan mereka dengan membuat pesan mereka tampak berasal dari seseorang. Namun, teknik ini mungkin membuat pesan Anda tampak seperti upaya phishing.

Jangan lakukan ini:

```
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.
```

Coba ini sebagai gantinya:

```
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.
```

Hati-hati ketika berbicara tentang uang

Scammers sering memangsa keinginan orang untuk menabung dan menerima uang. Jangan membuat penawaran tampak terlalu bagus untuk menjadi kenyataan. Jangan gunakan iming-iming uang untuk menipu orang. Jangan gunakan simbol mata uang untuk menunjukkan uang.

Jangan lakukan ini:

```
Save big $$$ on your next car repair by going to https://
www.example.com.
```

Coba ini sebagai gantinya:

```
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.
```

Gunakan hanya karakter yang diperlukan

Merek sering cenderung melindungi merek dagang mereka dengan memasukkan simbol merek dagang seperti™ atau® dalam pesan mereka. Namun, simbol-simbol ini bukan bagian dari set standar karakter (dikenal sebagai GSM alfabet) yang dapat dimasukkan dalam pesan 160 SMS karakter. Ketika Anda mengirim pesan yang berisi salah satu karakter ini, pesan Anda secara otomatis dikirim menggunakan sistem pengkodean karakter yang berbeda, yang hanya mendukung

70 karakter per bagian pesan. Akibatnya, pesan Anda dapat dipecah menjadi beberapa bagian. Karena Anda ditagih untuk setiap bagian pesan yang Anda kirim, itu bisa dikenakan biaya lebih dari yang Anda harapkan untuk mengirim seluruh pesan. Selain itu, penerima Anda mungkin menerima beberapa pesan berurutan dari Anda, bukan satu pesan tunggal. Untuk informasi selengkapnya tentang pengkodean SMS karakter, lihatSMSbatas karakter di Amazon SNS.

Jangan lakukan ini:

```
ExampleCorp Alerts: Save 20% when you buy a new ExampleCorp Widget® at
example.com and use the promo code WIDGET.
```

Coba ini sebagai gantinya:

```
ExampleCorp Alerts: Save 20% when you buy a new ExampleCorp Widget(R) at
example.com and use the promo code WIDGET.
```

> **Note**
>
> Dua contoh sebelumnya hampir identik, tetapi contoh pertama berisi simbol Merek Dagang Terdaftar (®), yang bukan bagian dari alfabet. GSM Akibatnya, contoh pertama dikirim sebagai dua bagian pesan, sedangkan contoh kedua dikirim sebagai satu bagian pesan.

Gunakan tautan yang valid dan aman

Jika pesan Anda menyertakan tautan, periksa kembali tautan untuk memastikannya berfungsi. Uji tautan Anda di perangkat di luar jaringan perusahaan Anda untuk memastikan bahwa tautan teratasi dengan benar. Karena batas 160 karakter SMS pesan, sangat panjang URLs dapat dibagi menjadi beberapa pesan. Anda harus menggunakan domain pengalihan untuk memberikan yang dipersingkat. URLs Namun, Anda tidak boleh menggunakan layanan pemendekan tautan gratis seperti tinyurl.com atau bitly.com, karena operator cenderung memfilter pesan yang menyertakan tautan pada domain ini. Namun, Anda dapat menggunakan layanan pemendekan tautan berbayar selama tautan Anda mengarah ke domain yang didedikasikan untuk penggunaan eksklusif perusahaan atau organisasi Anda.

Jangan lakukan ini:

```
Go to https://tinyurl.com/4585y8mr today for a special offer!
```

Coba ini sebagai gantinya:

```
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.
```

Batasi jumlah singkatan yang Anda gunakan

Keterbatasan 160 karakter SMS saluran membuat beberapa pengirim percaya bahwa mereka perlu menggunakan singkatan secara ekstensif dalam pesan mereka. Namun, penggunaan singkatan yang berlebihan dapat tampak tidak profesional bagi banyak pembaca, dan dapat menyebabkan beberapa pengguna melaporkan pesan Anda sebagai spam. Sangat mungkin untuk menulis pesan yang koheren tanpa menggunakan jumlah singkatan yang berlebihan.

Jangan lakukan ini:

```
Get a gr8 deal on ExampleCorp widgets when u buy a 4-pack 2day.
```

Coba ini sebagai gantinya:

```
ExampleCorp Alerts: Today only—an exclusive deal on ExampleCorp Widgets
at example.com. Text STOP to opt-out.
```

Merespons dengan tepat

Saat penerima membalas pesan Anda, pastikan Anda merespons dengan informasi yang berguna. Misalnya, ketika pelanggan menanggapi salah satu pesan Anda dengan kata kunci HELP “”, kirimi mereka informasi tentang program yang mereka langgani, jumlah pesan yang akan Anda kirim setiap bulan, dan cara mereka dapat menghubungi Anda untuk informasi lebih lanjut. HELPRespons mungkin menyerupai contoh berikut:

```
HELP: ExampleCorp alerts: email help@example.com or call 425-555-0199. 2
msgs/month. Msg & data rates may apply. Reply STOP to cancel.
```

Ketika pelanggan membalas dengan kata kunci STOP “”, beri tahu mereka bahwa mereka tidak akan menerima pesan lebih lanjut. STOPRespons mungkin menyerupai contoh berikut:

```
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.
```

Sesuaikan pengiriman Anda berdasarkan keterlibatan

Prioritas pelanggan Anda dapat berubah seiring waktu. Jika pelanggan tidak lagi menganggap pesan Anda berguna, mereka mungkin memilih untuk tidak menerima pesan Anda sama sekali, atau bahkan melaporkan pesan Anda sebagai tidak diminta. Untuk alasan ini, penting untuk menyesuaikan praktik pengiriman berdasarkan keterlibatan pelanggan.

Untuk pelanggan yang jarang terlibat dengan pesan Anda, Anda harus menyesuaikan frekuensi pesan Anda. Misalnya, jika Anda mengirim pesan mingguan ke pelanggan yang terlibat, Anda dapat membuat rencana pengiriman bulanan terpisah untuk pelanggan yang kurang terlibat.

Lalu, hapus pelanggan yang benar-benar tidak terlibat dari daftar pelanggan Anda. Langkah ini mencegah pelanggan frustrasi terhadap pesan Anda. Ini juga menghemat pengeluaran Anda dan membantu melindungi reputasi Anda sebagai pengirim.

Kirim pada waktu yang tepat

Hanya kirim pesan selama jam kerja normal siang hari. Jika Anda mengirim pesan di waktu makan malam atau di tengah malam, ada kemungkinan bahwa pelanggan Anda akan berhenti berlangganan dari daftar Anda agar tidak terganggu. Selain itu, tidak masuk akal untuk mengirim SMS pesan ketika pelanggan Anda tidak dapat segera meresponsnya.

Jika Anda mengirim kampanye atau perjalanan ke audiens yang sangat besar, periksa ulang tingkat throughput untuk nomor originasi Anda. Bagilah jumlah penerima dengan tingkat throughput Anda untuk menentukan berapa lama waktu yang dibutuhkan untuk mengirim pesan ke semua penerima Anda.

Hindari keletihan lintas-saluran

Dalam kampanye Anda, jika Anda menggunakan beberapa saluran komunikasi (seperti emailSMS, dan pesan push), jangan mengirim pesan yang sama di setiap saluran. Ketika Anda mengirim pesan yang sama pada saat yang sama di lebih dari satu saluran, pelanggan Anda mungkin akan menganggap perilaku pengiriman Anda mengganggu, bukan membantu.

Gunakan kode pendek khusus

Jika Anda menggunakan kode pendek, gunakan kode pendek terpisah untuk setiap merek dan setiap jenis pesan. Misalnya, jika perusahaan Anda memiliki dua merek, gunakan kode pendek terpisah untuk masing-masing merek. Demikian pula, jika Anda mengirim pesan transaksional dan promosi, gunakan kode pendek terpisah untuk setiap jenis pesan. Untuk mempelajari lebih lanjut tentang

meminta kode pendek, lihat Meminta kode singkat untuk SMS pengiriman pesan AWS Olah Pesan Pengguna Akhir SMS di AWS Olah Pesan Pengguna Akhir SMS Panduan Pengguna.

Verifikasi nomor telepon tujuan

Ketika Anda mengirim SMS pesan melalui AmazonSNS, Anda ditagih untuk setiap bagian pesan yang Anda kirim. Harga yang Anda bayar per bagian pesan bervariasi di negara atau wilayah penerima. Untuk informasi selengkapnya tentang SMS harga, lihat SMSHarga AWS Seluruh Dunia.

Ketika Amazon SNS menerima permintaan untuk mengirim SMS pesan (sebagai hasil dari panggilan ke SendMessagesAPI, atau sebagai hasil dari kampanye atau perjalanan yang diluncurkan), Anda dikenakan biaya untuk mengirim pesan tersebut. Pernyataan ini benar bahkan jika penerima yang dituju tidak benar-benar menerima pesan. Misalnya, jika nomor telepon penerima tidak lagi dalam layanan, atau jika nomor yang Anda kirimi pesan bukan nomor ponsel yang valid, Anda masih ditagih untuk mengirim pesan.

Amazon SNS menerima permintaan yang valid untuk mengirim SMS pesan dan mencoba mengirimkannya. Untuk alasan ini, Anda harus memvalidasi bahwa nomor telepon yang Anda kirimi pesan adalah nomor ponsel yang valid. Anda dapat menggunakan AWS Olah Pesan Pengguna Akhir SMS untuk mengirim pesan pengujian untuk menentukan apakah nomor telepon valid dan jenis nomornya (seperti ponsel, telepon rumah, atau VoIP). Untuk informasi selengkapnya, lihat Mengirim pesan pengujian dengan SMS simulator di Panduan AWS Olah Pesan Pengguna Akhir SMS Pengguna.

Desain dengan mempertimbangkan redundansi

Untuk program pesan penting misi, kami menyarankan Anda mengonfigurasi Amazon SNS di lebih dari satu. Wilayah AWS Amazon SNS tersedia dalam beberapa Wilayah AWS. Untuk daftar lengkap Wilayah tempat Amazon SNS tersedia, lihat Referensi Umum AWS.

Nomor telepon yang Anda gunakan untuk SMS pesan—termasuk kode pendek, kode panjang, nomor bebas pulsa, dan 10 DLC nomor—tidak dapat direplikasi. Wilayah AWS Akibatnya, untuk menggunakan Amazon SNS di beberapa Wilayah, Anda harus meminta nomor telepon terpisah di setiap Wilayah tempat Anda ingin menggunakan AmazonSNS. Misalnya, jika Anda menggunakan kode singkat untuk mengirim pesan teks ke penerima di Amerika Serikat, Anda perlu meminta kode pendek terpisah di setiap kode Wilayah AWS yang Anda rencanakan untuk digunakan.

Di beberapa negara, Anda juga dapat menggunakan beberapa jenis nomor telepon untuk menambah redundansi. Misalnya, di Amerika Serikat, Anda dapat meminta kode pendek, 10 DLC nomor, dan nomor bebas pulsa. Masing-masing jenis nomor telepon ini mengambil rute yang berbeda ke

penerima. Memiliki beberapa jenis nomor telepon yang tersedia—baik dalam hal yang sama Wilayah AWS atau tersebar di beberapa Wilayah AWS lain—memberikan lapisan redundansi tambahan, yang dapat membantu meningkatkan ketahanan.

SMSbatas dan batasan

Untuk SMS batasan dan batasan, lihat SMSdan MMS batasan dan batasan dalam Panduan AWS Olah Pesan Pengguna Akhir SMS Pengguna.

Mengelola kata kunci keluar

SMSPenerima dapat menggunakan perangkat mereka untuk memilih keluar dari pesan dengan membalas dengan kata kunci. Untuk informasi selengkapnya, lihat Memilih keluar dari menerima pesan SMS.

CreatePool

Gunakan `CreatePool` API tindakan untuk membuat pool baru dan mengaitkan identitas originasi tertentu ke pool. Untuk informasi lebih lanjut, lihat CreatePooldi AWS Olah Pesan Pengguna Akhir SMS APIReferensi.

PutKeyword

Gunakan `PutKeyword` API tindakan untuk membuat atau memperbarui konfigurasi kata kunci pada nomor telepon atau kumpulan originasi. Untuk informasi lebih lanjut, lihat PutKeyworddi AWS Olah Pesan Pengguna Akhir SMS APIReferensi.

Mengelola pengaturan nomor

Untuk mengelola setelan kode pendek khusus dan kode panjang yang Anda minta dari AWS Support dan ditetapkan ke akun Anda, lihat Mengubah kemampuan nomor telepon dengan AWS CLI masuk AWS Olah Pesan Pengguna Akhir SMS.

SMSbatas karakter di Amazon SNS

Satu SMS pesan dapat berisi hingga 140 byte informasi. Jumlah karakter yang dapat Anda sertakan dalam satu SMS pesan tergantung pada jenis karakter yang terkandung dalam pesan tersebut.

Jika pesan Anda hanya menggunakan karakter dalam set karakter GSM 03.38, juga dikenal sebagai alfabet GSM 7-bit, pesan tersebut dapat berisi hingga 160 karakter. Jika pesan Anda berisi karakter apa pun yang berada di luar set karakter GSM 03.38, pesan tersebut dapat memiliki hingga 70

karakter. Saat Anda mengirim SMS pesan, Amazon SNS secara otomatis menentukan pengkodean yang paling efisien untuk digunakan.

Ketika pesan berisi lebih dari jumlah maksimum karakter, pesan dibagi menjadi beberapa bagian. Ketika pesan dibagi menjadi beberapa bagian, setiap bagian berisi informasi tambahan tentang bagian pesan yang mendahuluinya. Ketika perangkat penerima menerima bagian pesan yang dipisahkan dengan cara ini, ia menggunakan informasi tambahan ini untuk memastikan bahwa semua bagian pesan ditampilkan dalam urutan yang benar. Bergantung pada operator seluler dan perangkat penerima, beberapa pesan mungkin ditampilkan sebagai satu pesan, atau sebagai urutan pesan terpisah. Akibatnya jumlah karakter di setiap bagian pesan dikurangi menjadi 153 (untuk pesan yang hanya berisi GSM 03.38 karakter) atau 67 (untuk pesan yang berisi karakter lain). Anda dapat memperkirakan berapa banyak bagian pesan yang berisi pesan Anda sebelum Anda mengirimnya dengan menggunakan alat kalkulator SMS panjang, beberapa di antaranya tersedia secara online. Ukuran maksimum yang didukung dari pesan apa pun adalah 1600 GSM karakter atau 630 GSM non-karakter. Untuk informasi selengkapnya tentang throughput dan ukuran pesan, lihat batas SMS karakter di Amazon Pinpoint di Panduan Pengguna Amazon Pinpoint.

Untuk melihat jumlah bagian pesan untuk setiap pesan yang Anda kirim, Anda harus terlebih dahulu mengaktifkan pengaturan aliran Acara. Ketika Anda melakukannya, Amazon SNS menghasilkan `_SMS.SUCCESS` acara ketika pesan dikirimkan ke penyedia seluler penerima. Catatan `_SMS.SUCCESS` acara berisi atribut yang disebut`attributes.number_of_message_parts`. Atribut ini menentukan jumlah bagian pesan yang berisi pesan.

> **Important**
>
> Ketika Anda mengirim pesan yang berisi lebih dari satu bagian pesan, Anda akan dikenakan biaya untuk jumlah bagian pesan yang terkandung dalam pesan.

GSM03.38 set karakter

Tabel berikut mencantumkan semua karakter yang hadir dalam set karakter GSM 03.38. Jika Anda mengirim pesan yang hanya menyertakan karakter yang ditampilkan dalam tabel berikut, maka pesan tersebut dapat berisi hingga 160 karakter.

GSM03.38 karakter standar												
A	B	C	D	E	F	G	H	I	J	K	L	M

GSM03.38 karakter standar												
T	O	P	Q	R	D	T	U	V	W	X	T	Z
a	b	c	d	e	f	g	-h	saya	j	k	l	m
n	o	p	q	r	detik	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						

Set karakter GSM 03.38 mencakup beberapa simbol selain yang ditunjukkan pada tabel sebelumnya. Namun, masing-masing karakter ini dihitung sebagai dua karakter karena juga mencakup karakter pelarian yang tidak terlihat:

- ^
- {
- }
- \
- [
-]
- ~
- |
- €

Terakhir, set karakter GSM 03.38 juga mencakup karakter non-cetak berikut:

- Karakter luar angkasa.

- Kontrol umpan baris, yang menandakan akhir dari satu baris teks dan awal baris lainnya.
- Sebuah carriage return control, yang bergerak ke awal baris teks (biasanya mengikuti karakter line feed).
- Kontrol pelarian, yang secara otomatis ditambahkan ke karakter dalam daftar sebelumnya.

Contoh pesan

Bagian ini berisi beberapa contoh SMS pesan. Untuk setiap contoh, bagian ini menunjukkan jumlah total karakter, serta jumlah bagian pesan untuk pesan.

Contoh 1: Pesan panjang yang hanya berisi karakter dalam alfabet GSM 03.38

Pesan berikut hanya berisi karakter yang ada dalam alfabet GSM 03.38.

```
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.
```

Pesan sebelumnya berisi 180 karakter, sehingga harus dibagi menjadi beberapa bagian pesan. Ketika pesan dibagi menjadi beberapa bagian pesan, setiap bagian dapat berisi 153 GSM 03.38 karakter. Akibatnya, pesan ini dikirim sebagai 2 bagian pesan.

Contoh 2: Pesan yang berisi karakter multi-byte

Pesan berikut berisi beberapa karakter Mandarin, yang semuanya berada di luar alfabet GSM 03.38.

```
##################################################·####1994#7#########
```

Pesan sebelumnya berisi 71 karakter. Namun, karena hampir semua karakter dalam pesan berada di luar alfabet GSM 03.38, itu dikirim sebagai dua bagian pesan. Masing-masing bagian pesan ini dapat berisi maksimal 67 karakter.

Contoh 3: Pesan yang berisi satu GSM karakter non-

Pesan berikut berisi satu karakter yang bukan bagian dari alfabet GSM 03.38. Dalam contoh ini, karakter adalah kutipan tunggal penutup (’), yang merupakan karakter yang berbeda dari apostrof biasa (’). Aplikasi pengolah kata seperti Microsoft Word sering secara otomatis mengganti apostrof dengan menutup tanda kutip tunggal. Jika Anda menyusun SMS pesan Anda di Microsoft Word dan menempelkannya ke AmazonSNS, Anda harus menghapus karakter khusus ini dan menggantinya dengan apostrof.

```
John: Your appointment with Dr. Salazar’s office is scheduled for next
Thursday at 4:30pm. Reply YES to confirm, NO to reschedule.
```

Pesan sebelumnya berisi 130 karakter. Namun, karena berisi karakter kutipan tunggal penutup, yang bukan bagian dari alfabet GSM 03.38, itu dikirim sebagai dua bagian pesan.

Jika Anda mengganti karakter kutipan tunggal penutup dalam pesan ini dengan tanda kutip (yang merupakan bagian dari alfabet GSM 03.38), maka pesan dikirim sebagai bagian pesan tunggal.

Contoh kode untuk Amazon SNS menggunakan AWS SDKs

Contoh kode berikut menunjukkan cara menggunakan Amazon SNS dengan kit pengembangan AWS perangkat lunak (SDK).

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

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

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

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Memulai

Halo Amazon SNS

Contoh kode berikut menunjukkan cara memulai menggunakan AmazonSNS.

.NET

AWS SDK for .NET

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

```
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();
        }
    }
}
```

- Untuk API detailnya, lihat ListTopicsdi AWS SDK for .NET APIReferensi.

C++

SDKuntuk C ++

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

Kode untuk CMakeLists file.txtCMake.

```
# 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})
```

Kode untuk file sumber hello_sns.cpp.

```
#include <aws/core/Aws.h>
#include <aws/sns/SNSClient.h>
#include <aws/sns/model/ListTopicsRequest.h>
#include <iostream>

/*
 *  A "Hello SNS" starter application which initializes an Amazon Simple
 Notification
 *  Service (Amazon SNS) client and lists the SNS topics in the current account.
 *
 *  main function
 *
 *  Usage: 'hello_sns'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

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

        Aws::Vector<Aws::SNS::Model::Topic> allTopics;
        Aws::String nextToken; // Next token is used to handle a paginated
 response.
        do {
            Aws::SNS::Model::ListTopicsRequest request;

            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }

            const Aws::SNS::Model::ListTopicsOutcome outcome =
 snsClient.ListTopics(
                    request);

            if (outcome.IsSuccess()) {
```

```
                const Aws::Vector<Aws::SNS::Model::Topic> &paginatedTopics =
                        outcome.GetResult().GetTopics();
                if (!paginatedTopics.empty()) {
                    allTopics.insert(allTopics.cend(), paginatedTopics.cbegin(),
                                     paginatedTopics.cend());
                }
            }
            else {
                std::cerr << "Error listing topics " <<
 outcome.GetError().GetMessage()
                          << std::endl;
                return 1;
            }

            nextToken = outcome.GetResult().GetNextToken();
        } while (!nextToken.empty());

        std::cout << "Hello Amazon SNS! You have " << allTopics.size() << "
 topic"
                  << (allTopics.size() == 1 ? "" : "s") << " in your account."
                  << std::endl;

        if (!allTopics.empty()) {
            std::cout << "Here are your topic ARNs." << std::endl;
            for (const Aws::SNS::Model::Topic &topic: allTopics) {
                std::cout << "  * " << topic.GetTopicArn() << std::endl;
            }
        }
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Untuk API detailnya, lihat ListTopicsdi AWS SDK for C++ APIReferensi.

Go

SDKuntuk Go V2

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

```
package main

import (
 "context"
 "fmt"
 "log"

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

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
```

```
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
   fmt.Printf("\t%v\n", *topic.TopicArn)
  }
 }
}
```

- Untuk API detailnya, lihat ListTopicsdi AWS SDK for Go APIReferensi.

Java

SDKuntuk Java 2.x

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

```
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);
        }
    }
}
```

- Untuk API detailnya, lihat ListTopicsdi AWS SDK for Java 2.x APIReferensi.

JavaScript

SDKuntuk JavaScript (v3)

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

Inisialisasi SNS klien dan dan daftar topik di akun Anda.

```
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"));
};
```

- Untuk API detailnya, lihat ListTopicsdi AWS SDK for JavaScript APIReferensi.

Kotlin

SDKuntuk Kotlin

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

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

- Untuk API detailnya, lihat ListTopics AWSSDKAPIreferensi Kotlin.

Contoh kode

- Contoh dasar untuk Amazon SNS menggunakan AWS SDKs
  - Halo Amazon SNS
  - Tindakan untuk Amazon SNS menggunakan AWS SDKs
    - Gunakan CheckIfPhoneNumberIsOptedOut dengan AWS SDK atau CLI
    - Gunakan ConfirmSubscription dengan AWS SDK atau CLI
    - Gunakan CreateTopic dengan AWS SDK atau CLI
    - Gunakan DeleteTopic dengan AWS SDK atau CLI
    - Gunakan GetSMSAttributes dengan AWS SDK atau CLI
    - Gunakan GetTopicAttributes dengan AWS SDK atau CLI
    - Gunakan ListPhoneNumbersOptedOut dengan AWS SDK atau CLI
    - Gunakan ListSubscriptions dengan AWS SDK atau CLI
    - Gunakan ListTopics dengan AWS SDK atau CLI

Contoh dasar untuk Amazon SNS menggunakan AWS SDKs

Contoh kode berikut menunjukkan cara menggunakan dasar-dasar Amazon Simple Notification Service dengan AWS SDKs.

Contoh

Halo Amazon SNS

Contoh kode berikut menunjukkan cara memulai menggunakan AmazonSNS.

.NET

AWS SDK for .NET

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

```
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();
        }
    }
}
```

- Untuk API detailnya, lihat ListTopicsdi AWS SDK for .NET APIReferensi.

C++

SDKuntuk C ++

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

Kode untuk CMakeLists file.txtCMake.

```
# 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})
```

Kode untuk file sumber hello_sns.cpp.

```
#include <aws/core/Aws.h>
#include <aws/sns/SNSClient.h>
#include <aws/sns/model/ListTopicsRequest.h>
#include <iostream>

/*
 *  A "Hello SNS" starter application which initializes an Amazon Simple
 Notification
 *  Service (Amazon SNS) client and lists the SNS topics in the current account.
 *
 *  main function
 *
 *  Usage: 'hello_sns'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

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

        Aws::Vector<Aws::SNS::Model::Topic> allTopics;
        Aws::String nextToken; // Next token is used to handle a paginated
 response.
        do {
            Aws::SNS::Model::ListTopicsRequest request;

            if (!nextToken.empty()) {
                request.SetNextToken(nextToken);
            }
```

```
            const Aws::SNS::Model::ListTopicsOutcome outcome =
 snsClient.ListTopics(
                    request);

            if (outcome.IsSuccess()) {
                const Aws::Vector<Aws::SNS::Model::Topic> &paginatedTopics =
                        outcome.GetResult().GetTopics();
                if (!paginatedTopics.empty()) {
                    allTopics.insert(allTopics.cend(), paginatedTopics.cbegin(),
                                     paginatedTopics.cend());
                }
            }
            else {
                std::cerr << "Error listing topics " <<
 outcome.GetError().GetMessage()
                          << std::endl;
                return 1;
            }

            nextToken = outcome.GetResult().GetNextToken();
        } while (!nextToken.empty());

        std::cout << "Hello Amazon SNS! You have " << allTopics.size() << "
 topic"
                  << (allTopics.size() == 1 ? "" : "s") << " in your account."
                  << std::endl;

        if (!allTopics.empty()) {
            std::cout << "Here are your topic ARNs." << std::endl;
            for (const Aws::SNS::Model::Topic &topic: allTopics) {
                std::cout << "  * " << topic.GetTopicArn() << std::endl;
            }
        }
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return 0;
}
```

- Untuk API detailnya, lihat ListTopicsdi AWS SDK for C++ APIReferensi.

Go

SDKuntuk Go V2

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

```
package main

import (
 "context"
 "fmt"
 "log"

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

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 snsClient := sns.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the topics for your account.")
 var topics []types.Topic
 paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
 for paginator.HasMorePages() {
```

```
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get topics. Here's why: %v\n", err)
   break
  } else {
   topics = append(topics, output.Topics...)
  }
 }
 if len(topics) == 0 {
  fmt.Println("You don't have any topics!")
 } else {
  for _, topic := range topics {
   fmt.Printf("\t%v\n", *topic.TopicArn)
  }
 }
}
```

- Untuk API detailnya, lihat ListTopicsdi AWS SDK for Go APIReferensi.

Java

SDKuntuk Java 2.x

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

```
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);
        }
    }
}
```

- Untuk API detailnya, lihat ListTopicsdi AWS SDK for Java 2.x APIReferensi.

JavaScript

SDKuntuk JavaScript (v3)

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

Inisialisasi SNS klien dan dan daftar topik di akun Anda.

```
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"));
};
```

- Untuk API detailnya, lihat ListTopicsdi AWS SDK for JavaScript APIReferensi.

Kotlin

SDKuntuk Kotlin

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

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

- Untuk API detailnya, lihat ListTopics AWSSDKAPIreferensi Kotlin.

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Tindakan untuk Amazon SNS menggunakan AWS SDKs

Contoh kode berikut menunjukkan cara melakukan SNS tindakan Amazon individual dengan AWS SDKs. Setiap contoh menyertakan tautan ke GitHub, di mana Anda dapat menemukan instruksi untuk mengatur dan menjalankan kode.

Kutipan ini menyebut Amazon SNS API dan merupakan kutipan kode dari program yang lebih besar yang harus dijalankan dalam konteks. Anda dapat melihat tindakan dalam konteks diSkenario untuk Amazon SNS menggunakan AWS SDKs .

Contoh berikut hanya mencakup tindakan yang paling umum digunakan. Untuk daftar lengkapnya, lihat APIReferensi Layanan Pemberitahuan Sederhana Amazon.

Contoh

- Gunakan CheckIfPhoneNumberIsOptedOut dengan AWS SDK atau CLI
- Gunakan ConfirmSubscription dengan AWS SDK atau CLI
- Gunakan CreateTopic dengan AWS SDK atau CLI
- Gunakan DeleteTopic dengan AWS SDK atau CLI
- Gunakan GetSMSAttributes dengan AWS SDK atau CLI
- Gunakan GetTopicAttributes dengan AWS SDK atau CLI
- Gunakan ListPhoneNumbersOptedOut dengan AWS SDK atau CLI
- Gunakan ListSubscriptions dengan AWS SDK atau CLI
- Gunakan ListTopics dengan AWS SDK atau CLI
- Gunakan Publish dengan AWS SDK atau CLI
- Gunakan SetSMSAttributes dengan AWS SDK atau CLI
- Gunakan SetSubscriptionAttributes dengan AWS SDK atau CLI
- Gunakan SetSubscriptionAttributesRedrivePolicy dengan AWS SDK
- Gunakan SetTopicAttributes dengan AWS SDK atau CLI
- Gunakan Subscribe dengan AWS SDK atau CLI
- Gunakan TagResource dengan AWS SDK atau CLI
- Gunakan Unsubscribe dengan AWS SDK atau CLI

Gunakan **`CheckIfPhoneNumberIsOptedOut`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`CheckIfPhoneNumberIsOptedOut`.

.NET

AWS SDK for .NET

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

```
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}");
            }
        }
    }
```

- Untuk API detailnya, lihat CheckIfPhoneNumberIsOptedOutdi AWS SDK for .NET APIReferensi.

CLI

AWS CLI

Untuk memeriksa SMS pesan opt-out untuk nomor telepon

`check-if-phone-number-is-opted-out`Contoh berikut memeriksa apakah nomor telepon yang ditentukan dipilih untuk tidak menerima SMS pesan dari AWS akun saat ini.

```
aws sns check-if-phone-number-is-opted-out \
    --phone-number +1555550100
```

Output:

```
{
    "isOptedOut": false
}
```

- Untuk API detailnya, lihat CheckIfPhoneNumberIsOptedOutdi Referensi AWS CLI Perintah.

Java

SDKuntuk Java 2.x

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

```
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);
        }
    }
}
```

- Untuk API detailnya, lihat CheckIfPhoneNumberIsOptedOutdi AWS SDK for Java 2.x APIReferensi.

JavaScript

SDKuntuk JavaScript (v3)

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

Buat klien dalam modul terpisah dan ekspor klien tersebut.

```
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({});
```

Impor modul SDK dan klien dan panggil fileAPI.

```
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;
};
```

- Untuk informasi selengkapnya, lihat AWS SDK for JavaScript Panduan Developer.
- Untuk API detailnya, lihat CheckIfPhoneNumberIsOptedOutdi AWS SDK for JavaScript APIReferensi.

PHP

SDKuntuk PHP

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

```
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());
}
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for PHP.
- Untuk API detailnya, lihat CheckIfPhoneNumberIsOptedOutdi AWS SDK for PHP APIReferensi.

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Gunakan **`ConfirmSubscription`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`ConfirmSubscription`.

CLI

AWS CLI

Untuk mengonfirmasi langganan

`confirm-subscription`Perintah berikut menyelesaikan proses konfirmasi yang dimulai saat Anda berlangganan SNS topik bernama. `my-topic` Parameter --token berasal dari pesan konfirmasi yang dikirim ke titik akhir notifikasi yang ditentukan dalam panggilan berlangganan.

```
aws sns confirm-subscription \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
```

```
    --
token 2336412f37fb687f5d51e6e241d7700ae02f7124d8268910b858cb4db727ceeb2474bb937929d3bdd7c
```

Output:

```
{
    "SubscriptionArn": "arn:aws:sns:us-west-2:123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f"
}
```

- Untuk API detailnya, lihat ConfirmSubscriptiondi Referensi AWS CLI Perintah.

Java

SDKuntuk Java 2.x

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

```
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);
        }
    }
}
```

- Untuk API detailnya, lihat ConfirmSubscriptiondi AWS SDK for Java 2.x APIReferensi.

JavaScript

SDKuntuk JavaScript (v3)

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

Buat klien dalam modul terpisah dan ekspor klien tersebut.

```
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({});
```

Impor modul SDK dan klien dan panggil fileAPI.

```
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;
};
```

- Untuk informasi selengkapnya, lihat AWS SDK for JavaScript Panduan Developer.
- Untuk API detailnya, lihat ConfirmSubscriptiondi AWS SDK for JavaScript APIReferensi.

PHP

SDKuntuk PHP

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

```
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());
}
```

- Untuk API detailnya, lihat ConfirmSubscriptiondi AWS SDK for PHP APIReferensi.

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Gunakan `CreateTopic` dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`CreateTopic`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Membuat dan mempublikasikan ke FIFO topik
- Publikasikan pesan ke antrian

.NET

AWS SDK for .NET

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

Buat topik dengan nama tertentu.

```
    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;
        }
    }
```

Buat topik baru dengan nama dan atribut spesifik FIFO dan de-duplikasi.

```
    /// <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;
    }
```

- Untuk API detailnya, lihat CreateTopicdi AWS SDK for .NET APIReferensi.

C++

SDKuntuk C ++

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

```
//! Create an Amazon Simple Notification Service (Amazon SNS) topic.
```

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

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

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

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

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

    return outcome.IsSuccess();
}
```

- Untuk API detailnya, lihat CreateTopicdi AWS SDK for C++ APIReferensi.

CLI

AWS CLI

Untuk membuat SNS topik

`create-topic`Contoh berikut membuat SNS topik bernama`my-topic`.

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

Output:

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

Untuk informasi selengkapnya, lihat Menggunakan Antarmuka Baris AWS Perintah dengan Amazon SQS dan Amazon SNS di Panduan Pengguna Antarmuka Baris AWS Perintah.

- Untuk API detailnya, lihat CreateTopicdi Referensi AWS CLI Perintah.

Go

SDKuntuk Go V2

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

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
```

```
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(ctx context.Context, topicName string,
 isFifoTopic bool, contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
 }
 topic, err := actor.SnsClient.CreateTopic(ctx, &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
  topicArn = *topic.TopicArn
 }

 return topicArn, err
}
```

- Untuk API detailnya, lihat CreateTopicdi AWS SDK for Go APIReferensi.

Java

SDKuntuk Java 2.x

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

```
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 "";
    }
}
```

- Untuk API detailnya, lihat CreateTopicdi AWS SDK for Java 2.x APIReferensi.

JavaScript

SDKuntuk JavaScript (v3)

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

Buat klien dalam modul terpisah dan ekspor klien tersebut.

```
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({});
```

Impor modul SDK dan klien dan panggil fileAPI.

```
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;
};
```

- Untuk informasi selengkapnya, lihat AWS SDK for JavaScript Panduan Developer.
- Untuk API detailnya, lihat CreateTopicdi AWS SDK for JavaScript APIReferensi.

Kotlin

SDKuntuk Kotlin

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

```
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()
    }
}
```

- Untuk API detailnya, lihat CreateTopic AWSSDKAPIreferensi Kotlin.

PHP

SDKuntuk PHP

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

```
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());
}
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for PHP.
- Untuk API detailnya, lihat CreateTopicdi AWS SDK for PHP APIReferensi.

Python

SDKuntuk Python (Boto3)

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

```
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
```

- Untuk API detailnya, lihat CreateTopic AWSSDKReferensi Python (Boto3). API

Ruby

SDKuntuk Ruby

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

```
# 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
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for Ruby.
- Untuk API detailnya, lihat CreateTopicdi AWS SDK for Ruby APIReferensi.

Rust

SDKuntuk Rust

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

```
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(())
}
```

- Untuk API detailnya, lihat CreateTopic AWSSDKuntuk API referensi Rust.

SAP ABAP

SDKuntuk SAP ABAP

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

```
   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.
```

- Untuk API detailnya, lihat CreateTopic AWSSDKuntuk SAP ABAP API referensi.

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Gunakan **`DeleteTopic`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`DeleteTopic`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Publikasikan pesan ke antrian

.NET

AWS SDK for .NET

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

Hapus topik berdasarkan topiknyaARN.

```
/// <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;
}
```

- Untuk API detailnya, lihat DeleteTopicdi AWS SDK for .NET APIReferensi.

C++

SDKuntuk C ++

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

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

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

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

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

    return outcome.IsSuccess();
}
```

- Untuk API detailnya, lihat DeleteTopicdi AWS SDK for C++ APIReferensi.

CLI

AWS CLI

Untuk menghapus SNS topik

`delete-topic`Contoh berikut menghapus SNS topik yang ditentukan.

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

Perintah ini tidak menghasilkan output.

- Untuk API detailnya, lihat DeleteTopicdi Referensi AWS CLI Perintah.

Go

SDKuntuk Go V2

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

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(ctx context.Context, topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(ctx, &sns.DeleteTopicInput{
  TopicArn: aws.String(topicArn)})
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
```

```
 }
 return err
}
```

- Untuk API detailnya, lihat DeleteTopicdi AWS SDK for Go APIReferensi.

Java

SDKuntuk Java 2.x

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

```
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);
        }
    }
}
```

- Untuk API detailnya, lihat DeleteTopicdi AWS SDK for Java 2.x APIReferensi.

JavaScript

SDKuntuk JavaScript (v3)

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

Buat klien dalam modul terpisah dan ekspor klien tersebut.

```
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({});
```

Impor modul SDK dan klien dan panggil fileAPI.

```
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
  //   }
```

```
  // }
};
```

- Untuk informasi selengkapnya, lihat AWS SDK for JavaScript Panduan Developer.
- Untuk API detailnya, lihat DeleteTopicdi AWS SDK for JavaScript APIReferensi.

Kotlin

SDKuntuk Kotlin

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

```
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.")
    }
}
```

- Untuk API detailnya, lihat DeleteTopic AWSSDKAPIreferensi Kotlin.

PHP

SDKuntuk PHP

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

```
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());
}
```

- Untuk API detailnya, lihat DeleteTopicdi AWS SDK for PHP APIReferensi.

Python

SDKuntuk Python (Boto3)

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

```
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
```

- Untuk API detailnya, lihat DeleteTopic AWSSDKReferensi Python (Boto3). API

SAP ABAP

SDKuntuk SAP ABAP

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

```
    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.
```

- Untuk API detailnya, lihat DeleteTopic AWSSDKuntuk SAP ABAP API referensi.

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Gunakan **`GetSMSAttributes`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`GetSMSAttributes`.

C++

SDKuntuk C ++

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

```
//! Retrieve the default settings for sending SMS messages from your AWS account
 by using
```

```
//! Amazon Simple Notification Service (Amazon SNS).
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::getSMSType(const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::GetSMSAttributesRequest request;
    //Set the request to only retrieve the DefaultSMSType setting.
    //Without the following line, GetSMSAttributes would retrieve all settings.
    request.AddAttributes("DefaultSMSType");

    const Aws::SNS::Model::GetSMSAttributesOutcome outcome =
 snsClient.GetSMSAttributes(
            request);

    if (outcome.IsSuccess()) {
        const Aws::Map<Aws::String, Aws::String> attributes =
                outcome.GetResult().GetAttributes();
        if (!attributes.empty()) {
            for (auto const &att: attributes) {
                std::cout << att.first << ":  " << att.second << std::endl;
            }
        }
        else {
            std::cout
                    << "AwsDoc::SNS::getSMSType - an empty map of attributes was
 retrieved."
                    << std::endl;
        }
    }
    else {
        std::cerr << "Error while getting SMS Type: '"
                  << outcome.GetError().GetMessage()
                  << "'" << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Untuk API detailnya, lihat G etSMSAttributes di AWS SDK for C++ APIReferensi.

CLI

AWS CLI

Untuk mencantumkan atribut SMS pesan default

`get-sms-attributes`Contoh berikut mencantumkan atribut default untuk mengirim SMS pesan.

```
aws sns get-sms-attributes
```

Output:

```
{
    "attributes": {
        "DefaultSenderID": "MyName"
    }
}
```

- Untuk API detailnya, lihat G etSMSAttributes di Referensi AWS CLI Perintah.

Java

SDKuntuk Java 2.x

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

```
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");
    }
}
```

- Untuk API detailnya, lihat G etSMSAttributes di AWS SDK for Java 2.x APIReferensi.

JavaScript

SDKuntuk JavaScript (v3)

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

Buat klien dalam modul terpisah dan ekspor klien tersebut.

```
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({});
```

Impor modul SDK dan klien dan panggil fileAPI.

```
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;
};
```

- Untuk informasi selengkapnya, lihat AWS SDK for JavaScript Panduan Developer.
- Untuk API detailnya, lihat G etSMSAttributes di AWS SDK for JavaScript APIReferensi.

PHP

SDKuntuk PHP

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

```
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());
}
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for PHP.
- Untuk API detailnya, lihat G etSMSAttributes di AWS SDK for PHP APIReferensi.

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Gunakan **`GetTopicAttributes`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`GetTopicAttributes`.

.NET

AWS SDK for .NET

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

```
    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");
            }
        }
    }
```

- Untuk API detailnya, lihat GetTopicAttributesdi AWS SDK for .NET APIReferensi.

C++

SDKuntuk C ++

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

```
//! Retrieve the properties of an Amazon Simple Notification Service (Amazon SNS)
 topic.
/*!
  \param topicARN: The Amazon Resource Name (ARN) for an Amazon SNS topic.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::getTopicAttributes(const Aws::String &topicARN,
                                     const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);
    Aws::SNS::Model::GetTopicAttributesRequest request;
    request.SetTopicArn(topicARN);

    const Aws::SNS::Model::GetTopicAttributesOutcome outcome =
 snsClient.GetTopicAttributes(
            request);

    if (outcome.IsSuccess()) {
        std::cout << "Topic Attributes:" << std::endl;
        for (auto const &attribute: outcome.GetResult().GetAttributes()) {
            std::cout << "  * " << attribute.first << " : " << attribute.second
                      << std::endl;
        }
    }
    else {
        std::cerr << "Error while getting Topic attributes "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
```

```
}
```

- Untuk API detailnya, lihat GetTopicAttributesdi AWS SDK for C++ APIReferensi.

CLI

AWS CLI

Untuk mengambil atribut topik

`get-topic-attributes`Contoh berikut menampilkan atribut untuk topik yang ditentukan.

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

Output:

```
{
    "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"
    }
}
```

- Untuk API detailnya, lihat GetTopicAttributesdi Referensi AWS CLI Perintah.

Java

SDKuntuk Java 2.x

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

```
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);
        }
    }
}
```

- Untuk API detailnya, lihat GetTopicAttributesdi AWS SDK for Java 2.x APIReferensi.

JavaScript

SDKuntuk JavaScript (v3)

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

Buat klien dalam modul terpisah dan ekspor klien tersebut.

```
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({});
```

Impor modul SDK dan klien dan panggil fileAPI.

```
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;
};
```

- Untuk informasi selengkapnya, lihat AWS SDK for JavaScript Panduan Developer.
- Untuk API detailnya, lihat GetTopicAttributesdi AWS SDK for JavaScript APIReferensi.

SDKuntuk JavaScript (v2)

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

Impor modul SDK dan klien dan panggil fileAPI.

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

- Untuk informasi selengkapnya, lihat AWS SDK for JavaScript Panduan Developer.
- Untuk API detailnya, lihat GetTopicAttributesdi AWS SDK for JavaScript APIReferensi.

Kotlin

SDKuntuk Kotlin

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

```
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}")
    }
}
```

- Untuk API detailnya, lihat GetTopicAttributes AWSSDKAPIreferensi Kotlin.

PHP

SDKuntuk PHP

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

```
$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());
}
```

- Untuk API detailnya, lihat GetTopicAttributesdi AWS SDK for PHP APIReferensi.

SAP ABAP

SDKuntuk SAP ABAP

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

```
   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.
```

- Untuk API detailnya, lihat GetTopicAttributes AWSSDKuntuk SAP ABAP API referensi.

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Gunakan `ListPhoneNumbersOptedOut` dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`ListPhoneNumbersOptedOut`.

CLI

AWS CLI

Untuk mencantumkan SMS opt-out pesan

`list-phone-numbers-opted-out`Contoh berikut mencantumkan nomor telepon yang dipilih untuk tidak menerima SMS pesan.

```
aws sns list-phone-numbers-opted-out
```

Output:

```
{
    "phoneNumbers": [
        "+15555550100"
    ]
}
```

- Untuk API detailnya, lihat ListPhoneNumbersOptedOutdi Referensi AWS CLI Perintah.

Java

SDKuntuk Java 2.x

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

```
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);
        }
    }
}
```

- Untuk API detailnya, lihat ListPhoneNumbersOptedOutdi AWS SDK for Java 2.x APIReferensi.

PHP

SDKuntuk PHP

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

```
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());
}
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for PHP.
- Untuk API detailnya, lihat ListPhoneNumbersOptedOutdi AWS SDK for PHP APIReferensi.

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Gunakan **`ListSubscriptions`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`ListSubscriptions`.

.NET

AWS SDK for .NET

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

```
    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();
            }
        }
    }
```

- Untuk API detailnya, lihat ListSubscriptionsdi AWS SDK for .NET APIReferensi.

C++

SDKuntuk C ++

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

```
//! Retrieve a list of Amazon Simple Notification Service (Amazon SNS)
 subscriptions.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::listSubscriptions(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::String nextToken; // Next token is used to handle a paginated response.
    bool result = true;
    Aws::Vector<Aws::SNS::Model::Subscription> subscriptions;
    do {
        Aws::SNS::Model::ListSubscriptionsRequest request;

        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        const Aws::SNS::Model::ListSubscriptionsOutcome outcome =
 snsClient.ListSubscriptions(
                request);

        if (outcome.IsSuccess()) {
            const Aws::Vector<Aws::SNS::Model::Subscription> &newSubscriptions =
                    outcome.GetResult().GetSubscriptions();
            subscriptions.insert(subscriptions.cend(), newSubscriptions.begin(),
                                 newSubscriptions.end());
        }
        else {
            std::cerr << "Error listing subscriptions "
```

```
                        << outcome.GetError().GetMessage()
                        <<
                        std::endl;
            result = false;
            break;
        }

        nextToken = outcome.GetResult().GetNextToken();
    } while (!nextToken.empty());

    if (result) {
        if (subscriptions.empty()) {
            std::cout << "No subscriptions found" << std::endl;
        }
        else {
            std::cout << "Subscriptions list:" << std::endl;
            for (auto const &subscription: subscriptions) {
                std::cout << "  * " << subscription.GetSubscriptionArn() <<
 std::endl;
            }
        }
    }
    return result;
}
```

- Untuk API detailnya, lihat ListSubscriptionsdi AWS SDK for C++ APIReferensi.

CLI

AWS CLI

Untuk daftar SNS langganan Anda

`list-subscriptions`Contoh berikut menampilkan daftar SNS langganan di AWS akun Anda.

```
aws sns list-subscriptions
```

Output:

```
{
```

```
    "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"
        }
    ]
}
```

- Untuk API detailnya, lihat ListSubscriptionsdi Referensi AWS CLI Perintah.

Java

SDKuntuk Java 2.x

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

```
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);
        }
    }
}
```

- Untuk API detailnya, lihat ListSubscriptionsdi AWS SDK for Java 2.x APIReferensi.

JavaScript

SDKuntuk JavaScript (v3)

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

Buat klien dalam modul terpisah dan ekspor klien tersebut.

```
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({});
```

Impor modul SDK dan klien dan panggil fileAPI.

```
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;
};
```

- Untuk informasi selengkapnya, lihat AWS SDK for JavaScript Panduan Developer.
- Untuk API detailnya, lihat ListSubscriptionsdi AWS SDK for JavaScript APIReferensi.

Kotlin

SDKuntuk Kotlin

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

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

- Untuk API detailnya, lihat ListSubscriptions AWSSDKAPIreferensi Kotlin.

PHP

SDKuntuk PHP

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

```
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());
}
```

- Untuk API detailnya, lihat ListSubscriptionsdi AWS SDK for PHP APIReferensi.

Python

SDKuntuk Python (Boto3)

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

```
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
```

- Untuk API detailnya, lihat ListSubscriptions AWSSDKReferensi Python (Boto3). API

Ruby

SDKuntuk Ruby

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

```
# 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
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for Ruby.
- Untuk API detailnya, lihat ListSubscriptionsdi AWS SDK for Ruby APIReferensi.

SAP ABAP

SDKuntuk SAP ABAP

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

```
   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.
```

- Untuk API detailnya, lihat ListSubscriptions AWSSDKuntuk SAP ABAP API referensi.

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Gunakan **`ListTopics`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`ListTopics`.

.NET

AWS SDK for .NET

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

```
   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}");
            }
        }
    }
```

- Untuk API detailnya, lihat ListTopicsdi AWS SDK for .NET APIReferensi.

C++

SDKuntuk C ++

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

```
//! Retrieve a list of Amazon Simple Notification Service (Amazon SNS) topics.
/*!
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool
AwsDoc::SNS::listTopics(const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::String nextToken; // Next token is used to handle a paginated response.
    bool result = true;
    do {
        Aws::SNS::Model::ListTopicsRequest request;
```

```
        if (!nextToken.empty()) {
            request.SetNextToken(nextToken);
        }

        const Aws::SNS::Model::ListTopicsOutcome outcome = snsClient.ListTopics(
                request);

        if (outcome.IsSuccess()) {
            std::cout << "Topics list:" << std::endl;
            for (auto const &topic: outcome.GetResult().GetTopics()) {
                std::cout << "  * " << topic.GetTopicArn() << std::endl;
            }
        }
        else {
            std::cerr << "Error listing topics " <<
 outcome.GetError().GetMessage() <<
                      std::endl;
            result = false;
            break;
        }

        nextToken = outcome.GetResult().GetNextToken();
    } while (!nextToken.empty());

    return result;
}
```

- Untuk API detailnya, lihat ListTopicsdi AWS SDK for C++ APIReferensi.

CLI

AWS CLI

Untuk daftar SNS topik Anda

`list-topics`Contoh berikut mencantumkan semua SNS topik di AWS akun Anda.

```
aws sns list-topics
```

Output:

```
{
```

```
    "Topics": [
        {
            "TopicArn": "arn:aws:sns:us-west-2:123456789012:my-topic"
        }
    ]
}
```

- Untuk API detailnya, lihat ListTopicsdi Referensi AWS CLI Perintah.

Go

SDKuntuk Go V2

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

```
package main

import (
 "context"
 "fmt"
 "log"

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

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
```

```
 fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
 fmt.Println(err)
 return
}
snsClient := sns.NewFromConfig(sdkConfig)
fmt.Println("Let's list the topics for your account.")
var topics []types.Topic
paginator := sns.NewListTopicsPaginator(snsClient, &sns.ListTopicsInput{})
for paginator.HasMorePages() {
 output, err := paginator.NextPage(ctx)
 if err != nil {
  log.Printf("Couldn't get topics. Here's why: %v\n", err)
  break
 } else {
  topics = append(topics, output.Topics...)
 }
}
if len(topics) == 0 {
 fmt.Println("You don't have any topics!")
} else {
 for _, topic := range topics {
  fmt.Printf("\t%v\n", *topic.TopicArn)
 }
}
}
```

- Untuk API detailnya, lihat ListTopicsdi AWS SDK for Go APIReferensi.

Java

SDKuntuk Java 2.x

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

```
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);
        }
    }
}
```

- Untuk API detailnya, lihat ListTopicsdi AWS SDK for Java 2.x APIReferensi.

JavaScript

SDKuntuk JavaScript (v3)

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

Buat klien dalam modul terpisah dan ekspor klien tersebut.

```
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({});
```

Impor modul SDK dan klien dan panggil fileAPI.

```
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;
};
```

- Untuk informasi selengkapnya, lihat AWS SDK for JavaScript Panduan Developer.
- Untuk API detailnya, lihat ListTopicsdi AWS SDK for JavaScript APIReferensi.

Kotlin

SDKuntuk Kotlin

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

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

- Untuk API detailnya, lihat ListTopics AWSSDKAPIreferensi Kotlin.

PHP

SDKuntuk PHP

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

```
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());
}
```

- Untuk API detailnya, lihat ListTopicsdi AWS SDK for PHP APIReferensi.

Python

SDKuntuk Python (Boto3)

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

```
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
```

- Untuk API detailnya, lihat ListTopics AWSSDKReferensi Python (Boto3). API

Ruby

SDKuntuk Ruby

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

```
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.'

  return if list_topics?(sns_client)

  puts 'The bucket was not created. Stopping program.'
  exit 1
end

# Example usage:
run_me if $PROGRAM_NAME == __FILE__
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for Ruby.
- Untuk API detailnya, lihat ListTopicsdi AWS SDK for Ruby APIReferensi.

Rust

SDKuntuk Rust

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

```
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(())
}
```

- Untuk API detailnya, lihat ListTopics AWSSDKuntuk API referensi Rust.

SAP ABAP

SDKuntuk SAP ABAP

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

```
    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.
```

- Untuk API detailnya, lihat ListTopics AWSSDKuntuk SAP ABAP API referensi.

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Gunakan **`Publish`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`Publish`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Membuat dan mempublikasikan ke FIFO topik
- Publikasikan pesan SMS teks
- Publikasikan pesan ke antrian

.NET

AWS SDK for .NET

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

Publikasikan pesan ke topik.

```
    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}");
        }
    }
```

Publikasikan pesan ke topik dengan opsi grup, duplikasi, dan atribut.

```
    /// <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);
        }
    }
```

Terapkan pilihan pengguna ke tindakan publikasi.

```
    /// <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;
    }
```

- Untuk API detailnya, lihat Publikasikan di AWS SDK for .NET APIReferensi.

C++

SDKuntuk C ++

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

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

```
                                    const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

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

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

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

    return outcome.IsSuccess();
}
```

Publikasikan pesan dengan atribut.

```
        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;
        }
```

- Untuk API detailnya, lihat Publikasikan di AWS SDK for C++ APIReferensi.

CLI

AWS CLI

Contoh 1: Untuk mempublikasikan pesan ke topik

`publish`Contoh berikut menerbitkan pesan yang ditentukan ke SNS topik yang ditentukan. Pesan berasal dari file teks, yang memungkinkan Anda untuk memasukkan jeda baris.

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

Isi dari `message.txt`:

```
Hello World
Second Line
```

Output:

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

Contoh 2: Untuk mempublikasikan SMS pesan ke nomor telepon

`publish`Contoh berikut menerbitkan pesan `Hello world!` ke nomor `+1-555-555-0100` telepon.

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

Output:

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

- Untuk API detailnya, lihat Publikasikan di Referensi AWS CLI Perintah.

Go

SDKuntuk Go V2

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

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
func (actor SnsActions) Publish(ctx context.Context, topicArn string, message
 string, groupId string, dedupId string, filterKey string, filterValue string)
 error {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
```

```
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(ctx, &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
 return err
}
```

- Untuk API detailnya, lihat Publikasikan di AWS SDK for Go APIReferensi.

Java

SDKuntuk Java 2.x

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

```
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);
        }
    }
}
```

- Untuk API detailnya, lihat Publikasikan di AWS SDK for Java 2.x APIReferensi.

JavaScript

SDKuntuk JavaScript (v3)

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

Buat klien dalam modul terpisah dan ekspor klien tersebut.

```
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({});
```

Impor modul SDK dan klien dan panggil fileAPI.

```
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;
};
```

Publikasikan pesan ke topik dengan opsi grup, duplikasi, dan atribut.

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

    let groupId;
    let deduplicationId;
    let 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();
    }
  }
```

- Untuk informasi selengkapnya, lihat AWS SDK for JavaScript Panduan Developer.
- Untuk API detailnya, lihat Publikasikan di AWS SDK for JavaScript APIReferensi.

Kotlin

SDKuntuk Kotlin

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

```
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.")
    }
}
```

- Untuk API detailnya, lihat Publish in AWS SDKuntuk API referensi Kotlin.

PHP

SDKuntuk PHP

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

```
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());
```

```
}
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for PHP.
- Untuk API detailnya, lihat Publikasikan di AWS SDK for PHP APIReferensi.

PowerShell

Alat untuk PowerShell

Contoh 1: Contoh ini menunjukkan penerbitan pesan dengan satu baris yang MessageAttribute dideklarasikan.

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

Contoh 2: Contoh ini menunjukkan penerbitan pesan dengan beberapa MessageAttributes dideklarasikan sebelumnya.

```
$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
```

- Untuk API detailnya, lihat Menerbitkan di AWS Tools for PowerShell Referensi Cmdlet.

Python

SDKuntuk Python (Boto3)

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

Publikasikan pesan dengan atribut sehingga langganan dapat memfilter berdasarkan atribut.

```
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
```

Publikasikan pesan yang mengambil bentuk berbeda berdasarkan protokol pelanggan.

```
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
```

- Untuk API detailnya, lihat Publikasikan AWS SDKuntuk Referensi Python (Boto3). API

Ruby

SDKuntuk Ruby

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

```
# 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
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for Ruby.
- Untuk API detailnya, lihat Publikasikan di AWS SDK for Ruby APIReferensi.

Rust

SDKuntuk Rust

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

```
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(())
}
```

- Untuk API detailnya, lihat Publikasikan AWS SDKuntuk API referensi Rust.

SAP ABAP

SDKuntuk SAP ABAP

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

```
    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.
```

- Untuk API detailnya, lihat Publikasikan AWS SDKuntuk SAP ABAP API referensi.

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Gunakan **`SetSMSAttributes`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`SetSMSAttributes`.

C++

SDKuntuk C ++

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

Cara menggunakan Amazon SNS untuk mengatur efaultSMSType atribut D.

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

- Untuk API detailnya, lihat S etSMSAttributes di AWS SDK for C++ APIReferensi.

CLI

AWS CLI

Untuk mengatur atribut SMS pesan

`set-sms-attributes`Contoh berikut menetapkan ID pengirim default untuk SMS pesan ke`MyName`.

```
aws sns set-sms-attributes \
    --attributes DefaultSenderID=MyName
```

Perintah ini tidak menghasilkan output.

- Untuk API detailnya, lihat S etSMSAttributes di Referensi AWS CLI Perintah.

Java

SDKuntuk Java 2.x

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

```
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);
        }
    }
}
```

- Untuk API detailnya, lihat S etSMSAttributes di AWS SDK for Java 2.x APIReferensi.

JavaScript

SDKuntuk JavaScript (v3)

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

Buat klien dalam modul terpisah dan ekspor klien tersebut.

```
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({});
```

Impor modul SDK dan klien dan panggil fileAPI.

```
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;
};
```

- Untuk informasi selengkapnya, lihat AWS SDK for JavaScript Panduan Developer.
- Untuk API detailnya, lihat S etSMSAttributes di AWS SDK for JavaScript APIReferensi.

PHP

SDKuntuk PHP

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

```
$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());
}
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for PHP.
- Untuk API detailnya, lihat S etSMSAttributes di AWS SDK for PHP APIReferensi.

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Gunakan **`SetSubscriptionAttributes`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`SetSubscriptionAttributes`.

CLI

AWS CLI

Untuk mengatur atribut langganan

`set-subscription-attributes`Contoh berikut menetapkan `RawMessageDelivery` atribut ke SQS langganan.

```
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
```

Perintah ini tidak menghasilkan output.

`set-subscription-attributes`Contoh berikut menetapkan `FilterPolicy` atribut ke SQS langganan.

```
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\"] }"
```

Perintah ini tidak menghasilkan output.

`set-subscription-attributes`Contoh berikut menghapus `FilterPolicy` atribut dari SQS langganan.

```
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 "{}"
```

Perintah ini tidak menghasilkan output.

- Untuk API detailnya, lihat SetSubscriptionAttributesdi Referensi AWS CLI Perintah.

Java

SDKuntuk Java 2.x

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

```
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);
        }
    }
}
```

- Untuk API detailnya, lihat SetSubscriptionAttributesdi AWS SDK for Java 2.x APIReferensi.

Python

SDKuntuk Python (Boto3)

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

```
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
```

- Untuk API detailnya, lihat SetSubscriptionAttributes AWSSDKReferensi Python (Boto3). API

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Gunakan **`SetSubscriptionAttributesRedrivePolicy`** dengan AWS SDK

Contoh kode berikut menunjukkan cara menggunakan`SetSubscriptionAttributesRedrivePolicy`.

Java

SDKuntuk Java 1.x

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

```
// 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);
```

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Gunakan **`SetTopicAttributes`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`SetTopicAttributes`.

CLI

AWS CLI

Untuk menetapkan atribut untuk topik

`set-topic-attributes`Contoh berikut menetapkan `DisplayName` atribut untuk topik yang ditentukan.

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name DisplayName \
    --attribute-value MyTopicDisplayName
```

Perintah ini tidak menghasilkan output.

- Untuk API detailnya, lihat SetTopicAttributesdi Referensi AWS CLI Perintah.

Java

SDKuntuk Java 2.x

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

```
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);
        }
    }
}
```

- Untuk API detailnya, lihat SetTopicAttributesdi AWS SDK for Java 2.x APIReferensi.

JavaScript

SDKuntuk JavaScript (v3)

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

Buat klien dalam modul terpisah dan ekspor klien tersebut.

```
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({});
```

Impor modul SDK dan klien dan panggil fileAPI.

```
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;
};
```

- Untuk informasi selengkapnya, lihat AWS SDK for JavaScript Panduan Developer.
- Untuk API detailnya, lihat SetTopicAttributesdi AWS SDK for JavaScript APIReferensi.

Kotlin

SDKuntuk Kotlin

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

```
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.")
    }
}
```

- Untuk API detailnya, lihat SetTopicAttributes AWSSDKAPIreferensi Kotlin.

PHP

SDKuntuk PHP

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

```
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());
}
```

- Untuk API detailnya, lihat SetTopicAttributesdi AWS SDK for PHP APIReferensi.

Ruby

SDKuntuk Ruby

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

```
# 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
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for Ruby.
- Untuk API detailnya, lihat SetTopicAttributesdi AWS SDK for Ruby APIReferensi.

SAP ABAP

SDKuntuk SAP ABAP

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

```
    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.
```

- Untuk API detailnya, lihat SetTopicAttributes AWSSDKuntuk SAP ABAP API referensi.

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Gunakan **`Subscribe`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`Subscribe`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Membuat dan mempublikasikan ke FIFO topik
- Publikasikan pesan ke antrian

.NET

AWS SDK for .NET

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

Berlangganan alamat email ke suatu topik.

```
        /// <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;
        }
```

Berlangganan antrian ke topik dengan filter opsional.

```
    /// <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;
    }
```

- Untuk API detailnya, lihat Berlangganan di AWS SDK for .NET APIReferensi.

C++

SDKuntuk C ++

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

Berlangganan alamat email ke suatu topik.

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

Berlangganan aplikasi seluler ke suatu topik.

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

Berlangganan fungsi Lambda ke suatu topik.

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

Berlangganan SQS antrian ke suatu topik.

```
        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;
            }
```

Berlangganan dengan filter ke topik.

```
       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;
}
```

- Untuk API detailnya, lihat Berlangganan di AWS SDK for C++ APIReferensi.

CLI

AWS CLI

Untuk berlangganan topik

`subscribe`Perintah berikut berlangganan alamat email ke topik yang ditentukan.

```
aws sns subscribe \
    --topic-arn arn:aws:sns:us-west-2:123456789012:my-topic \
    --protocol email \
```

```
    --notification-endpoint my-email@example.com
```

Output:

```
{
    "SubscriptionArn": "pending confirmation"
}
```

- Untuk API detailnya, lihat Berlangganan di Referensi AWS CLI Perintah.

Go

SDKuntuk Go V2

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

Berlangganan antrian ke topik dengan filter opsional.

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(ctx context.Context, topicArn string,
 queueArn string, filterMap map[string][]string) (string, error) {
 var subscriptionArn string
```

```
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(ctx, &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}
```

- Untuk API detailnya, lihat Berlangganan di AWS SDK for Go APIReferensi.

Java

SDKuntuk Java 2.x

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

Berlangganan alamat email ke suatu topik.

```
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);
        }
    }
}
```

Berlangganan HTTP titik akhir ke suatu topik.

```
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);
        }
    }
}
```

Berlangganan fungsi Lambda ke suatu topik.

```
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 "";
    }
}
```

- Untuk API detailnya, lihat Berlangganan di AWS SDK for Java 2.x APIReferensi.

JavaScript

SDKuntuk JavaScript (v3)

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

Buat klien dalam modul terpisah dan ekspor klien tersebut.

```
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({});
```

Impor modul SDK dan klien dan panggil fileAPI.

```
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'
  // }
};
```

Berlangganan aplikasi seluler ke suatu topik.

```
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;
};
```

Berlangganan fungsi Lambda ke suatu topik.

```
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;
};
```

Berlangganan SQS antrian ke suatu topik.

```
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;
};
```

Berlangganan dengan filter ke topik.

```
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;
};
```

- Untuk informasi selengkapnya, lihat AWS SDK for JavaScript Panduan Developer.
- Untuk API detailnya, lihat Berlangganan di AWS SDK for JavaScript APIReferensi.

Kotlin

SDKuntuk Kotlin

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

Berlangganan alamat email ke suatu topik.

```
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()
    }
}
```

Berlangganan fungsi Lambda ke suatu topik.

```
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}")
    }
}
```

- Untuk API detailnya, lihat Berlangganan AWS SDKuntuk API referensi Kotlin.

PHP

SDKuntuk PHP

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

Berlangganan alamat email ke suatu topik.

```
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());
}
```

Berlangganan HTTP titik akhir ke suatu topik.

```
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());
}
```

- Untuk API detailnya, lihat Berlangganan di AWS SDK for PHP APIReferensi.

Python

SDKuntuk Python (Boto3)

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

Berlangganan alamat email ke suatu topik.

```
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
```

- Untuk API detailnya, lihat Berlangganan AWS SDKuntuk Referensi Python (Boto3). API

Ruby

SDKuntuk Ruby

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

Berlangganan alamat email ke suatu topik.

```
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
```

- Untuk informasi selengkapnya, lihat AWS SDK for Ruby Panduan Developer.
- Untuk API detailnya, lihat Berlangganan di AWS SDK for Ruby APIReferensi.

Rust

SDKuntuk Rust

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

Berlangganan alamat email ke suatu topik.

```
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(())
}
```

- Untuk API detailnya, lihat Berlangganan AWS SDKuntuk API referensi Rust.

SAP ABAP

SDKuntuk SAP ABAP

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

Berlangganan alamat email ke suatu topik.

```
   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.
```

- Untuk API detailnya, lihat Berlangganan AWS SDKuntuk SAP ABAP API referensi.

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Gunakan `TagResource` dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`TagResource`.

CLI

AWS CLI

Untuk menambahkan tag ke topik

`tag-resource`Contoh berikut menambahkan tag metadata ke topik Amazon SNS yang ditentukan.

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

Perintah ini tidak menghasilkan output.

- Untuk API detailnya, lihat TagResourcedi Referensi AWS CLI Perintah.

Java

SDKuntuk Java 2.x

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

```
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);
        }
    }
}
```

- Untuk API detailnya, lihat TagResourcedi AWS SDK for Java 2.x APIReferensi.

Kotlin

SDKuntuk Kotlin

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

```
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")
    }
}
```

- Untuk API detailnya, lihat TagResource AWSSDKAPIreferensi Kotlin.

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Gunakan **`Unsubscribe`** dengan AWS SDK atau CLI

Contoh kode berikut menunjukkan cara menggunakan`Unsubscribe`.

Contoh tindakan adalah kutipan kode dari program yang lebih besar dan harus dijalankan dalam konteks. Anda dapat melihat tindakan ini dalam konteks dalam contoh kode berikut:

- Publikasikan pesan ke antrian

.NET

AWS SDK for .NET

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

Berhenti berlangganan dari topik dengan berlangganan ARN.

```
/// <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;
}
```

- Untuk API detailnya, lihat Berhenti berlangganan di AWS SDK for .NET API Referensi.

C++

SDK untuk C ++

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

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

    Aws::SNS::Model::UnsubscribeRequest request;
    request.SetSubscriptionArn(subscriptionARN);

    const Aws::SNS::Model::UnsubscribeOutcome outcome =
 snsClient.Unsubscribe(request);

    if (outcome.IsSuccess()) {
        std::cout << "Unsubscribed successfully " << std::endl;
    }
    else {
        std::cerr << "Error while unsubscribing " <<
 outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- Untuk API detailnya, lihat Berhenti berlangganan di AWS SDK for C++ APIReferensi.

CLI

AWS CLI

Untuk berhenti berlangganan dari suatu topik

`unsubscribe`Contoh berikut menghapus langganan yang ditentukan dari suatu topik.

```
aws sns unsubscribe \
    --subscription-arn arn:aws:sns:us-west-2:0123456789012:my-
topic:8a21d249-4329-4871-acc6-7be709c6ea7f
```

Perintah ini tidak menghasilkan output.

- Untuk API detailnya, lihat Berhenti berlangganan di Referensi AWS CLI Perintah.

Java

SDKuntuk Java 2.x

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

```
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);
        }
    }
}
```

- Untuk API detailnya, lihat Berhenti berlangganan di AWS SDK for Java 2.x APIReferensi.

JavaScript

SDKuntuk JavaScript (v3)

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

Buat klien dalam modul terpisah dan ekspor klien tersebut.

```
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({});
```

Impor modul SDK dan klien dan panggil fileAPI.

```
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;
};
```

- Untuk informasi selengkapnya, lihat AWS SDK for JavaScript Panduan Developer.
- Untuk API detailnya, lihat Berhenti berlangganan di AWS SDK for JavaScript APIReferensi.

Kotlin

SDKuntuk Kotlin

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

```
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}")
    }
}
```

- Untuk API detailnya, lihat Berhenti berlangganan AWS SDKuntuk referensi Kotlin API.

PHP

SDKuntuk PHP

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

```
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());
}
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for PHP.
- Untuk API detailnya, lihat Berhenti berlangganan di AWS SDK for PHP APIReferensi.

Python

SDKuntuk Python (Boto3)

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

```
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
```

- Untuk API detailnya, lihat Berhenti berlangganan AWS SDKuntuk Referensi Python (APIBoto3).

SAP ABAP

SDKuntuk SAP ABAP

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

```
    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.
```

- Untuk API detailnya, lihat Berhenti berlangganan AWS SDKuntuk SAP ABAP API referensi.

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Skenario untuk Amazon SNS menggunakan AWS SDKs

Contoh kode berikut menunjukkan cara menerapkan skenario umum di Amazon SNS dengan AWS SDKs. Skenario ini menunjukkan kepada Anda cara menyelesaikan tugas tertentu dengan memanggil beberapa fungsi di Amazon SNS atau digabungkan dengan yang lain Layanan AWS. Setiap skenario menyertakan tautan ke kode sumber lengkap, di mana Anda dapat menemukan instruksi tentang cara mengatur dan menjalankan kode.

Skenario menargetkan tingkat pengalaman menengah untuk membantu Anda memahami tindakan layanan dalam konteks.

Contoh

- Membangun aplikasi untuk mengirimkan data ke tabel DynamoDB
- Membangun aplikasi terbitkan dan berlangganan yang menerjemahkan pesan
- Buat titik akhir platform untuk notifikasi SNS push Amazon menggunakan AWS SDK
- Membuat aplikasi manajemen aset foto yang memungkinkan pengguna mengelola foto menggunakan label
- Membuat aplikasi penjelajah Amazon Textract
- Membuat dan mempublikasikan ke SNS topik FIFO Amazon menggunakan AWS SDK
- Mendeteksi orang dan objek dalam video dengan Amazon Rekognition menggunakan AWS SDK
- Mempublikasikan SMS pesan ke SNS topik Amazon menggunakan AWS SDK
- Publikasikan pesan besar ke Amazon SNS dengan Amazon S3 menggunakan AWS SDK
- Menerbitkan pesan SNS SMS teks Amazon menggunakan AWS SDK
- Publikasikan SNS pesan Amazon ke SQS antrian Amazon menggunakan AWS SDK
- Gunakan API Gateway untuk menjalankan fungsi Lambda
- Menggunakan peristiwa terjadwal untuk menginvokasi fungsi Lambda

Membangun aplikasi untuk mengirimkan data ke tabel DynamoDB

Contoh kode berikut menunjukkan cara membangun aplikasi yang mengirimkan data ke tabel Amazon DynamoDB dan memberi tahu Anda saat pengguna memperbarui tabel.

Java

SDKuntuk Java 2.x

Menunjukkan cara membuat aplikasi web dinamis yang mengirimkan data menggunakan Amazon API DynamoDB Java dan mengirim pesan teks menggunakan Amazon Simple Notification Service Java. API

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

Layanan yang digunakan dalam contoh ini

- DynamoDB
- Amazon SNS

JavaScript

SDKuntuk JavaScript (v3)

Contoh ini menunjukkan cara membuat aplikasi yang memungkinkan pengguna mengirimkan data ke tabel Amazon DynamoDB, dan mengirim pesan teks ke administrator menggunakan Amazon Simple Notification Service (Amazon). SNS

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

Contoh ini juga tersedia di panduan developer v3 AWS SDK for JavaScript.

Layanan yang digunakan dalam contoh ini

- DynamoDB
- Amazon SNS

Kotlin

SDKuntuk Kotlin

Menunjukkan cara membuat aplikasi Android asli yang mengirimkan data menggunakan Amazon API DynamoDB Kotlin dan mengirim pesan teks menggunakan Amazon Kotlin. SNS API

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

Layanan yang digunakan dalam contoh ini

- DynamoDB
- Amazon SNS

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Membangun aplikasi terbitkan dan berlangganan yang menerjemahkan pesan

Contoh kode berikut menunjukkan cara membuat aplikasi yang memiliki langganan dan mempublikasikan fungsionalitas dan menerjemahkan pesan.

.NET

AWS SDK for .NET

Menunjukkan cara menggunakan Amazon Simple Notification Service. NETAPIuntuk membuat aplikasi web yang memiliki fungsi berlangganan dan mempublikasikan. Selain itu, contoh aplikasi ini juga menerjemahkan pesan.

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

Layanan yang digunakan dalam contoh ini

- Amazon SNS
- Amazon Translate

Java

SDKuntuk Java 2.x

Menunjukkan cara menggunakan Amazon Simple Notification Service Java API untuk membuat aplikasi web yang memiliki fungsi berlangganan dan mempublikasikan. Selain itu, contoh aplikasi ini juga menerjemahkan pesan.

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

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

Layanan yang digunakan dalam contoh ini

- Amazon SNS
- Amazon Translate

Kotlin

SDKuntuk Kotlin

Menunjukkan cara menggunakan Amazon SNS Kotlin API untuk membuat aplikasi yang memiliki fungsionalitas langganan dan publikasi. Selain itu, contoh aplikasi ini juga menerjemahkan pesan.

Untuk kode sumber lengkap dan petunjuk tentang cara membuat aplikasi web, lihat contoh lengkapnya di GitHub.

Untuk kode sumber lengkap dan petunjuk tentang cara membuat aplikasi Android asli, lihat contoh selengkapnya di GitHub.

Layanan yang digunakan dalam contoh ini

- Amazon SNS
- Amazon Translate

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Buat titik akhir platform untuk notifikasi SNS push Amazon menggunakan AWS SDK

Contoh kode berikut menunjukkan cara membuat titik akhir platform untuk notifikasi SNS push Amazon.

CLI

AWS CLI

Untuk membuat endpoint aplikasi platform

`create-platform-endpoint`Contoh berikut membuat titik akhir untuk aplikasi platform tertentu menggunakan token yang ditentukan.

```
aws sns create-platform-endpoint \
    --platform-application-arn arn:aws:sns:us-west-2:123456789012:app/GCM/
MyApplication \
    --token EXAMPLE12345...
```

Output:

```
{
      "EndpointArn": "arn:aws:sns:us-west-2:1234567890:endpoint/GCM/
MyApplication/12345678-abcd-9012-efgh-345678901234"
}
```

Java

SDKuntuk Java 2.x

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

```
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 device token or registration ID of the mobile device.
 This is a unique
               identifier provided by the device platform (e.g., Apple Push
 Notification Service (APNS) for iOS devices, Firebase Cloud Messaging (FCM)
               for Android devices) when the mobile app is registered to receive
 push notifications.

               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);
            return;
        }

        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());
        }
    }
}
```

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Membuat aplikasi manajemen aset foto yang memungkinkan pengguna mengelola foto menggunakan label

Contoh kode berikut ini menunjukkan cara membuat aplikasi nirserver yang memungkinkan pengguna mengelola foto menggunakan label.

.NET

AWS SDK for .NET

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

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

Untuk mendalami tentang asal usul contoh ini, lihat postingan di Komunitas AWS.

Layanan yang digunakan dalam contoh ini

- APIGerbang
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

C++

SDKuntuk C ++

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

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

Untuk mendalami tentang asal usul contoh ini, lihat postingan di Komunitas AWS.

Layanan yang digunakan dalam contoh ini

- APIGerbang
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Java

SDKuntuk Java 2.x

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

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

Untuk mendalami tentang asal usul contoh ini, lihat postingan di Komunitas AWS.

Layanan yang digunakan dalam contoh ini

- APIGerbang
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

JavaScript

SDKuntuk JavaScript (v3)

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

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

Untuk mendalami tentang asal usul contoh ini, lihat postingan di Komunitas AWS.

Layanan yang digunakan dalam contoh ini

- APIGerbang
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Kotlin

SDKuntuk Kotlin

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

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

Untuk mendalami tentang asal usul contoh ini, lihat postingan di Komunitas AWS.

Layanan yang digunakan dalam contoh ini

- APIGerbang
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

PHP

SDKuntuk PHP

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

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

Untuk mendalami tentang asal usul contoh ini, lihat postingan di Komunitas AWS.

Layanan yang digunakan dalam contoh ini

- APIGerbang
- DynamoDB

- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Rust

SDKuntuk Rust

Menunjukkan cara mengembangkan aplikasi manajemen aset foto yang mendeteksi label dalam gambar menggunakan Amazon Rekognition dan menyimpannya untuk pengambilan nanti.

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

Untuk mendalami tentang asal usul contoh ini, lihat postingan di Komunitas AWS.

Layanan yang digunakan dalam contoh ini

- APIGerbang
- DynamoDB
- Lambda
- Amazon Rekognition
- Amazon S3
- Amazon SNS

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Membuat aplikasi penjelajah Amazon Textract

Contoh kode berikut ini menunjukkan cara menjelajahi output Amazon Textract melalui aplikasi interaktif.

JavaScript

SDKuntuk JavaScript (v3)

Menunjukkan cara menggunakan aplikasi AWS SDK for JavaScript untuk membangun aplikasi React yang menggunakan Amazon Textract untuk mengekstrak data dari gambar dokumen dan menampilkannya di halaman web interaktif. Contoh ini berjalan di peramban web dan memerlukan identitas Amazon Cognito yang diautentikasi sebagai kredensialnya. Ini menggunakan Amazon Simple Storage Service (Amazon S3) Simple Storage Service (Amazon S3) untuk penyimpanan, dan untuk notifikasi, ia melakukan polling antrian Amazon Simple Queue Service (SQSAmazon) yang berlangganan topik Amazon Simple Notification Service (Amazon). SNS

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

Layanan yang digunakan dalam contoh ini

- Identitas Amazon Cognito
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

SDKuntuk Python (Boto3)

Menunjukkan cara menggunakan Amazon Textract untuk mendeteksi elemen teks, formulir, dan tabel dalam gambar dokumen. AWS SDK for Python (Boto3) Gambar input dan output Amazon Textract ditampilkan dalam aplikasi Tkinter yang memungkinkan Anda menjelajahi elemen yang terdeteksi.

- Kirim gambar dokumen ke Amazon Textract dan jelajahi output elemen yang terdeteksi.
- Kirim gambar langsung ke Amazon Textract atau melalui bucket Amazon Simple Storage Service (Amazon S3).
- Gunakan asinkron APIs untuk memulai pekerjaan yang menerbitkan pemberitahuan ke topik Amazon Simple Notification Service (AmazonSNS) saat pekerjaan selesai.

- Polling antrian Amazon Simple Queue Service (AmazonSQS) untuk pesan penyelesaian pekerjaan dan tampilkan hasilnya.

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

Layanan yang digunakan dalam contoh ini

- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Membuat dan mempublikasikan ke SNS topik FIFO Amazon menggunakan AWS SDK

Contoh kode berikut menunjukkan cara membuat dan mempublikasikan ke SNS topik FIFO Amazon.

Java

SDKuntuk Java 2.x

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

Contoh ini

- membuat SNS FIFO topik Amazon, dua SQS FIFO antrian Amazon, dan satu antrian Standar.
- berlangganan antrian ke topik dan menerbitkan pesan ke topik tersebut.

Tes memverifikasi penerimaan pesan ke setiap antrian. Contoh lengkap juga menunjukkan penambahan kebijakan akses dan menghapus sumber daya di akhir.

```
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);
        }
    }
```

- Untuk API detailnya, lihat topik berikut di AWS SDK for Java 2.x APIReferensi.
  - CreateTopic
  - Publikasikan
  - Berlangganan

Python

SDKuntuk Python (Boto3)

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

Buat SNS FIFO topik Amazon, berlangganan Amazon SQS FIFO dan antrian standar ke topik, dan publikasikan pesan ke topik tersebut.

```
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
```

- Untuk API detailnya, lihat topik berikut AWS SDKuntuk Referensi Python (Boto3). API
  - CreateTopic
  - Publikasikan
  - Berlangganan

SAP ABAP

SDKuntuk SAP ABAP

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

Buat FIFO topik, berlangganan SQS FIFO antrian Amazon ke topik, dan publikasikan pesan ke SNS topik Amazon.

```
    " 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.
```

- Untuk API detailnya, lihat topik berikut AWS SDKuntuk SAP ABAP API referensi.
  - CreateTopic
  - Publikasikan
  - Berlangganan

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Mendeteksi orang dan objek dalam video dengan Amazon Rekognition menggunakan AWS SDK

Contoh kode berikut menunjukkan cara mendeteksi orang dan objek dalam video dengan Amazon Rekognition.

Python

SDKuntuk Python (Boto3)

Gunakan Amazon Rekognition untuk mendeteksi wajah, objek, dan orang dalam video dengan memulai tugas deteksi asinkron. Contoh ini juga mengonfigurasi Amazon Rekognition untuk memberi tahu topik Amazon Simple Notification Service (SNSAmazon) saat pekerjaan selesai dan berlangganan antrian Amazon Simple Queue Service (SQSAmazon) ke topik tersebut. Ketika antrian menerima pesan tentang pekerjaan, pekerjaan diambil dan hasilnya adalah output.

Contoh ini paling baik dilihat di GitHub. Untuk kode sumber lengkap dan instruksi tentang cara mengatur dan menjalankan, lihat contoh lengkapnya di GitHub.

Layanan yang digunakan dalam contoh ini

- Amazon Rekognition
- Amazon SNS
- Amazon SQS

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Mempublikasikan SMS pesan ke SNS topik Amazon menggunakan AWS SDK

Contoh kode berikut ini menunjukkan cara:

- Buat SNS topik Amazon.
- Berlangganan nomor telepon ke topik.
- Publikasikan SMS pesan ke topik sehingga semua nomor telepon berlangganan menerima pesan sekaligus.

Java

SDKuntuk Java 2.x

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

Buat topik dan kembalikanARN.

```
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 "";
    }
}
```

Berlangganan titik akhir ke suatu topik.

```
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);
        }
    }
}
```

Tetapkan atribut pada pesan, seperti ID pengirim, harga maksimum, dan jenisnya. Atribut pesan bersifat opsional.

```
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);
        }
    }
}
```

Publikasikan pesan ke topik. Pesan dikirim ke setiap pelanggan.

```
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);
        }
    }
}
```

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Publikasikan pesan besar ke Amazon SNS dengan Amazon S3 menggunakan AWS SDK

Contoh kode berikut menunjukkan cara mempublikasikan pesan besar ke Amazon SNS menggunakan Amazon S3 untuk menyimpan muatan pesan.

Java

SDKuntuk Java 1.x

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

Untuk mempublikasikan pesan besar, gunakan Amazon SNS Extended Client Library for Java. Pesan yang Anda kirim mereferensikan objek Amazon S3 yang berisi konten pesan yang sebenarnya.

```
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());
        }
}
```

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Menerbitkan pesan SNS SMS teks Amazon menggunakan AWS SDK

Contoh kode berikut menunjukkan cara mempublikasikan SMS pesan menggunakan AmazonSNS.

.NET

AWS SDK for .NET

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

```
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}");
            }
        }
    }
}
```

- Untuk API detailnya, lihat Publikasikan di AWS SDK for .NET APIReferensi.

C++

SDKuntuk C ++

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

```
/**
 * Publish SMS: use Amazon Simple Notification Service (Amazon SNS) to send an
 SMS text message to a phone number.
 * Note: This requires additional AWS configuration prior to running example.
 *
 *  NOTE: When you start using Amazon SNS to send SMS messages, your AWS account
 is in the SMS sandbox and you can only
 *  use verified destination phone numbers. See https://docs.aws.amazon.com/sns/
latest/dg/sns-sms-sandbox.html.
 *  NOTE: If destination is in the US, you also have an additional restriction
 that you have use a dedicated
 *  origination ID (phone number). You can request an origination number using
 Amazon Pinpoint for a fee.
 *  See https://aws.amazon.com/blogs/compute/provisioning-and-using-10dlc-
origination-numbers-with-amazon-sns/
 *  for more information.
 *
 *  <phone_number_value> input parameter uses E.164 format.
 *  For example, in United States, this input value should be of the form:
 +12223334444
 */

//! Send an SMS text message to a phone number.
/*!
  \param message: The message to publish.
  \param phoneNumber: The phone number of the recipient in E.164 format.
  \param clientConfiguration: AWS client configuration.
  \return bool: Function succeeded.
 */
bool AwsDoc::SNS::publishSms(const Aws::String &message,
                             const Aws::String &phoneNumber,
```

```
                                const Aws::Client::ClientConfiguration
 &clientConfiguration) {
    Aws::SNS::SNSClient snsClient(clientConfiguration);

    Aws::SNS::Model::PublishRequest request;
    request.SetMessage(message);
    request.SetPhoneNumber(phoneNumber);

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

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

    return outcome.IsSuccess();
}
```

- Untuk API detailnya, lihat Publikasikan di AWS SDK for C++ APIReferensi.

Java

SDKuntuk Java 2.x

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

```
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);
        }
    }
}
```

- Untuk API detailnya, lihat Publikasikan di AWS SDK for Java 2.x APIReferensi.

Kotlin

SDKuntuk Kotlin

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

```
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.")
    }
}
```

- Untuk API detailnya, lihat Publish in AWS SDKuntuk API referensi Kotlin.

PHP

SDKuntuk PHP

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

```
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());
}
```

- Untuk informasi selengkapnya, silakan lihat Panduan Developer AWS SDK for PHP.
- Untuk API detailnya, lihat Publikasikan di AWS SDK for PHP APIReferensi.

Python

SDKuntuk Python (Boto3)

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

```
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
```

- Untuk API detailnya, lihat Publikasikan AWS SDKuntuk Referensi Python (Boto3). API

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Publikasikan SNS pesan Amazon ke SQS antrian Amazon menggunakan AWS SDK

Contoh kode berikut ini menunjukkan cara:

- Buat topik (FIFOatau non-FIFO).
- Berlangganan beberapa antrian ke topik dengan opsi untuk menerapkan filter.
- Publikasikan pesan ke topik.
- Polling antrian untuk pesan yang diterima.

.NET

AWS SDK for .NET

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

Jalankan skenario interaktif di penggugah/prompt perintah.

```
/// <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;
    }
}
```

Buat kelas yang membungkus SQS operasi Amazon.

```
/// <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;
```

```
    }
}
```

Buat kelas yang membungkus SNS operasi Amazon.

```
/// <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;
    }
}
```

- Untuk API detailnya, lihat topik berikut di AWS SDK for .NET APIReferensi.

- CreateQueue
- CreateTopic
- DeleteMessageBatch
- DeleteQueue
- DeleteTopic
- GetQueueAttributes
- Publikasikan
- ReceiveMessage
- SetQueueAttributes
- Berlangganan
- Berhenti berlangganan

C++

SDKuntuk C ++

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

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

//! Workflow for messaging with topics and queues using Amazon SNS and Amazon
 SQS.
/*!
 \param clientConfig Aws client configuration.
 \return bool: Successful completion.
 */
bool AwsDoc::TopicsAndQueues::messagingWithTopicsAndQueues(
        const Aws::Client::ClientConfiguration &clientConfiguration) {
    std::cout << "Welcome to messaging with topics and queues." << std::endl;
    printAsterisksLine();
    std::cout << "In this workflow, you will create an SNS topic and subscribe "
```

```
              << NUMBER_OF_QUEUES <<
              " SQS queues to the topic." << std::endl;
   std::cout
           << "You can select from several options for configuring the topic and
the subscriptions for the "
           << NUMBER_OF_QUEUES << " queues." << std::endl;
   std::cout << "You can then post to the topic and see the results in the
queues."
             << std::endl;

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

   printAsterisksLine();

   std::cout << "SNS topics can be configured as FIFO (First-In-First-Out)."
             << std::endl;
   std::cout
           << "FIFO topics deliver messages in order and support deduplication
and message filtering."
           << std::endl;
   bool isFifoTopic = askYesNoQuestion(
           "Would you like to work with FIFO topics? (y/n) ");

   bool contentBasedDeduplication = false;
   Aws::String topicName;
   if (isFifoTopic) {
       printAsterisksLine();
       std::cout << "Because you have chosen a FIFO topic, deduplication is
supported."
                 << std::endl;
       std::cout
               << "Deduplication IDs are either set in the message or
automatically generated "
               << "from content using a hash function." << std::endl;
       std::cout
               << "If a message is successfully published to an SNS FIFO topic,
any message "
               << "published and determined to have the same deduplication ID, "
               << std::endl;
       std::cout
               << "within the five-minute deduplication interval, is accepted
but not delivered."
               << std::endl;
       std::cout
```

```
                << "For more information about deduplication, "
                << "see https://docs.aws.amazon.com/sns/latest/dg/fifo-message-
dedup.html."
                << std::endl;
        contentBasedDeduplication = askYesNoQuestion(
                "Use content-based deduplication instead of entering a
 deduplication ID? (y/n) ");
    }

    printAsterisksLine();

    Aws::SQS::SQSClient sqsClient(clientConfiguration);
    Aws::Vector<Aws::String> queueURLS;
    Aws::Vector<Aws::String> subscriptionARNS;

    Aws::String topicARN;
    {
        topicName = askQuestion("Enter a name for your SNS topic. ");

        // 1.  Create an Amazon SNS topic, either FIFO or non-FIFO.
        Aws::SNS::Model::CreateTopicRequest request;

        if (isFifoTopic) {
            request.AddAttributes("FifoTopic", "true");
            if (contentBasedDeduplication) {
                request.AddAttributes("ContentBasedDeduplication", "true");
            }
            topicName = topicName + FIFO_SUFFIX;

            std::cout
                    << "Because you have selected a FIFO topic, '.fifo' must be
 appended to the topic name."
                    << std::endl;
        }

        request.SetName(topicName);

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

        if (outcome.IsSuccess()) {
            topicARN = outcome.GetResult().GetTopicArn();
            std::cout << "Your new topic with the name '" << topicName
```

```
                    << "' and the topic Amazon Resource Name (ARN) " <<
std::endl;
            std::cout << "'" << topicARN << "' has been created." << std::endl;

        }
        else {
            std::cerr << "Error with TopicsAndQueues::CreateTopic. "
                      << outcome.GetError().GetMessage()
                      << std::endl;

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

            return false;
        }
    }

    printAsterisksLine();

    std::cout << "Now you will create " << NUMBER_OF_QUEUES
              << " SQS queues to subscribe to the topic." << std::endl;
    Aws::Vector<Aws::String> queueNames;
    bool filteringMessages = false;
    bool first = true;
    for (int i = 1; i <= NUMBER_OF_QUEUES; ++i) {
        Aws::String queueURL;
        Aws::String queueName;
        {
            printAsterisksLine();
            std::ostringstream 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();
}
```

- Untuk API detailnya, lihat topik berikut di AWS SDK for C++ APIReferensi.
  - CreateQueue
  - CreateTopic

- DeleteMessageBatch
- DeleteQueue
- DeleteTopic
- GetQueueAttributes
- Publikasikan
- ReceiveMessage
- SetQueueAttributes
- Berlangganan
- Berhenti berlangganan

Go

SDKuntuk Go V2

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

Jalankan skenario interaktif di penggugah/prompt perintah.

```
const FIFO_SUFFIX = ".fifo"
const TONE_KEY = "tone"

var ToneChoices = []string{"cheerful", "funny", "serious", "sincere"}

// MessageBody is used to deserialize the body of a message from a JSON string.
type MessageBody struct {
 Message string
}

// ScenarioRunner separates the steps of this scenario into individual functions
 so that
// they are simpler to read and understand.
type ScenarioRunner struct {
 questioner demotools.IQuestioner
```

```
 snsActor   *actions.SnsActions
 sqsActor   *actions.SqsActions
}

func (runner ScenarioRunner) CreateTopic(ctx context.Context) (string, string,
 bool, bool) {
 log.Println("SNS topics can be configured as FIFO (First-In-First-Out) or
 standard.\n" +
  "FIFO topics deliver messages in order and support deduplication and message
 filtering.")
 isFifoTopic := runner.questioner.AskBool("\nWould you like to work with FIFO
 topics? (y/n) ", "y")

 contentBasedDeduplication := false
 if isFifoTopic {
  log.Println(strings.Repeat("-", 88))
  log.Println("Because you have chosen a FIFO topic, deduplication is supported.
\n" +
   "Deduplication IDs are either set in the message or are automatically
 generated\n" +
   "from content using a hash function. If a message is successfully published to
\n" +
   "an SNS FIFO topic, any message published and determined to have the same\n" +
   "deduplication ID, within the five-minute deduplication interval, is accepted
\n" +
   "but not delivered. For more information about deduplication, see:\n" +
   "\thttps://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.")
  contentBasedDeduplication = runner.questioner.AskBool(
   "\nDo you want to use content-based deduplication instead of entering a
 deduplication ID? (y/n) ", "y")
 }
 log.Println(strings.Repeat("-", 88))

 topicName := runner.questioner.Ask("Enter a name for your SNS topic. ")
 if isFifoTopic {
  topicName = fmt.Sprintf("%v%v", topicName, FIFO_SUFFIX)
  log.Printf("Because you have selected a FIFO topic, '%v' must be appended to
\n"+
   "the topic name.", FIFO_SUFFIX)
 }

 topicArn, err := runner.snsActor.CreateTopic(ctx, topicName, isFifoTopic,
 contentBasedDeduplication)
 if err != nil {
```

```
  panic(err)
 }
 log.Printf("Your new topic with the name '%v' and Amazon Resource Name (ARN)
 \n"+
  "'%v' has been created.", topicName, topicArn)

 return topicName, topicArn, isFifoTopic, contentBasedDeduplication
}

func (runner ScenarioRunner) CreateQueue(ctx context.Context, ordinal string,
 isFifoTopic bool) (string, string) {
 queueName := runner.questioner.Ask(fmt.Sprintf("Enter a name for the %v SQS
 queue. ", ordinal))
 if isFifoTopic {
  queueName = fmt.Sprintf("%v%v", queueName, FIFO_SUFFIX)
  if ordinal == "first" {
   log.Printf("Because you are creating a FIFO SQS queue, '%v' must "+
    "be appended to the queue name.\n", FIFO_SUFFIX)
  }
 }
 queueUrl, err := runner.sqsActor.CreateQueue(ctx, queueName, isFifoTopic)
 if err != nil {
  panic(err)
 }
 log.Printf("Your new SQS queue with the name '%v' and the queue URL "+
  "'%v' has been created.", queueName, queueUrl)

 return queueName, queueUrl
}

func (runner ScenarioRunner) SubscribeQueueToTopic(
 ctx context.Context, queueName string, queueUrl string, topicName string,
 topicArn string, ordinal string,
 isFifoTopic bool) (string, bool) {

 queueArn, err := runner.sqsActor.GetQueueArn(ctx, queueUrl)
 if err != nil {
  panic(err)
 }
 log.Printf("The ARN of your queue is: %v.\n", queueArn)

 err = runner.sqsActor.AttachSendMessagePolicy(ctx, queueUrl, queueArn, topicArn)
 if err != nil {
  panic(err)
```

```
}
log.Println("Attached an IAM policy to the queue so the SNS topic can send " +
 "messages to it.")
log.Println(strings.Repeat("-", 88))

var filterPolicy map[string][]string
if isFifoTopic {
 if ordinal == "first" {
  log.Println("Subscriptions to a FIFO topic can have filters.\n" +
   "If you add a filter to this subscription, then only the filtered messages\n"
+
   "will be received in the queue.\n" +
   "For information about message filtering, see\n" +
   "\thttps://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n" +
   "For this example, you can filter messages by a \"tone\" attribute.")
 }

 wantFiltering := runner.questioner.AskBool(
  fmt.Sprintf("Do you want to filter messages that are sent to \"%v\"\n"+
   "from the %v topic? (y/n) ", queueName, topicName), "y")
 if wantFiltering {
  log.Println("You can filter messages by one or more of the following \"tone\"
attributes.")

  var toneSelections []string
  askAboutTones := true
  for askAboutTones {
   toneIndex := runner.questioner.AskChoice(
    "Enter the number of the tone you want to filter by:\n", ToneChoices)
   toneSelections = append(toneSelections, ToneChoices[toneIndex])
   askAboutTones = runner.questioner.AskBool("Do you want to add another tone to
the filter? (y/n) ", "y")
  }
  log.Printf("Your subscription will be filtered to only pass the following
tones: %v\n", toneSelections)
  filterPolicy = map[string][]string{TONE_KEY: toneSelections}
 }
}

subscriptionArn, err := runner.snsActor.SubscribeQueue(ctx, topicArn, queueArn,
filterPolicy)
if err != nil {
 panic(err)
}
```

```
 log.Printf("The queue %v is now subscribed to the topic %v with the subscription
 ARN %v.\n",
  queueName, topicName, subscriptionArn)

 return subscriptionArn, filterPolicy != nil
}

func (runner ScenarioRunner) PublishMessages(ctx context.Context, topicArn
 string, isFifoTopic bool, contentBasedDeduplication bool, usingFilters bool) {
 var message string
 var groupId string
 var dedupId string
 var toneSelection string
 publishMore := true
 for publishMore {
  groupId = ""
  dedupId = ""
  toneSelection = ""
  message = runner.questioner.Ask("Enter a message to publish: ")
  if isFifoTopic {
   log.Println("Because you are using a FIFO topic, you must set a message group
 ID.\n" +
    "All messages within the same group will be received in the order they were
 published.")
   groupId = runner.questioner.Ask("Enter a message group ID: ")
   if !contentBasedDeduplication {
    log.Println("Because you are not using content-based deduplication,\n" +
     "you must enter a deduplication ID.")
    dedupId = runner.questioner.Ask("Enter a deduplication ID: ")
   }
  }
  if usingFilters {
   if runner.questioner.AskBool("Add a tone attribute so this message can be
 filtered? (y/n) ", "y") {
    toneIndex := runner.questioner.AskChoice(
     "Enter the number of the tone you want to filter by:\n", ToneChoices)
    toneSelection = ToneChoices[toneIndex]
   }
  }

  err := runner.snsActor.Publish(ctx, topicArn, message, groupId, dedupId,
 TONE_KEY, toneSelection)
  if err != nil {
   panic(err)
```

```
  }
  log.Println(("Your message was published."))

  publishMore = runner.questioner.AskBool("Do you want to publish another
 messsage? (y/n) ", "y")
 }
}

func (runner ScenarioRunner) PollForMessages(ctx context.Context, queueUrls
 []string) {
 log.Println("Polling queues for messages...")
 for _, queueUrl := range queueUrls {
  var messages []types.Message
  for {
   currentMsgs, err := runner.sqsActor.GetMessages(ctx, queueUrl, 10, 1)
   if err != nil {
    panic(err)
   }
   if len(currentMsgs) == 0 {
    break
   }
   messages = append(messages, currentMsgs...)
  }
  if len(messages) == 0 {
   log.Printf("No messages were received by queue %v.\n", queueUrl)
  } else if len(messages) == 1 {
   log.Printf("One message was received by queue %v:\n", queueUrl)

  } else {
   log.Printf("%v messages were received by queue %v:\n", len(messages),
 queueUrl)
  }
  for msgIndex, message := range messages {
   messageBody := MessageBody{}
   err := json.Unmarshal([]byte(*message.Body), &messageBody)
   if err != nil {
    panic(err)
   }
   log.Printf("Message %v: %v\n", msgIndex+1, messageBody.Message)
  }

  if len(messages) > 0 {
   log.Printf("Deleting %v messages from queue %v.\n", len(messages), queueUrl)
   err := runner.sqsActor.DeleteMessages(ctx, queueUrl, messages)
```

```
   if err != nil {
    panic(err)
   }
  }
 }
}

// RunTopicsAndQueuesScenario is an interactive example that shows you how to use
 the
// AWS SDK for Go to create and use Amazon SNS topics and Amazon SQS queues.
//
// 1. Create a topic (FIFO or non-FIFO).
// 2. Subscribe several queues to the topic with an option to apply a filter.
// 3. Publish messages to the topic.
// 4. Poll the queues for messages received.
// 5. Delete the topic and the queues.
//
// This example creates service clients from the specified sdkConfig so that
// you can replace it with a mocked or stubbed config for unit testing.
//
// It uses a questioner from the `demotools` package to get input during the
 example.
// This package can be found in the ..\..\demotools folder of this repo.
func RunTopicsAndQueuesScenario(
 ctx context.Context, sdkConfig aws.Config, questioner demotools.IQuestioner) {
 resources := Resources{}
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.\n" +
    "Cleaning up any resources that were created...")
   resources.Cleanup(ctx)
  }
 }()
 queueCount := 2

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome to messaging with topics and queues.\n\n"+
  "In this 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(ctx)
 resources.topicArn = topicArn
 log.Println(strings.Repeat("-", 88))

 log.Printf("Now you will create %v SQS queues and subscribe them to the topic.
\n", queueCount)
 ordinals := []string{"first", "next"}
 usingFilters := false
 for _, ordinal := range ordinals {
  queueName, queueUrl := runner.CreateQueue(ctx, ordinal, isFifoTopic)
  resources.queueUrls = append(resources.queueUrls, queueUrl)

  _, filtering := runner.SubscribeQueueToTopic(ctx, queueName, queueUrl,
 topicName, topicArn, ordinal, isFifoTopic)
  usingFilters = usingFilters || filtering
 }

 log.Println(strings.Repeat("-", 88))
 runner.PublishMessages(ctx, topicArn, isFifoTopic, contentBasedDeduplication,
 usingFilters)
 log.Println(strings.Repeat("-", 88))
 runner.PollForMessages(ctx, resources.queueUrls)

 log.Println(strings.Repeat("-", 88))

 wantCleanup := questioner.AskBool("Do you want to remove all AWS resources
 created for this scenario? (y/n) ", "y")
 if wantCleanup {
  log.Println("Cleaning up resources...")
  resources.Cleanup(ctx)
 }
```

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

Tentukan struct yang membungkus SNS tindakan Amazon yang digunakan dalam contoh ini.

```
// SnsActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type SnsActions struct {
 SnsClient *sns.Client
}



// CreateTopic creates an Amazon SNS topic with the specified name. You can
 optionally
// specify that the topic is created as a FIFO topic and whether it uses content-
based
// deduplication instead of ID-based deduplication.
func (actor SnsActions) CreateTopic(ctx context.Context, topicName string,
 isFifoTopic bool, contentBasedDeduplication bool) (string, error) {
 var topicArn string
 topicAttributes := map[string]string{}
 if isFifoTopic {
  topicAttributes["FifoTopic"] = "true"
 }
 if contentBasedDeduplication {
  topicAttributes["ContentBasedDeduplication"] = "true"
 }
 topic, err := actor.SnsClient.CreateTopic(ctx, &sns.CreateTopicInput{
  Name:       aws.String(topicName),
  Attributes: topicAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create topic %v. Here's why: %v\n", topicName, err)
 } else {
  topicArn = *topic.TopicArn
 }
```

```
 return topicArn, err
}



// DeleteTopic delete an Amazon SNS topic.
func (actor SnsActions) DeleteTopic(ctx context.Context, topicArn string) error {
 _, err := actor.SnsClient.DeleteTopic(ctx, &sns.DeleteTopicInput{
  TopicArn: aws.String(topicArn)})
 if err != nil {
  log.Printf("Couldn't delete topic %v. Here's why: %v\n", topicArn, err)
 }
 return err
}



// SubscribeQueue subscribes an Amazon Simple Queue Service (Amazon SQS) queue to
 an
// Amazon SNS topic. When filterMap is not nil, it is used to specify a filter
 policy
// so that messages are only sent to the queue when the message has the specified
 attributes.
func (actor SnsActions) SubscribeQueue(ctx context.Context, topicArn string,
 queueArn string, filterMap map[string][]string) (string, error) {
 var subscriptionArn string
 var attributes map[string]string
 if filterMap != nil {
  filterBytes, err := json.Marshal(filterMap)
  if err != nil {
   log.Printf("Couldn't create filter policy, here's why: %v\n", err)
   return "", err
  }
  attributes = map[string]string{"FilterPolicy": string(filterBytes)}
 }
 output, err := actor.SnsClient.Subscribe(ctx, &sns.SubscribeInput{
  Protocol:              aws.String("sqs"),
  TopicArn:              aws.String(topicArn),
  Attributes:            attributes,
  Endpoint:              aws.String(queueArn),
  ReturnSubscriptionArn: true,
 })
 if err != nil {
```

```
  log.Printf("Couldn't susbscribe queue %v to topic %v. Here's why: %v\n",
   queueArn, topicArn, err)
 } else {
  subscriptionArn = *output.SubscriptionArn
 }

 return subscriptionArn, err
}



// Publish publishes a message to an Amazon SNS topic. The message is then sent
 to all
// subscribers. When the topic is a FIFO topic, the message must also contain a
 group ID
// and, when ID-based deduplication is used, a deduplication ID. An optional key-
value
// filter attribute can be specified so that the message can be filtered
 according to
// a filter policy.
func (actor SnsActions) Publish(ctx context.Context, topicArn string, message
 string, groupId string, dedupId string, filterKey string, filterValue string)
 error {
 publishInput := sns.PublishInput{TopicArn: aws.String(topicArn), Message:
 aws.String(message)}
 if groupId != "" {
  publishInput.MessageGroupId = aws.String(groupId)
 }
 if dedupId != "" {
  publishInput.MessageDeduplicationId = aws.String(dedupId)
 }
 if filterKey != "" && filterValue != "" {
  publishInput.MessageAttributes = map[string]types.MessageAttributeValue{
   filterKey: {DataType: aws.String("String"), StringValue:
 aws.String(filterValue)},
  }
 }
 _, err := actor.SnsClient.Publish(ctx, &publishInput)
 if err != nil {
  log.Printf("Couldn't publish message to topic %v. Here's why: %v", topicArn,
 err)
 }
 return err
}
```

Tentukan struct yang membungkus SQS tindakan Amazon yang digunakan dalam contoh ini.

```
// SqsActions encapsulates the Amazon Simple Queue Service (Amazon SQS) actions
// used in the examples.
type SqsActions struct {
 SqsClient *sqs.Client
}



// CreateQueue creates an Amazon SQS queue with the specified name. You can
 specify
// whether the queue is created as a FIFO queue.
func (actor SqsActions) CreateQueue(ctx context.Context, queueName string,
 isFifoQueue bool) (string, error) {
 var queueUrl string
 queueAttributes := map[string]string{}
 if isFifoQueue {
  queueAttributes["FifoQueue"] = "true"
 }
 queue, err := actor.SqsClient.CreateQueue(ctx, &sqs.CreateQueueInput{
  QueueName:  aws.String(queueName),
  Attributes: queueAttributes,
 })
 if err != nil {
  log.Printf("Couldn't create queue %v. Here's why: %v\n", queueName, err)
 } else {
  queueUrl = *queue.QueueUrl
 }

 return queueUrl, err
}



// GetQueueArn uses the GetQueueAttributes action to get the Amazon Resource Name
 (ARN)
// of an Amazon SQS queue.
```

```
func (actor SqsActions) GetQueueArn(ctx context.Context, queueUrl string)
 (string, error) {
 var queueArn string
 arnAttributeName := types.QueueAttributeNameQueueArn
 attribute, err := actor.SqsClient.GetQueueAttributes(ctx,
 &sqs.GetQueueAttributesInput{
  QueueUrl:       aws.String(queueUrl),
  AttributeNames: []types.QueueAttributeName{arnAttributeName},
 })
 if err != nil {
  log.Printf("Couldn't get ARN for queue %v. Here's why: %v\n", queueUrl, err)
 } else {
  queueArn = attribute.Attributes[string(arnAttributeName)]
 }
 return queueArn, err
}



// AttachSendMessagePolicy uses the SetQueueAttributes action to attach a policy
 to an
// Amazon SQS queue that allows the specified Amazon SNS topic to send messages
 to the
// queue.
func (actor SqsActions) AttachSendMessagePolicy(ctx context.Context, queueUrl
 string, queueArn string, topicArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
   Action:    "sqs:SendMessage",
   Principal: map[string]string{"Service": "sns.amazonaws.com"},
   Resource:  aws.String(queueArn),
   Condition: PolicyCondition{"ArnEquals": map[string]string{"aws:SourceArn":
 topicArn}},
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document. Here's why: %v\n", err)
  return err
 }
 _, err = actor.SqsClient.SetQueueAttributes(ctx, &sqs.SetQueueAttributesInput{
  Attributes: map[string]string{
```

```
   string(types.QueueAttributeNamePolicy): string(policyBytes),
  },
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't set send message policy on queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}

// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect    string
 Action    string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
 Condition PolicyCondition   `json:",omitempty"`
}

// PolicyCondition defines a condition in a policy.
type PolicyCondition map[string]map[string]string



// GetMessages uses the ReceiveMessage action to get messages from an Amazon SQS
 queue.
func (actor SqsActions) GetMessages(ctx context.Context, queueUrl string,
 maxMessages int32, waitTime int32) ([]types.Message, error) {
 var messages []types.Message
 result, err := actor.SqsClient.ReceiveMessage(ctx, &sqs.ReceiveMessageInput{
  QueueUrl:            aws.String(queueUrl),
  MaxNumberOfMessages: maxMessages,
  WaitTimeSeconds:     waitTime,
 })
 if err != nil {
```

```
  log.Printf("Couldn't get messages from queue %v. Here's why: %v\n", queueUrl,
 err)
 } else {
  messages = result.Messages
 }
 return messages, err
}



// DeleteMessages uses the DeleteMessageBatch action to delete a batch of
 messages from
// an Amazon SQS queue.
func (actor SqsActions) DeleteMessages(ctx context.Context, queueUrl string,
 messages []types.Message) error {
 entries := make([]types.DeleteMessageBatchRequestEntry, len(messages))
 for msgIndex := range messages {
  entries[msgIndex].Id = aws.String(fmt.Sprintf("%v", msgIndex))
  entries[msgIndex].ReceiptHandle = messages[msgIndex].ReceiptHandle
 }
 _, err := actor.SqsClient.DeleteMessageBatch(ctx, &sqs.DeleteMessageBatchInput{
  Entries:  entries,
  QueueUrl: aws.String(queueUrl),
 })
 if err != nil {
  log.Printf("Couldn't delete messages from queue %v. Here's why: %v\n",
 queueUrl, err)
 }
 return err
}



// DeleteQueue deletes an Amazon SQS queue.
func (actor SqsActions) DeleteQueue(ctx context.Context, queueUrl string) error {
 _, err := actor.SqsClient.DeleteQueue(ctx, &sqs.DeleteQueueInput{
  QueueUrl: aws.String(queueUrl)})
 if err != nil {
  log.Printf("Couldn't delete queue %v. Here's why: %v\n", queueUrl, err)
 }
 return err
}
```

- Untuk API detailnya, lihat topik berikut di AWS SDK for Go APIReferensi.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publikasikan
  - ReceiveMessage
  - SetQueueAttributes
  - Berlangganan
  - Berhenti berlangganan

Java

SDKuntuk Java 2.x

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

```
package com.example.sns;

import
 software.amazon.awssdk.auth.credentials.EnvironmentVariableCredentialsProvider;
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.DeleteTopicRequest;
import software.amazon.awssdk.services.sns.model.DeleteTopicResponse;
import software.amazon.awssdk.services.sns.model.MessageAttributeValue;
import software.amazon.awssdk.services.sns.model.PublishRequest;
```

```
import software.amazon.awssdk.services.sns.model.PublishResponse;
import
 software.amazon.awssdk.services.sns.model.SetSubscriptionAttributesRequest;
import software.amazon.awssdk.services.sns.model.SnsException;
import software.amazon.awssdk.services.sns.model.SubscribeRequest;
import software.amazon.awssdk.services.sns.model.SubscribeResponse;
import software.amazon.awssdk.services.sns.model.UnsubscribeRequest;
import software.amazon.awssdk.services.sns.model.UnsubscribeResponse;
import software.amazon.awssdk.services.sqs.SqsClient;
import software.amazon.awssdk.services.sqs.model.CreateQueueRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequest;
import software.amazon.awssdk.services.sqs.model.DeleteMessageBatchRequestEntry;
import software.amazon.awssdk.services.sqs.model.DeleteQueueRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueAttributesResponse;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlRequest;
import software.amazon.awssdk.services.sqs.model.GetQueueUrlResponse;
import software.amazon.awssdk.services.sqs.model.Message;
import software.amazon.awssdk.services.sqs.model.QueueAttributeName;
import software.amazon.awssdk.services.sqs.model.ReceiveMessageRequest;
import software.amazon.awssdk.services.sqs.model.SetQueueAttributesRequest;
import software.amazon.awssdk.services.sqs.model.SqsException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;
import com.google.gson.Gson;
import com.google.gson.JsonArray;
import com.google.gson.JsonObject;
import com.google.gson.JsonPrimitive;

/**
 * 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
 *
 * This Java example performs these tasks:
 *
 * 1. Gives the user three options to choose from.
```

```
 * 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 * 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 * 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 * 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 * 6. Subscribes to the SQS queue.
 * 7. Publishes a message to the topic.
 * 8. Displays the messages.
 * 9. Deletes the received message.
 * 10. Unsubscribes from the topic.
 * 11. Deletes the SNS topic.
 */
public class SNSWorkflow {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) {
        final String usage = "\n" +
                "Usage:\n" +
                "    <fifoQueueARN>\n\n" +
                "Where:\n" +
                "    accountId - Your AWS account Id value.";

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

        SnsClient snsClient = SnsClient.builder()
                .region(Region.US_EAST_1)

 .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
                .build();

        SqsClient sqsClient = SqsClient.builder()
                .region(Region.US_EAST_1)

 .credentialsProvider(EnvironmentVariableCredentialsProvider.create())
                .build();

        Scanner in = new Scanner(System.in);
        String accountId = "814548047983";
        String useFIFO;
        String duplication = "n";
        String topicName;
```

```
        String deduplicationID = null;
        String groupId = null;

        String topicArn;
        String sqsQueueName;
        String sqsQueueUrl;
        String sqsQueueArn;
        String subscriptionArn;
        boolean selectFIFO = false;

        String message;
        List<Message> messageList;
        List<String> filterList = new ArrayList<>();
        String msgAttValue = "";

        System.out.println(DASHES);
        System.out.println("Welcome to messaging with topics and queues.");
        System.out.println("In this workflow, you will create an SNS topic and
 subscribe an SQS queue to the topic.\n" +
                "You can select from several options for configuring the topic
 and the subscriptions for the queue.\n" +
                "You can then post to the topic and see the results in the
 queue.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("SNS topics can be configured as FIFO (First-In-First-
Out).\n" +
                "FIFO topics deliver messages in order and support deduplication
 and message filtering.\n" +
                "Would you like to work with FIFO topics? (y/n)");
        useFIFO = in.nextLine();
        if (useFIFO.compareTo("y") == 0) {
            selectFIFO = true;
            System.out.println("You have selected FIFO");
            System.out.println(" Because you have chosen a FIFO topic,
 deduplication is supported.\n" +
                    "       Deduplication IDs are either set in the message or
 automatically generated from content using a hash function.\n"
                    +
                    "       If a message is successfully published to an SNS
 FIFO topic, any message published and determined to have the same deduplication
 ID,\n"
                    +
```

```
                    "       within the five-minute deduplication interval, is
accepted but not delivered.\n" +
                    "       For more information about deduplication, see
https://docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.");

            System.out.println(
                    "Would you like to use content-based deduplication instead of
entering a deduplication ID? (y/n)");
            duplication = in.nextLine();
            if (duplication.compareTo("y") == 0) {
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            } else {
                System.out.println("Please enter deduplication Id value");
                deduplicationID = in.nextLine();
                System.out.println("Please enter a group id value");
                groupId = in.nextLine();
            }
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Create a topic.");
        System.out.println("Enter a name for your SNS topic.");
        topicName = in.nextLine();
        if (selectFIFO) {
            System.out.println("Because you have selected a FIFO topic, '.fifo'
must be appended to the topic name.");
            topicName = topicName + ".fifo";
            System.out.println("The name of the topic is " + topicName);
            topicArn = createFIFO(snsClient, topicName, duplication);
            System.out.println("The ARN of the FIFO topic is " + topicArn);

        } else {
            System.out.println("The name of the topic is " + topicName);
            topicArn = createSNSTopic(snsClient, topicName);
            System.out.println("The ARN of the non-FIFO topic is " + topicArn);

        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Create an SQS queue.");
        System.out.println("Enter a name for your SQS queue.");
```

```
        sqsQueueName = in.nextLine();
        if (selectFIFO) {
            sqsQueueName = sqsQueueName + ".fifo";
        }
        sqsQueueUrl = createQueue(sqsClient, sqsQueueName, selectFIFO);
        System.out.println("The queue URL is " + sqsQueueUrl);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Get the SQS queue ARN attribute.");
        sqsQueueArn = getSQSQueueAttrs(sqsClient, sqsQueueUrl);
        System.out.println("The ARN of the new queue is " + sqsQueueArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Attach an IAM policy to the queue.");

        // Define the policy to use. Make sure that you change the REGION if you
are
        // running this code
        // in a different region.
        String policy = "{\n" +
                "     \"Statement\": [\n" +
                "     {\n" +
                "         \"Effect\": \"Allow\",\n" +
                "                 \"Principal\": {\n" +
                "             \"Service\": \"sns.amazonaws.com\"\n" +
                "         },\n" +
                "         \"Action\": \"sqs:SendMessage\",\n" +
                "                 \"Resource\": \"arn:aws:sqs:us-east-1:" +
accountId + ":" + sqsQueueName + "\",\n" +
                "                 \"Condition\": {\n" +
                "             \"ArnEquals\": {\n" +
                "                 \"aws:SourceArn\": \"arn:aws:sns:us-east-1:" +
accountId + ":" + topicName + "\"\n" +
                "             }\n" +
                "         }\n" +
                "     }\n" +
                "     ]\n" +
                " }";

        setQueueAttr(sqsClient, sqsQueueUrl, policy);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("6. Subscribe to the SQS queue.");
        if (selectFIFO) {
            System.out.println(
                    "If you add a filter to this subscription, then only the
filtered messages will be received in the queue.\n"
                            +
                            "For information about message filtering, see
https://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html\n"
                            +
                            "For this example, you can filter messages by a
\"tone\" attribute.");
            System.out.println("Would you like to filter messages for " +
sqsQueueName + "'s subscription to the topic "
                    + topicName + "?  (y/n)");
            String filterAns = in.nextLine();
            if (filterAns.compareTo("y") == 0) {
                boolean moreAns = false;
                System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                while (!moreAns) {
                    System.out.println("Select a number or choose 0 to end.");
                    String ans = in.nextLine();
                    switch (ans) {
                        case "1":
                            filterList.add("cheerful");
                            break;
                        case "2":
                            filterList.add("funny");
                            break;
                        case "3":
                            filterList.add("serious");
                            break;
                        case "4":
                            filterList.add("sincere");
                            break;
                        default:
                            moreAns = true;
                            break;
                    }
```

```
                }
            }
        }
        subscriptionArn = subQueue(snsClient, topicArn, sqsQueueArn, filterList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Publish a message to the topic.");
        if (selectFIFO) {
            System.out.println("Would you like to add an attribute to this
message?  (y/n)");
            String msgAns = in.nextLine();
            if (msgAns.compareTo("y") == 0) {
                System.out.println("You can filter messages by one or more of the
following \"tone\" attributes.");
                System.out.println("1. cheerful");
                System.out.println("2. funny");
                System.out.println("3. serious");
                System.out.println("4. sincere");
                System.out.println("Select a number or choose 0 to end.");
                String ans = in.nextLine();
                switch (ans) {
                    case "1":
                        msgAttValue = "cheerful";
                        break;
                    case "2":
                        msgAttValue = "funny";
                        break;
                    case "3":
                        msgAttValue = "serious";
                        break;
                    default:
                        msgAttValue = "sincere";
                        break;
                }

                System.out.println("Selected value is " + msgAttValue);
            }
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessageFIFO(snsClient, message, topicArn, msgAttValue,
duplication, groupId, deduplicationID);

        } else {
```

```
            System.out.println("Enter a message.");
            message = in.nextLine();
            pubMessage(snsClient, message, topicArn);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Display the message. Press any key to continue.");
        in.nextLine();
        messageList = receiveMessages(sqsClient, sqsQueueUrl, msgAttValue);
        for (Message mes : messageList) {
            System.out.println("Message Id: " + mes.messageId());
            System.out.println("Full Message: " + mes.body());
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Delete the received message. Press any key to
continue.");
        in.nextLine();
        deleteMessages(sqsClient, sqsQueueUrl, messageList);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Unsubscribe from the topic and delete the queue.
Press any key to continue.");
        in.nextLine();
        unSub(snsClient, subscriptionArn);
        deleteSQSQueue(sqsClient, sqsQueueName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Delete the topic. Press any key to continue.");
        in.nextLine();
        deleteSNSTopic(snsClient, topicArn);

        System.out.println(DASHES);
        System.out.println("The SNS/SQS workflow has completed successfully.");
        System.out.println(DASHES);
    }

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

```
                    .topicArn(topicArn)
                    .build();

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

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

    public static void deleteSQSQueue(SqsClient sqsClient, String queueName) {
        try {
            GetQueueUrlRequest getQueueRequest = GetQueueUrlRequest.builder()
                    .queueName(queueName)
                    .build();

            String queueUrl = sqsClient.getQueueUrl(getQueueRequest).queueUrl();
            DeleteQueueRequest deleteQueueRequest = DeleteQueueRequest.builder()
                    .queueUrl(queueUrl)
                    .build();

            sqsClient.deleteQueue(deleteQueueRequest);
            System.out.println(queueName + " was successfully deleted.");

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

    public static void unSub(SnsClient snsClient, String subscriptionArn) {
        try {
            UnsubscribeRequest request = UnsubscribeRequest.builder()
                    .subscriptionArn(subscriptionArn)
                    .build();

            UnsubscribeResponse result = snsClient.unsubscribe(request);
            System.out.println("Status was " +
result.sdkHttpResponse().statusCode()
                    + "\nSubscription was removed for " +
request.subscriptionArn());
```

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

    public static void deleteMessages(SqsClient sqsClient, String queueUrl,
List<Message> messages) {
        try {
            List<DeleteMessageBatchRequestEntry> entries = new ArrayList<>();
            for (Message msg : messages) {
                DeleteMessageBatchRequestEntry entry =
DeleteMessageBatchRequestEntry.builder()
                        .id(msg.messageId())
                        .build();

                entries.add(entry);
            }

            DeleteMessageBatchRequest deleteMessageBatchRequest =
DeleteMessageBatchRequest.builder()
                    .queueUrl(queueUrl)
                    .entries(entries)
                    .build();

            sqsClient.deleteMessageBatch(deleteMessageBatchRequest);
            System.out.println("The batch delete of messages was successful");

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

    public static List<Message> receiveMessages(SqsClient sqsClient, String
queueUrl, String msgAttValue) {
        try {
            if (msgAttValue.isEmpty()) {
                ReceiveMessageRequest receiveMessageRequest =
ReceiveMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .maxNumberOfMessages(5)
                        .build();
```

```
                return
sqsClient.receiveMessage(receiveMessageRequest).messages();
            } else {
                // We know there are filters on the message.
                ReceiveMessageRequest receiveRequest =
ReceiveMessageRequest.builder()
                        .queueUrl(queueUrl)
                        .messageAttributeNames(msgAttValue) // Include other
message attributes if needed.
                        .maxNumberOfMessages(5)
                        .build();

                return sqsClient.receiveMessage(receiveRequest).messages();
            }

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

    public static void pubMessage(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);
        }
    }

    public static void pubMessageFIFO(SnsClient snsClient,
            String message,
            String topicArn,
```

```
            String msgAttValue,
            String duplication,
            String groupId,
            String deduplicationID) {

        try {
            PublishRequest request;
            // Means the user did not choose to use a message attribute.
            if (msgAttValue.isEmpty()) {
                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                            .message(message)
                            .messageGroupId(groupId)
                            .topicArn(topicArn)
                            .build();
                } else {
                    request = PublishRequest.builder()
                            .message(message)
                            .messageDeduplicationId(deduplicationID)
                            .messageGroupId(groupId)
                            .topicArn(topicArn)
                            .build();
                }

            } else {
                Map<String, MessageAttributeValue> messageAttributes = new
HashMap<>();
                messageAttributes.put(msgAttValue,
MessageAttributeValue.builder()
                        .dataType("String")
                        .stringValue("true")
                        .build());

                if (duplication.compareTo("y") == 0) {
                    request = PublishRequest.builder()
                            .message(message)
                            .messageGroupId(groupId)
                            .topicArn(topicArn)
                            .build();
                } else {
                    // Create a publish request with the message and attributes.
                    request = PublishRequest.builder()
                            .topicArn(topicArn)
                            .message(message)
```

```
                            .messageDeduplicationId(deduplicationID)
                            .messageGroupId(groupId)
                            .messageAttributes(messageAttributes)
                            .build();
                }
            }

            // Publish the message to the topic.
            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);
        }
    }

    // Subscribe to the SQS queue.
    public static String subQueue(SnsClient snsClient, String topicArn, String
queueArn, List<String> filterList) {
        try {
            SubscribeRequest request;
            if (filterList.isEmpty()) {
                // No filter subscription is added.
                request = SubscribeRequest.builder()
                        .protocol("sqs")
                        .endpoint(queueArn)
                        .returnSubscriptionArn(true)
                        .topicArn(topicArn)
                        .build();

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been
subscribed to the topic " + topicArn + "\n" +
                        "with the subscription ARN " + result.subscriptionArn());
                return result.subscriptionArn();
            } else {
                request = SubscribeRequest.builder()
                        .protocol("sqs")
                        .endpoint(queueArn)
                        .returnSubscriptionArn(true)
                        .topicArn(topicArn)
```

```
                        .build();

                SubscribeResponse result = snsClient.subscribe(request);
                System.out.println("The queue " + queueArn + " has been
subscribed to the topic " + topicArn + "\n" +
                        "with the subscription ARN " + result.subscriptionArn());

                String attributeName = "FilterPolicy";
                Gson gson = new Gson();
                String jsonString = "{\"tone\": []}";
                JsonObject jsonObject = gson.fromJson(jsonString,
JsonObject.class);
                JsonArray toneArray = jsonObject.getAsJsonArray("tone");
                for (String value : filterList) {
                    toneArray.add(new JsonPrimitive(value));
                }

                String updatedJsonString = gson.toJson(jsonObject);
                System.out.println(updatedJsonString);
                SetSubscriptionAttributesRequest attRequest =
SetSubscriptionAttributesRequest.builder()
                        .subscriptionArn(result.subscriptionArn())
                        .attributeName(attributeName)
                        .attributeValue(updatedJsonString)
                        .build();

                snsClient.setSubscriptionAttributes(attRequest);
                return result.subscriptionArn();
            }

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

    // Attach a policy to the queue.
    public static void setQueueAttr(SqsClient sqsClient, String queueUrl, String
policy) {
        try {
            Map<software.amazon.awssdk.services.sqs.model.QueueAttributeName,
String> attrMap = new HashMap<>();
            attrMap.put(QueueAttributeName.POLICY, policy);
```

```
            SetQueueAttributesRequest attributesRequest =
SetQueueAttributesRequest.builder()
                    .queueUrl(queueUrl)
                    .attributes(attrMap)
                    .build();

            sqsClient.setQueueAttributes(attributesRequest);
            System.out.println("The policy has been successfully attached.");

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

    public static String getSQSQueueAttrs(SqsClient sqsClient, String queueUrl) {
        // Specify the attributes to retrieve.
        List<QueueAttributeName> atts = new ArrayList<>();
        atts.add(QueueAttributeName.QUEUE_ARN);

        GetQueueAttributesRequest attributesRequest =
GetQueueAttributesRequest.builder()
                .queueUrl(queueUrl)
                .attributeNames(atts)
                .build();

        GetQueueAttributesResponse response =
sqsClient.getQueueAttributes(attributesRequest);
        Map<String, String> queueAtts = response.attributesAsStrings();
        for (Map.Entry<String, String> queueAtt : queueAtts.entrySet())
            return queueAtt.getValue();

        return "";
    }

    public static String createQueue(SqsClient sqsClient, String queueName,
Boolean selectFIFO) {
        try {
            System.out.println("\nCreate Queue");
            if (selectFIFO) {
                Map<QueueAttributeName, String> attrs = new HashMap<>();
                attrs.put(QueueAttributeName.FIFO_QUEUE, "true");
```

```
                CreateQueueRequest createQueueRequest =
CreateQueueRequest.builder()
                        .queueName(queueName)
                        .attributes(attrs)
                        .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            } else {
                CreateQueueRequest createQueueRequest =
CreateQueueRequest.builder()
                        .queueName(queueName)
                        .build();

                sqsClient.createQueue(createQueueRequest);
                System.out.println("\nGet queue url");
                GetQueueUrlResponse getQueueUrlResponse = sqsClient

.getQueueUrl(GetQueueUrlRequest.builder().queueName(queueName).build());
                return getQueueUrlResponse.queueUrl();
            }

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

    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 "";
    }

    public static String createFIFO(SnsClient snsClient, String topicName, String
 duplication) {
        try {
            // Create a FIFO topic by using the SNS service client.
            Map<String, String> topicAttributes = new HashMap<>();
            if (duplication.compareTo("n") == 0) {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "false");
            } else {
                topicAttributes.put("FifoTopic", "true");
                topicAttributes.put("ContentBasedDeduplication", "true");
            }

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

            CreateTopicResponse response = snsClient.createTopic(topicRequest);
            return response.topicArn();

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

- Untuk API detailnya, lihat topik berikut di AWS SDK for Java 2.x APIReferensi.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue

- DeleteTopic
- GetQueueAttributes
- Publikasikan
- ReceiveMessage
- SetQueueAttributes
- Berlangganan
- Berhenti berlangganan

JavaScript

SDKuntuk JavaScript (v3)

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

Ini adalah titik masuk untuk alur kerja ini.

```
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";

export const startSnsWorkflow = () => {
  const snsClient = new SNSClient({});
  const sqsClient = new SQSClient({});
  const prompter = new Prompter();
  const logger = console;

  const wkflw = new TopicsQueuesWkflw(snsClient, sqsClient, prompter, logger);

  wkflw.start();
};
```

Kode sebelumnya menyediakan dependensi yang diperlukan dan memulai alur kerja. Bagian selanjutnya berisi sebagian besar contoh.

```
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.
    const 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;
    let deduplicationId;
    let 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();
    }
  }
}
```

- Untuk API detailnya, lihat topik berikut di AWS SDK for JavaScript APIReferensi.
  - CreateQueue
  - CreateTopic
  - DeleteMessageBatch
  - DeleteQueue
  - DeleteTopic
  - GetQueueAttributes
  - Publikasikan
  - ReceiveMessage
  - SetQueueAttributes
  - Berlangganan
  - Berhenti berlangganan

Kotlin

SDKuntuk Kotlin

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

```
package com.example.sns

import aws.sdk.kotlin.services.sns.SnsClient
```

```
import aws.sdk.kotlin.services.sns.model.CreateTopicRequest
import aws.sdk.kotlin.services.sns.model.DeleteTopicRequest
import aws.sdk.kotlin.services.sns.model.PublishRequest
import aws.sdk.kotlin.services.sns.model.SetSubscriptionAttributesRequest
import aws.sdk.kotlin.services.sns.model.SubscribeRequest
import aws.sdk.kotlin.services.sns.model.UnsubscribeRequest
import aws.sdk.kotlin.services.sqs.SqsClient
import aws.sdk.kotlin.services.sqs.model.CreateQueueRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequest
import aws.sdk.kotlin.services.sqs.model.DeleteMessageBatchRequestEntry
import aws.sdk.kotlin.services.sqs.model.DeleteQueueRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueAttributesRequest
import aws.sdk.kotlin.services.sqs.model.GetQueueUrlRequest
import aws.sdk.kotlin.services.sqs.model.Message
import aws.sdk.kotlin.services.sqs.model.QueueAttributeName
import aws.sdk.kotlin.services.sqs.model.ReceiveMessageRequest
import aws.sdk.kotlin.services.sqs.model.SetQueueAttributesRequest
import com.google.gson.Gson
import com.google.gson.JsonObject
import com.google.gson.JsonPrimitive
import java.util.Scanner

/**
Before running this Kotlin code example, set up your development environment,
including your AWS credentials.

For more information, see the following documentation topic:
https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

This Kotlin example performs the following tasks:

 1. Gives the user three options to choose from.
 2. Creates an Amazon Simple Notification Service (Amazon SNS) topic.
 3. Creates an Amazon Simple Queue Service (Amazon SQS) queue.
 4. Gets the SQS queue Amazon Resource Name (ARN) attribute.
 5. Attaches an AWS Identity and Access Management (IAM) policy to the queue.
 6. Subscribes to the SQS queue.
 7. Publishes a message to the topic.
 8. Displays the messages.
 9. Deletes the received message.
 10. Unsubscribes from the topic.
 11. Deletes the SNS topic.
 */
```

```
val DASHES: String = String(CharArray(80)).replace("\u0000", "-")
suspend fun main() {
    val input = Scanner(System.`in`)
    val useFIFO: String
    var duplication = "n"
    var topicName: String
    var deduplicationID: String? = null
    var groupId: String? = null
    val topicArn: String?
    var sqsQueueName: String
    val sqsQueueUrl: String?
    val sqsQueueArn: String
    val subscriptionArn: String?
    var selectFIFO = false
    val message: String
    val messageList: List<Message?>?
    val filterList = ArrayList<String>()
    var msgAttValue = ""

    println(DASHES)
    println("Welcome to the AWS SDK for Kotlin messaging with topics and
 queues.")
    println(
        """
                In this workflow, you will create an SNS topic and subscribe an
 SQS queue to the topic.
                You can select from several options for configuring the topic and
 the subscriptions for the queue.
                You can then post to the topic and see the results in the queue.
        """.trimIndent(),
    )
    println(DASHES)

    println(DASHES)
    println(
        """
                SNS topics can be configured as FIFO (First-In-First-Out).
                FIFO topics deliver messages in order and support deduplication
 and message filtering.
                Would you like to work with FIFO topics? (y/n)
        """.trimIndent(),
    )
    useFIFO = input.nextLine()
    if (useFIFO.compareTo("y") == 0) {
```

```
        selectFIFO = true
        println("You have selected FIFO")
        println(
            """ Because you have chosen a FIFO topic, deduplication is supported.
        Deduplication IDs are either set in the message or automatically
 generated from content using a hash function.
        If a message is successfully published to an SNS FIFO topic, any message
 published and determined to have the same deduplication ID,
        within the five-minute deduplication interval, is accepted but not
 delivered.
        For more information about deduplication, see https://
docs.aws.amazon.com/sns/latest/dg/fifo-message-dedup.html.""",
        )

        println("Would you like to use content-based deduplication instead of
 entering a deduplication ID? (y/n)")
        duplication = input.nextLine()
        if (duplication.compareTo("y") == 0) {
            println("Enter a group id value")
            groupId = input.nextLine()
        } else {
            println("Enter deduplication Id value")
            deduplicationID = input.nextLine()
            println("Enter a group id value")
            groupId = input.nextLine()
        }
    }
    println(DASHES)

    println(DASHES)
    println("2. Create a topic.")
    println("Enter a name for your SNS topic.")
    topicName = input.nextLine()
    if (selectFIFO) {
        println("Because you have selected a FIFO topic, '.fifo' must be appended
 to the topic name.")
        topicName = "$topicName.fifo"
        println("The name of the topic is $topicName")
        topicArn = createFIFO(topicName, duplication)
        println("The ARN of the FIFO topic is $topicArn")
    } else {
        println("The name of the topic is $topicName")
        topicArn = createSNSTopic(topicName)
        println("The ARN of the non-FIFO topic is $topicArn")
```

```
    }
    println(DASHES)

    println(DASHES)
    println("3. Create an SQS queue.")
    println("Enter a name for your SQS queue.")
    sqsQueueName = input.nextLine()
    if (selectFIFO) {
        sqsQueueName = "$sqsQueueName.fifo"
    }
    sqsQueueUrl = createQueue(sqsQueueName, selectFIFO)
    println("The queue URL is $sqsQueueUrl")
    println(DASHES)

    println(DASHES)
    println("4. Get the SQS queue ARN attribute.")
    sqsQueueArn = getSQSQueueAttrs(sqsQueueUrl)
    println("The ARN of the new queue is $sqsQueueArn")
    println(DASHES)

    println(DASHES)
    println("5. Attach an IAM policy to the queue.")
    // Define the policy to use.
    val policy = """{
     "Statement": [
     {
         "Effect": "Allow",
                 "Principal": {
             "Service": "sns.amazonaws.com"
         },
         "Action": "sqs:SendMessage",
                 "Resource": "$sqsQueueArn",
                 "Condition": {
             "ArnEquals": {
                 "aws:SourceArn": "$topicArn"
             }
         }
     }
     ]
     }"""
    setQueueAttr(sqsQueueUrl, policy)
    println(DASHES)

    println(DASHES)
```

```
    println("6. Subscribe to the SQS queue.")
    if (selectFIFO) {
        println(
            """If you add a filter to this subscription, then only the filtered
 messages will be received in the queue.
For information about message filtering, see https://docs.aws.amazon.com/sns/
latest/dg/sns-message-filtering.html
For this example, you can filter messages by a "tone" attribute.""",
        )
        println("Would you like to filter messages for $sqsQueueName's
 subscription to the topic $topicName?  (y/n)")
        val filterAns: String = input.nextLine()
        if (filterAns.compareTo("y") == 0) {
            var moreAns = false
            println("You can filter messages by using one or more of the
 following \"tone\" attributes.")
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            while (!moreAns) {
                println("Select a number or choose 0 to end.")
                val ans: String = input.nextLine()
                when (ans) {
                    "1" -> filterList.add("cheerful")
                    "2" -> filterList.add("funny")
                    "3" -> filterList.add("serious")
                    "4" -> filterList.add("sincere")
                    else -> moreAns = true
                }
            }
        }
    }
    subscriptionArn = subQueue(topicArn, sqsQueueArn, filterList)
    println(DASHES)

    println(DASHES)
    println("7. Publish a message to the topic.")
    if (selectFIFO) {
        println("Would you like to add an attribute to this message?  (y/n)")
        val msgAns: String = input.nextLine()
        if (msgAns.compareTo("y") == 0) {
            println("You can filter messages by one or more of the following
 \"tone\" attributes.")
```

```
            println("1. cheerful")
            println("2. funny")
            println("3. serious")
            println("4. sincere")
            println("Select a number or choose 0 to end.")
            val ans: String = input.nextLine()
            msgAttValue = when (ans) {
                "1" -> "cheerful"
                "2" -> "funny"
                "3" -> "serious"
                else -> "sincere"
            }
            println("Selected value is $msgAttValue")
        }
        println("Enter a message.")
        message = input.nextLine()
        pubMessageFIFO(message, topicArn, msgAttValue, duplication, groupId,
deduplicationID)
    } else {
        println("Enter a message.")
        message = input.nextLine()
        pubMessage(message, topicArn)
    }
    println(DASHES)

    println(DASHES)
    println("8. Display the message. Press any key to continue.")
    input.nextLine()
    messageList = receiveMessages(sqsQueueUrl, msgAttValue)
    if (messageList != null) {
        for (mes in messageList) {
            println("Message Id: ${mes.messageId}")
            println("Full Message: ${mes.body}")
        }
    }
    println(DASHES)

    println(DASHES)
    println("9. Delete the received message. Press any key to continue.")
    input.nextLine()
    if (messageList != null) {
        deleteMessages(sqsQueueUrl, messageList)
    }
    println(DASHES)
```

```
    println(DASHES)
    println("10. Unsubscribe from the topic and delete the queue. Press any key
 to continue.")
    input.nextLine()
    unSub(subscriptionArn)
    deleteSQSQueue(sqsQueueName)
    println(DASHES)

    println(DASHES)
    println("11. Delete the topic. Press any key to continue.")
    input.nextLine()
    deleteSNSTopic(topicArn)
    println(DASHES)

    println(DASHES)
    println("The SNS/SQS workflow has completed successfully.")
    println(DASHES)
}

suspend fun deleteSNSTopic(topicArnVal: String?) {
    val request = DeleteTopicRequest {
        topicArn = topicArnVal
    }

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

suspend fun deleteSQSQueue(queueNameVal: String) {
    val getQueueRequest = GetQueueUrlRequest {
        queueName = queueNameVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val queueUrlVal = sqsClient.getQueueUrl(getQueueRequest).queueUrl
        val deleteQueueRequest = DeleteQueueRequest {
            queueUrl = queueUrlVal
        }

        sqsClient.deleteQueue(deleteQueueRequest)
        println("$queueNameVal was successfully deleted.")
```

```
    }
}

suspend fun unSub(subscripArn: String?) {
    val request = UnsubscribeRequest {
        subscriptionArn = subscripArn
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        snsClient.unsubscribe(request)
        println("Subscription was removed for $subscripArn")
    }
}

suspend fun deleteMessages(queueUrlVal: String?, messages: List<Message>) {
    val entriesVal: MutableList<DeleteMessageBatchRequestEntry> = mutableListOf()
    for (msg in messages) {
        val entry = DeleteMessageBatchRequestEntry {
            id = msg.messageId
        }
        entriesVal.add(entry)
    }

    val deleteMessageBatchRequest = DeleteMessageBatchRequest {
        queueUrl = queueUrlVal
        entries = entriesVal
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.deleteMessageBatch(deleteMessageBatchRequest)
        println("The batch delete of messages was successful")
    }
}

suspend fun receiveMessages(queueUrlVal: String?, msgAttValue: String):
 List<Message>? {
    if (msgAttValue.isEmpty()) {
        val request = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(request).messages
        }
    } else {
```

```
        val receiveRequest = ReceiveMessageRequest {
            queueUrl = queueUrlVal
            waitTimeSeconds = 1
            maxNumberOfMessages = 5
        }
        SqsClient { region = "us-east-1" }.use { sqsClient ->
            return sqsClient.receiveMessage(receiveRequest).messages
        }
    }
}

suspend fun pubMessage(messageVal: String?, topicArnVal: 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.")
    }
}

suspend fun pubMessageFIFO(
    messageVal: String?,
    topicArnVal: String?,
    msgAttValue: String,
    duplication: String,
    groupIdVal: String?,
    deduplicationID: String?,
) {
    // Means the user did not choose to use a message attribute.
    if (msgAttValue.isEmpty()) {
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

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

```
        } else {
            val request = PublishRequest {
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    } else {
        val messAttr = aws.sdk.kotlin.services.sns.model.MessageAttributeValue {
            dataType = "String"
            stringValue = "true"
        }

        val mapAtt: Map<String,
aws.sdk.kotlin.services.sns.model.MessageAttributeValue> =
            mapOf(msgAttValue to messAttr)
        if (duplication.compareTo("y") == 0) {
            val request = PublishRequest {
                message = messageVal
                messageGroupId = groupIdVal
                topicArn = topicArnVal
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        } else {
            // Create a publish request with the message and attributes.
            val request = PublishRequest {
                topicArn = topicArnVal
                message = messageVal
                messageDeduplicationId = deduplicationID
                messageGroupId = groupIdVal
                messageAttributes = mapAtt
            }

            SnsClient { region = "us-east-1" }.use { snsClient ->
```

```
                val result = snsClient.publish(request)
                println(result.messageId.toString() + " Message sent.")
            }
        }
    }
}

// Subscribe to the SQS queue.
suspend fun subQueue(topicArnVal: String?, queueArnVal: String, filterList:
 List<String?>): String? {
    val request: SubscribeRequest
    if (filterList.isEmpty()) {
        // No filter subscription is added.
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println(
                "The queue " + queueArnVal + " has been subscribed to the topic "
 + topicArnVal + "\n" +
                    "with the subscription ARN " + result.subscriptionArn,
            )
            return result.subscriptionArn
        }
    } else {
        request = SubscribeRequest {
            protocol = "sqs"
            endpoint = queueArnVal
            returnSubscriptionArn = true
            topicArn = topicArnVal
        }

        SnsClient { region = "us-east-1" }.use { snsClient ->
            val result = snsClient.subscribe(request)
            println("The queue $queueArnVal has been subscribed to the topic
 $topicArnVal with the subscription ARN ${result.subscriptionArn}")

            val attributeNameVal = "FilterPolicy"
            val gson = Gson()
```

```
            val jsonString = "{\"tone\": []}"
            val jsonObject = gson.fromJson(jsonString, JsonObject::class.java)
            val toneArray = jsonObject.getAsJsonArray("tone")
            for (value: String? in filterList) {
                toneArray.add(JsonPrimitive(value))
            }

            val updatedJsonString: String = gson.toJson(jsonObject)
            println(updatedJsonString)
            val attRequest = SetSubscriptionAttributesRequest {
                subscriptionArn = result.subscriptionArn
                attributeName = attributeNameVal
                attributeValue = updatedJsonString
            }

            snsClient.setSubscriptionAttributes(attRequest)
            return result.subscriptionArn
        }
    }
}

suspend fun setQueueAttr(queueUrlVal: String?, policy: String) {
    val attrMap: MutableMap<String, String> = HashMap()
    attrMap[QueueAttributeName.Policy.toString()] = policy

    val attributesRequest = SetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributes = attrMap
    }

    SqsClient { region = "us-east-1" }.use { sqsClient ->
        sqsClient.setQueueAttributes(attributesRequest)
        println("The policy has been successfully attached.")
    }
}

suspend fun getSQSQueueAttrs(queueUrlVal: String?): String {
    val atts: MutableList<QueueAttributeName> = ArrayList()
    atts.add(QueueAttributeName.QueueArn)

    val attributesRequest = GetQueueAttributesRequest {
        queueUrl = queueUrlVal
        attributeNames = atts
    }
```

```
    SqsClient { region = "us-east-1" }.use { sqsClient ->
        val response = sqsClient.getQueueAttributes(attributesRequest)
        val mapAtts = response.attributes
        if (mapAtts != null) {
            mapAtts.forEach { entry ->
                println("${entry.key} : ${entry.value}")
                return entry.value
            }
        }
    }
    return ""
}

suspend fun createQueue(queueNameVal: String?, selectFIFO: Boolean): String? {
    println("\nCreate Queue")
    if (selectFIFO) {
        val attrs = mutableMapOf<String, String>()
        attrs[QueueAttributeName.FifoQueue.toString()] = "true"

        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
            attributes = attrs
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("\nGet queue url")

            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    } else {
        val createQueueRequest = CreateQueueRequest {
            queueName = queueNameVal
        }

        SqsClient { region = "us-east-1" }.use { sqsClient ->
            sqsClient.createQueue(createQueueRequest)
            println("Get queue url")
```

```
            val urlRequest = GetQueueUrlRequest {
                queueName = queueNameVal
            }

            val getQueueUrlResponse = sqsClient.getQueueUrl(urlRequest)
            return getQueueUrlResponse.queueUrl
        }
    }
}

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
    }
}

suspend fun createFIFO(topicName: String?, duplication: String): String? {
    val topicAttributes: MutableMap<String, String> = HashMap()
    if (duplication.compareTo("n") == 0) {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "false"
    } else {
        topicAttributes["FifoTopic"] = "true"
        topicAttributes["ContentBasedDeduplication"] = "true"
    }

    val topicRequest = CreateTopicRequest {
        name = topicName
        attributes = topicAttributes
    }
    SnsClient { region = "us-east-1" }.use { snsClient ->
        val response = snsClient.createTopic(topicRequest)
        return response.topicArn
    }
}
```

- Untuk API detailnya, lihat topik berikut AWS SDKuntuk API referensi Kotlin.

- CreateQueue
- CreateTopic
- DeleteMessageBatch
- DeleteQueue
- DeleteTopic
- GetQueueAttributes
- Publikasikan
- ReceiveMessage
- SetQueueAttributes
- Berlangganan
- Berhenti berlangganan

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Gunakan API Gateway untuk menjalankan fungsi Lambda

Contoh kode berikut menunjukkan cara membuat AWS Lambda fungsi yang dipanggil oleh Amazon API Gateway.

Java

SDKuntuk Java 2.x

Menunjukkan cara membuat AWS Lambda fungsi dengan menggunakan runtime Lambda Java. API Contoh ini memanggil AWS layanan yang berbeda untuk melakukan kasus penggunaan tertentu. Contoh ini menunjukkan cara membuat fungsi Lambda yang dipanggil oleh API Amazon Gateway yang memindai tabel Amazon DynamoDB untuk peringatan kerja dan menggunakan Amazon Simple SNS Notification Service (Amazon) untuk mengirim pesan teks kepada karyawan Anda yang memberi selamat kepada mereka pada tanggal ulang tahun satu tahun mereka.

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

Layanan yang digunakan dalam contoh ini

- APIGerbang
- DynamoDB
- Lambda
- Amazon SNS

JavaScript

SDKuntuk JavaScript (v3)

Menunjukkan cara membuat AWS Lambda fungsi dengan menggunakan runtime Lambda JavaScript . API Contoh ini memanggil AWS layanan yang berbeda untuk melakukan kasus penggunaan tertentu. Contoh ini menunjukkan cara membuat fungsi Lambda yang dipanggil oleh API Amazon Gateway yang memindai tabel Amazon DynamoDB untuk peringatan kerja dan menggunakan Amazon Simple SNS Notification Service (Amazon) untuk mengirim pesan teks kepada karyawan Anda yang memberi selamat kepada mereka pada tanggal ulang tahun satu tahun mereka.

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

Contoh ini juga tersedia di panduan developer v3 AWS SDK for JavaScript.

Layanan yang digunakan dalam contoh ini

- APIGerbang
- DynamoDB
- Lambda
- Amazon SNS

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Menggunakan peristiwa terjadwal untuk menginvokasi fungsi Lambda

Contoh kode berikut menunjukkan cara membuat AWS Lambda fungsi yang dipanggil oleh acara EventBridge terjadwal Amazon.

Java

SDKuntuk Java 2.x

Menunjukkan cara membuat acara EventBridge terjadwal Amazon yang memanggil AWS Lambda fungsi. Konfigurasikan EventBridge untuk menggunakan ekspresi cron untuk menjadwalkan saat fungsi Lambda dipanggil. Dalam contoh ini, Anda membuat fungsi Lambda dengan menggunakan runtime Lambda Java. API Contoh ini memanggil AWS layanan yang berbeda untuk melakukan kasus penggunaan tertentu. Contoh ini menunjukkan cara membuat aplikasi yang mengirimkan pesan teks seluler kepada karyawan Anda berisi ucapan selamat pada hari jadi setahun kerja mereka.

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

Layanan yang digunakan dalam contoh ini

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

JavaScript

SDKuntuk JavaScript (v3)

Menunjukkan cara membuat acara EventBridge terjadwal Amazon yang memanggil AWS Lambda fungsi. Konfigurasikan EventBridge untuk menggunakan ekspresi cron untuk menjadwalkan saat fungsi Lambda dipanggil. Dalam contoh ini, Anda membuat fungsi Lambda dengan menggunakan runtime Lambda. JavaScript API Contoh ini memanggil AWS layanan yang berbeda untuk melakukan kasus penggunaan tertentu. Contoh ini menunjukkan cara membuat aplikasi yang mengirimkan pesan teks seluler kepada karyawan Anda berisi ucapan selamat pada hari jadi setahun kerja mereka.

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

Contoh ini juga tersedia di panduan developer v3 AWS SDK for JavaScript.

Layanan yang digunakan dalam contoh ini

- DynamoDB
- EventBridge
- Lambda
- Amazon SNS

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

Contoh tanpa server untuk Amazon menggunakan SNS AWS SDKs

Contoh kode berikut menunjukkan cara menggunakan Amazon SNS dengan AWS SDKs.

Contoh

- Memanggil fungsi Lambda dari pemicu Amazon SNS

Memanggil fungsi Lambda dari pemicu Amazon SNS

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

.NET

AWS SDK for .NET

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

Mengkonsumsi SNS acara dengan menggunakan Lambda. NET.

```
// 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

SDKuntuk Go V2

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

Mengkonsumsi SNS acara dengan Lambda menggunakan Go.

```
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0
package main

import (
 "context"
 "fmt"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
)

func handler(ctx context.Context, snsEvent events.SNSEvent) {
 for _, record := range snsEvent.Records {
  processMessage(record)
 }
 fmt.Println("done")
}

func processMessage(record events.SNSEventRecord) {
 message := record.SNS.Message
 fmt.Printf("Processed message: %s\n", message)
 // TODO: Process your record here
}

func main() {
 lambda.Start(handler)
}
```

Java

SDKuntuk Java 2.x

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

Mengkonsumsi SNS acara dengan Lambda menggunakan Java.

```
// 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

SDKuntuk JavaScript (v3)

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

Mengkonsumsi SNS acara dengan menggunakan Lambda. 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;
  }
}
```

Mengkonsumsi SNS acara dengan menggunakan Lambda. 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

SDKuntuk PHP

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

Mengkonsumsi SNS acara dengan menggunakan Lambda. PHP

```
// 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

SDKuntuk Python (Boto3)

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

Mengkonsumsi SNS acara dengan Lambda menggunakan Python.

```
# 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

SDKuntuk Ruby

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

Mengkonsumsi SNS acara dengan Lambda menggunakan Ruby.

```
# 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

SDKuntuk Rust

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

Mengkonsumsi SNS acara dengan Lambda menggunakan Rust.

```
// 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
}
```

Untuk daftar lengkap panduan AWS SDK pengembang dan contoh kode, lihatMenggunakan Amazon SNS dengan AWS SDK. Topik ini juga mencakup informasi tentang memulai dan detail tentang SDK versi sebelumnya.

SNSKeamanan Amazon

Bagian ini memberikan informasi tentang SNS keamanan Amazon, otentikasi dan kontrol akses, dan Bahasa Kebijakan SNS Akses Amazon.

Topik

Perlindungan SNS data Amazon

Model tanggung jawab AWS bersama model berlaku untuk perlindungan data di Amazon Simple Notification Service. Seperti yang dijelaskan dalam model AWS ini, bertanggung jawab untuk melindungi infrastruktur global yang menjalankan semua AWS Cloud. Anda bertanggung jawab untuk mempertahankan kendali atas konten yang di-host pada infrastruktur ini. Konten ini mencakup konfigurasi keamanan dan tugas manajemen untuk AWS layanan yang Anda gunakan. Untuk informasi selengkapnya tentang privasi data, lihat Privasi Data FAQ. Untuk informasi tentang perlindungan data di Eropa, lihat Model Tanggung Jawab AWS Bersama dan posting GDPR blog di Blog AWS Keamanan.

Untuk tujuan perlindungan data, kami menyarankan Anda melindungi Akun AWS kredensi dan menyiapkan akun pengguna individual dengan AWS Identity and Access Management ()IAM. Dengan cara ini, setiap pengguna hanya diberi izin yang diperlukan untuk memenuhi tanggung jawab tugas mereka. Kami juga merekomendasikan agar Anda mengamankan data Anda dengan cara-cara berikut ini:

- Gunakan otentikasi multi-faktor (MFA) dengan setiap akun.
- GunakanSSL/TLSuntuk berkomunikasi dengan AWS sumber daya. Kami merekomendasikan TLS 1.2 atau yang lebih baru.
- Siapkan API dan pencatatan aktivitas pengguna dengan AWS CloudTrail.

- Gunakan solusi AWS enkripsi, bersama dengan semua kontrol keamanan default dalam AWS layanan.
- Gunakan layanan keamanan terkelola lanjutan seperti Amazon Macie, yang membantu menemukan dan mengamankan data pribadi yang disimpan di Amazon S3.
- Jika Anda memerlukan FIPS 140-2 modul kriptografi yang divalidasi saat mengakses AWS melalui antarmuka baris perintah atau, gunakan titik akhir. API FIPS Untuk informasi selengkapnya tentang FIPS titik akhir yang tersedia, lihat Federal Information Processing Standard (FIPS) 140-2.
- Perlindungan data pesan
  - Perlindungan data pesan adalah fitur utama baru Amazon SNS
  - Gunakan MDP untuk memindai pesan untuk informasi rahasia atau sensitif
  - Memberikan audit pesan ke semua konten yang mengalir melalui topik
  - Menyediakan kontrol akses konten ke pesan yang dipublikasikan ke topik dan pesan yang disampaikan oleh topik

> **Important**
>
> Kami sangat merekomendasikan agar Anda tidak pernah memasukkan informasi identifikasi yang sensitif, seperti nomor rekening pelanggan Anda, ke dalam tanda atau bidang isian bebas seperti bidang Nama. Ini termasuk saat Anda bekerja dengan Amazon SNS atau Amazon Web Services lainnya menggunakan konsol,API, AWS CLI, atau AWS SDKs. Data apa pun yang Anda masukkan ke dalam tanda atau bidang isian bebas yang digunakan untuk nama dapat digunakan untuk log penagihan atau log diagnostik. Jika Anda memberikan URL ke server eksternal, kami sangat menyarankan agar Anda tidak menyertakan informasi kredensional dalam URL untuk memvalidasi permintaan Anda ke server tersebut.

Bagian berikut memberikan informasi tambahan tentang perlindungan data di AmazonSNS.

Topik

- Enkripsi SNS data Amazon
- Mengamankan SNS lalu lintas Amazon dengan titik akhir VPC
- Meningkatkan SNS keamanan Amazon dengan Perlindungan Data Pesan

Enkripsi SNS data Amazon

Perlindungan data mengacu pada melindungi data saat dalam perjalanan (saat bepergian ke dan dari AmazonSNS) dan saat istirahat (saat disimpan di disk di pusat SNS data Amazon). Anda dapat melindungi data dalam perjalanan menggunakan Secure Sockets Layer (SSL) atau enkripsi sisi klien. Secara default, Amazon SNS menyimpan pesan dan file menggunakan enkripsi disk. Anda dapat melindungi data saat istirahat dengan meminta Amazon SNS untuk mengenkripsi pesan Anda sebelum menyimpannya ke sistem file terenkripsi di pusat datanya. Amazon SNS merekomendasikan penggunaan SSE untuk enkripsi data yang dioptimalkan.

Topik

Mengamankan SNS data Amazon dengan enkripsi sisi server

Enkripsi sisi server (SSE) memungkinkan Anda menyimpan data sensitif dalam topik terenkripsi dengan melindungi konten pesan dalam SNS topik Amazon menggunakan kunci yang dikelola di (). AWS Key Management Service AWS KMS

SSEmengenkripsi pesan segera setelah Amazon SNS menerimanya. Pesan disimpan dalam bentuk terenkripsi, dan hanya didekripsi saat dikirim.

- Untuk informasi tentang mengelola SSE menggunakan AWS Management Console atau AWS SDK for Java (dengan menyetel `KmsMasterKeyId` atribut menggunakan `CreateTopic` dan `SetTopicAttributes` API tindakan), lihatMenyiapkan enkripsi SNS topik Amazon dengan enkripsi sisi server.
- Untuk informasi tentang membuat topik terenkripsi menggunakan AWS CloudFormation (dengan menyetel `KmsMasterKeyId` properti menggunakan `AWS::SNS::Topic` sumber daya), lihat AWS CloudFormation Panduan Pengguna.

> **Important**
>
> Semua permintaan ke topik dengan SSE diaktifkan harus menggunakan HTTPS dan Versi Tanda Tangan 4.
> Untuk informasi tentang kompatibilitas layanan lainnya dengan topik terenkripsi, lihat dokumentasi layanan Anda.
> Amazon SNS hanya mendukung KMS kunci enkripsi simetris. Anda tidak dapat menggunakan jenis KMS kunci lain untuk mengenkripsi sumber daya layanan Anda. Untuk bantuan menentukan apakah KMS kunci adalah kunci enkripsi simetris, lihat Mengidentifikasi kunci asimetris KMS.

AWS KMS menggabungkan perangkat keras dan perangkat lunak yang aman dan sangat tersedia untuk menyediakan sistem manajemen kunci yang diskalakan untuk cloud. Saat Anda menggunakan Amazon SNS AWS KMS, kunci data yang mengenkripsi data pesan Anda juga dienkripsi dan disimpan dengan data yang mereka lindungi.

Berikut ini adalah manfaat menggunakan AWS KMS:

- Anda dapat membuat dan mengelola AWS KMS keysendiri.
- Anda juga dapat menggunakan KMS kunci yang AWS dikelola untuk AmazonSNS, yang unik untuk setiap akun dan wilayah.
- Standar AWS KMS keamanan dapat membantu Anda memenuhi persyaratan kepatuhan terkait enkripsi.

Untuk informasi lebih lanjut, lihat Apa itu AWS Key Management Service? di Panduan AWS Key Management Service Pengembang.

Topik

- Lingkup enkripsi
- Istilah kunci

Lingkup enkripsi

SSEmengenkripsi isi pesan dalam topik AmazonSNS.

SSEtidak mengenkripsi yang berikut:

- Metadata topik (nama topik dan atribut)
- Metadata pesan (subjek, ID pesan, timestamp, dan atribut)
- Kebijakan perlindungan data
- Metrik per topik

> **Note**
>
> - Pesan dienkripsi hanya jika dikirim setelah enkripsi topik diaktifkan. Amazon SNS tidak mengenkripsi pesan yang di-backlog.
> - Setiap pesan terenkripsi tetap dienkripsi bahkan jika enkripsi topiknya dinonaktifkan.

Istilah kunci

Istilah-istilah kunci berikut dapat membantu Anda lebih memahami fungsionalitasSSE. Untuk deskripsi terperinci, lihat APIReferensi Layanan Pemberitahuan Sederhana Amazon.

Kunci data

Kunci enkripsi data (DEK) bertanggung jawab untuk mengenkripsi konten pesan AmazonSNS.

Untuk informasi lebih lanjut, lihat Kunci Data dalam Panduan Developer AWS Key Management Service dan Enkripsi Amplop dalam Panduan Developer AWS Encryption SDK .

AWS KMS key ID

Alias, aliasARN, ID kunci, atau kunci ARN dari AWS KMS key, atau kustom AWS KMS—di akun Anda atau di akun lain. Sementara alias yang AWS dikelola AWS KMS untuk Amazon SNS selalu`alias/aws/sns`, alias kustom AWS KMS dapat, misalnya, menjadi. `alias/MyAlias` Anda dapat menggunakan AWS KMS kunci ini untuk melindungi pesan dalam SNS topik Amazon.

> **Note**
>
> Ingatlah hal-hal berikut ini:
>
> - Pertama kali Anda menggunakan AWS Management Console untuk menentukan yang AWS dikelola KMS untuk Amazon SNS untuk suatu topik, AWS KMS membuat yang AWS dikelola KMS untuk AmazonSNS.

- Atau, saat pertama kali Anda menggunakan `Publish` tindakan pada topik yang SSE diaktifkan, AWS KMS membuat yang AWS dikelola KMS untuk AmazonSNS.

Anda dapat membuat AWS KMS kunci, menentukan kebijakan yang mengontrol bagaimana AWS KMS kunci dapat digunakan, dan mengaudit AWS KMS penggunaan menggunakan AWS KMS keysbagian AWS KMS konsol atau `CreateKey` AWS KMS tindakan. Untuk informasi selengkapnya, lihat AWS KMS keysdan Membuat Kunci di Panduan AWS Key Management Service Pengembang. Untuk contoh AWS KMS pengidentifikasi lainnya, lihat KeyIddi AWS Key Management Service APIReferensi. Untuk informasi tentang menemukan AWS KMS pengenal, lihat Menemukan ID Kunci dan ARN di Panduan AWS Key Management Service Pengembang.

Important

Ada biaya tambahan untuk penggunaan AWS KMS. Untuk informasi selengkapnya, lihat Memperkirakan biaya AWS KMS dan Harga AWS Key Management Service.

Mengelola kunci dan biaya SNS enkripsi Amazon

Bagian berikut ini memberikan informasi tentang bekerja dengan kunci yang terkelola di AWS Key Management Service (AWS KMS). Untuk lebih lanjut tentang

Note

Amazon SNS hanya mendukung KMS kunci enkripsi simetris. Anda tidak dapat menggunakan jenis KMS kunci lain untuk mengenkripsi sumber daya layanan Anda. Untuk bantuan menentukan apakah KMS kunci adalah kunci enkripsi simetris, lihat Mengidentifikasi kunci asimetris KMS.

Topik

- Memperkirakan biaya AWS KMS
- Mengkonfigurasi izin AWS KMS
- AWS KMS kesalahan

Memperkirakan biaya AWS KMS

Untuk memprediksi biaya dan lebih memahami AWS tagihan Anda, Anda mungkin ingin tahu seberapa sering Amazon SNS menggunakan tagihan Anda AWS KMS key.

> **Note**
>
> Meskipun rumus berikut dapat memberi Anda gambaran yang sangat baik tentang biaya yang diharapkan, biaya sebenarnya mungkin lebih tinggi karena sifat Amazon yang didistribusikanSNS.

Untuk menghitung jumlah API permintaan (R) per topik, gunakan rumus berikut:

```
R = B / D * (2 * P)
```

B adalah periode penagihan (dalam detik).

Dadalah periode penggunaan kembali kunci data (dalam detik—Amazon SNS menggunakan kembali kunci data hingga 5 menit).

Padalah jumlah prinsipal penerbitan yang mengirim ke topik Amazon. SNS

Berikut ini adalah contoh perhitungan. Untuk informasi harga sebenranya, lihat Harga AWS Key Management Service.

Contoh 1: Menghitung jumlah AWS KMS API panggilan untuk 1 penerbit dan 1 topik

Contoh ini mengasumsikan sebagai berikut:

- Periode penagihan adalah 1-31 Januari (2.678.400 detik).
- Periode penggunaan kembali kunci data adalah 5 menit (300 detik).
- Ada 1 topik.
- Ada 1 penerbitan utama.

```
2,678,400 / 300 * (2 * 1) = 17,856
```

Contoh 2: Menghitung jumlah AWS KMS API panggilan untuk beberapa penerbit dan 2 topik

Contoh ini mengasumsikan sebagai berikut:

- Periode penagihan adalah 1-28 Februari (2.419.200 detik).
- Periode penggunaan kembali kunci data adalah 5 menit (300 detik).
- Ada 2 topik.
- Topik pertama memiliki 3 prinsipal penerbitan.
- Topik kedua memiliki 5 prinsipal penerbitan.

```
(2,419,200 / 300 * (2 * 3)) + (2,419,200 / 300 * (2 * 5)) = 129,024
```

Mengkonfigurasi izin AWS KMS

Sebelum Anda dapat menggunakanSSE, Anda harus mengonfigurasi AWS KMS key kebijakan untuk mengizinkan enkripsi topik dan enkripsi dan dekripsi pesan. Untuk contoh dan informasi selengkapnya tentang AWS KMS izin, lihat AWS KMS APIIzin: Tindakan dan Referensi Sumber Daya di Panduan AWS Key Management Service Pengembang. Untuk detail tentang cara menyiapkan SNS topik Amazon dengan enkripsi sisi server, lihat. Siapkan SNS topik Amazon dengan enkripsi sisi server

> **Note**
>
> Anda juga dapat mengelola izin untuk KMS kunci enkripsi simetris menggunakan IAM kebijakan. Untuk informasi selengkapnya, lihat Menggunakan IAM Kebijakan dengan AWS KMS.
> Meskipun Anda dapat mengonfigurasi izin global untuk mengirim dan menerima dari AmazonSNS, AWS KMS perlu secara eksplisit memberi nama penuh ARN KMSs di wilayah tertentu di `Resource` bagian kebijakan. IAM

Anda juga harus memastikan bahwa kebijakan utama AWS KMS key mengizinkan izin yang diperlukan. Untuk melakukan ini, beri nama prinsip yang memproduksi dan menggunakan pesan terenkripsi di SNS Amazon sebagai pengguna dalam kebijakan utama. KMS

Atau, Anda dapat menentukan AWS KMS tindakan yang diperlukan dan KMS ARN dalam IAM kebijakan yang ditetapkan ke kepala sekolah yang memublikasikan dan berlangganan untuk menerima pesan terenkripsi di Amazon. SNS Untuk informasi selengkapnya, lihat Mengelola Akses ke AWS KMS dalam Panduan AWS Key Management Service Pengembang.

Jika memilih kunci yang dikelola pelanggan untuk SNS topik Amazon Anda dan Anda menggunakan alias untuk mengontrol akses ke KMS kunci menggunakan IAM kebijakan atau kebijakan kunci dengan KMS kunci kondisi`kms:ResourceAliases`, pastikan bahwa kunci yang dikelola pelanggan yang dipilih juga memiliki alias yang terkait. Untuk informasi selengkapnya tentang penggunaan alias untuk mengontrol akses ke KMS kunci, lihat Menggunakan alias untuk mengontrol akses ke KMS kunci di Panduan AWS Key Management Service Pengembang.

Memungkinkan pengguna untuk mengirim pesan ke topik dengan SSE

Penerbit harus memiliki `kms:GenerateDataKey*` dan `kms:Decrypt` izin untuk. AWS KMS key

```
{
  "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"
  }]
}
```

Aktifkan kompatibilitas antara sumber acara dari AWS layanan dan topik terenkripsi

Beberapa AWS layanan mempublikasikan acara ke SNS topik Amazon. Untuk mengizinkan sumber peristiwa tersebut agar bekerja dengan topik terenkripsi, Anda harus melakukan langkah-langkah berikut in.

1. Gunakan kunci yang dikelola pelanggan. Untuk informasi selengkapnya, lihat Membuat Kunci di Panduan Developer AWS Key Management Service .
2. Untuk mengizinkan AWS layanan memiliki `kms:Decrypt` izin `kms:GenerateDataKey*` dan, tambahkan pernyataan berikut ke KMS kebijakan.

   ```
   {
   ```

```
  "Statement": [{
    "Effect": "Allow",
    "Principal": {
      "Service": "service.amazonaws.com"
    },
    "Action": [
      "kms:GenerateDataKey*",
      "kms:Decrypt"
    ],
    "Resource": "*"
  }]
}
```

Sumber peristiwa	Prinsipal layanan
Amazon CloudWatch	cloudwatch.amazonaws.com
CloudWatch Acara Amazon	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
Amazon Inspector	inspector.amazonaws.com
Amazon Redshift	redshift.amazonaws.com
Amazon RDS	events.rds.amazonaws.com
Amazon S3 Glacier	glacier.amazonaws.com
Layanan Email Amazon Sederhana	ses.amazonaws.com
Layanan Penyimpanan Sederhana Amazon	s3.amazonaws.com

Sumber peristiwa	Prinsipal layanan
AWS Snowball	`importexport.amazonaws.com`
AWS Manajer Insiden Systems Manager	AWS Systems Manager Incident Manager terdiri dari dua prinsip layanan: `ssm-incidents.amazonaws.com` ; `ssm-contacts.amazonaws.com`

> **Note**
>
> Beberapa sumber SNS acara Amazon mengharuskan Anda untuk memberikan IAM peran (bukan prinsipal layanan) dalam AWS KMS key kebijakan:
>
> - EC2Auto Scaling Amazon
> - Amazon Elastic Transcoder
> - AWS CodePipeline
> - AWS Config
> - AWS Elastic Beanstalk
> - AWS IoT
> - EC2Image Builder

3. Tambahkan kunci `aws:SourceAccount` dan `aws:SourceArn` kondisi ke kebijakan KMS sumber daya untuk lebih melindungi KMS kunci dari serangan wakil yang membingungkan. Lihat daftar dokumentasi khusus layanan (di atas) untuk detail yang tepat dalam setiap kasus.

> **Important**
>
> Menambahkan `aws:SourceAccount` dan `aws:SourceArn` ke AWS KMS kebijakan tidak didukung untuk EventBridge-to-encrypted topik.

```
{
  "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. SSEAktifkan topik Anda menggunakan topik AndaKMS.
5. Berikan topik ARN terenkripsi ke sumber acara.

AWS KMS kesalahan

Saat Anda bekerja dengan Amazon SNS dan AWS KMS, Anda mungkin mengalami kesalahan. Daftar berikut ini menjelaskan kesalahan dan solusi pemecahan masalah yang dimungkinkan.

KMSAccessDeniedException

Ciphertext merujuk pada kunci yang tidak ada atau bahwa Anda tidak memiliki akses ke padanya.

HTTPKode Status: 400

KMSDisabledException

Permintaan ditolak karena yang ditentukan KMS tidak diaktifkan.

HTTPKode Status: 400

KMSInvalidStateException

Permintaan ditolak karena keadaan sumber daya yang ditentukan tidak valid untuk permintaan ini. Untuk informasi selengkapnya, lihat Status kunci AWS KMS keys dalam Panduan AWS Key Management Service Pengembang.

HTTPKode Status: 400

KMSNotFoundException

Permintaan ditolak karena entitas atau sumber daya yang ditentukan tidak dapat ditemukan.

HTTPKode Status: 400

KMSOptInRequired

ID kunci AWS akses memerlukan langganan untuk layanan ini.

HTTPKode Status: 403

KMSThrottlingException

Permintaan ditolak karena throttling permintaan. Untuk informasi selengkapnya tentang pembatasan, lihat Kuota di Panduan Pengembang.AWS Key Management Service

HTTPKode Status: 400

Menyiapkan enkripsi SNS topik Amazon dengan enkripsi sisi server

Dengan enkripsi sisi server (SSE), Anda dapat menyimpan data sensitif dalam topik terenkripsi. SSEmelindungi konten pesan dalam SNS topik Amazon menggunakan kunci yang dikelola di AWS Key Management Service (AWS KMS). Untuk informasi selengkapnya tentang enkripsi sisi server dengan AmazonSNS, lihat. Mengamankan SNS data Amazon dengan enkripsi sisi server Untuk selengkapnya tentang membuat AWS KMS kunci, lihat Membuat kunci di Panduan AWS Key Management Service Pengembang.

> **Important**
>
> Semua permintaan ke topik dengan SSE diaktifkan harus menggunakan HTTPS dan Versi Tanda Tangan 4.

Aktifkan enkripsi sisi server (SSE) untuk topik Amazon SNS menggunakan AWS Management Console

1. Masuk ke SNSkonsol Amazon.
2. Pada panel navigasi, pilih Topik.
3. Pada halaman Topik, pilih topik dan pilih Tindakan, Edit.
4. Perluas bagian Enkripsi dan lakukan hal berikut ini:

a. Pilih Aktifkan enkripsi.

b. Tentukan AWS KMS kuncinya. Untuk informasi selengkapnya, lihat Istilah kunci.

Untuk setiap KMS jenis, Deskripsi, Akun, dan KMSARNditampilkan.

> **Important**
>
> Jika Anda bukan pemilikKMS, atau jika Anda masuk dengan akun yang tidak memiliki izin `kms:ListAliases` dan `kms:DescribeKey` izin, Anda tidak akan dapat melihat informasi tentang SNS konsol Amazon. KMS
> Minta pemilik KMS untuk memberi Anda izin ini. Untuk informasi selengkapnya, lihat AWS KMS APIIzin: Tindakan dan Referensi Sumber Daya di Panduan AWS Key Management Service Pengembang.

- AWS Dikelola KMS untuk Amazon SNS (Default) alias/aws/sns dipilih secara default.

  > **Note**
  >
  > Ingatlah hal-hal berikut ini:
  >
  > - Pertama kali Anda menggunakan AWS Management Console untuk menentukan yang AWS dikelola KMS untuk Amazon SNS untuk suatu topik, AWS KMS membuat yang AWS dikelola KMS untuk AmazonSNS.
  > - Atau, saat pertama kali Anda menggunakan `Publish` tindakan pada topik yang SSE diaktifkan, AWS KMS membuat yang AWS dikelola KMS untuk AmazonSNS.

- Untuk menggunakan kustom KMS dari AWS akun Anda, pilih bidang KMSkunci dan kemudian pilih kustom KMS dari daftar.

  > **Note**
  >
  > Untuk petunjuk cara membuat kustomKMSs, lihat Membuat Kunci di Panduan AWS Key Management Service Pengembang

- Untuk menggunakan kustom KMS ARN dari AWS akun Anda atau dari AWS akun lain, masukkan ke dalam bidang KMSkunci.

5. Pilih Simpan perubahan.

   SSEdiaktifkan untuk topik Anda dan ***MyTopic***halaman ditampilkan.

   Status Enkripsi topik, AWS Akun, Kunci master Pelanggan (CMK) CMKARN, dan Deskripsi ditampilkan di tab Enkripsi.

Siapkan SNS topik Amazon dengan enkripsi sisi server

Saat membuat KMS kunci Anda, gunakan kebijakan KMS kunci berikut:

```
{
    "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"
        }
    }
}
```

Dampak pada konsumen

Ketika SSE diaktifkan untuk SNS topik Amazon, proses mengkonsumsi pesan tetap tidak berubah untuk pelanggan. AWS mengelola proses enkripsi dan dekripsi menggunakan. KMS Oleh karena itu, pelanggan tidak perlu membuat perubahan apa pun pada pengaturan yang ada untuk menangani pesan terenkripsi. AWS memastikan bahwa pesan dienkripsi saat istirahat dan secara otomatis didekripsi sebelum dikirim ke pelanggan. Ini berarti bahwa pelanggan akan terus menerima dan memproses pesan seperti yang mereka lakukan sebelum enkripsi diaktifkan, tanpa memerlukan

konfigurasi tambahan atau logika dekripsi. Selain itu, AWS merekomendasikan penggunaan HTTPS untuk memastikan transmisi pesan yang aman.

Menyiapkan enkripsi SNS topik Amazon dengan langganan antrian Amazon SQS terenkripsi

Anda dapat mengaktifkan enkripsi sisi server (SSE) untuk topik untuk melindungi datanya. Untuk memungkinkan Amazon mengirim pesan SNS ke SQS antrian Amazon terenkripsi, kunci terkelola pelanggan yang terkait dengan SQS antrian Amazon harus memiliki pernyataan kebijakan yang memberikan akses prinsipal SNS layanan Amazon ke tindakan dan. AWS KMS API `GenerateDataKey` `Decrypt` Untuk informasi selengkapnya tentang penggunaanSSE, lihatMengamankan SNS data Amazon dengan enkripsi sisi server.

Halaman ini menunjukkan bagaimana Anda dapat mengaktifkan SSE SNS topik Amazon di mana SQS antrian Amazon terenkripsi berlangganan, menggunakan. AWS Management Console

Langkah 1: Buat KMS kunci khusus

1. Masuk ke konsol AWS KMS dengan pengguna yang memiliki setidaknya kebijakan `AWSKeyManagementServicePowerUser`.
2. Pilih Buat kunci.
3. Untuk membuat KMS kunci enkripsi simetris, untuk Key type pilih Symmetric.

   Untuk informasi tentang cara membuat KMS kunci asimetris di AWS KMS konsol, lihat Membuat KMS kunci asimetris (konsol).
4. Dalam Penggunaan kunci, opsi Enkripsi dan dekripsi dipilih untuk Anda.

   Untuk informasi tentang cara membuat KMS kunci yang menghasilkan dan memverifikasi MAC kode, lihat Membuat HMAC KMS kunci.

   Untuk informasi tentang opsi lanjutan, lihat Kunci tujuan khusus.
5. Pilih Berikutnya.
6. Ketik alias untuk KMS kunci. Nama alias tidak dapat dimulai dengan **`aws/`**. **`aws/`**Awalan dicadangkan oleh Amazon Web Services untuk mewakili Kunci yang dikelola AWS di akun Anda.

> **Note**
>
> Menambahkan, menghapus, atau memperbarui alias dapat mengizinkan atau menolak izin ke kunci. KMS Untuk detailnya, lihat ABAC AWS KMS dan Menggunakan alias untuk mengontrol akses ke KMS kunci.

Alias adalah nama tampilan yang dapat Anda gunakan untuk mengidentifikasi KMS kunci. Kami menyarankan Anda memilih alias yang menunjukkan jenis data yang Anda rencanakan untuk dilindungi atau aplikasi yang Anda rencanakan untuk digunakan dengan KMS kunci tersebut.

Alias diperlukan saat Anda membuat KMS kunci di file. AWS Management Console Mereka opsional saat Anda menggunakan CreateKeyoperasi.

7. (Opsional) Ketik deskripsi untuk KMS kunci.

   Anda dapat menambahkan deskripsi sekarang atau memperbaruinya kapan saja kecuali status kuncinya adalah `Pending Deletion` atau`Pending Replica Deletion`. Untuk menambah, mengubah, atau menghapus deskripsi kunci terkelola pelanggan yang ada, edit deskripsi di AWS Management Console atau gunakan UpdateKeyDescriptionoperasi.

8. (Opsional) Ketik kunci tanda dan nilai tanda opsional. Untuk menambahkan lebih dari satu tag ke KMS tombol, pilih Tambah tag.

   > **Note**
   >
   > Menandai atau melepas tag KMS kunci dapat mengizinkan atau menolak izin ke kunci. KMS Untuk detailnya, lihat ABAC AWS KMS dan Menggunakan tag untuk mengontrol akses ke KMS kunci.

   Saat menambahkan tag ke AWS sumber daya, buat AWS laporan alokasi biaya dengan penggunaan dan biaya yang dikumpulkan berdasarkan tag. Tag juga dapat digunakan untuk mengontrol akses ke KMS kunci. Untuk informasi tentang menandai KMS kunci, lihat Menandai kunci dan ABAC untuk. AWS KMS

9. Pilih Berikutnya.

10. Pilih IAM pengguna dan peran yang dapat mengelola KMS kunci.

> **Note**
>
> Kebijakan kunci ini memberikan kendali Akun AWS penuh atas KMS kunci ini. Ini memungkinkan administrator akun untuk menggunakan IAM kebijakan untuk memberikan izin kepada prinsipal lain untuk mengelola kunci. KMS Untuk detailnya, lihat Kebijakan kunci default.
>
> IAMPraktik terbaik mencegah penggunaan IAM pengguna dengan kredensi jangka panjang. Bila memungkinkan, gunakan IAM peran, yang menyediakan kredensi sementara. Untuk detailnya, lihat Praktik terbaik keamanan IAM di Panduan IAM Pengguna.

11. (Opsional) Untuk mencegah IAM pengguna dan peran yang dipilih menghapus KMS kunci ini, di bagian Penghapusan kunci di bagian bawah halaman, kosongkan kotak centang Izinkan administrator kunci untuk menghapus kunci ini.
12. Pilih Berikutnya.
13. Pilih IAM pengguna dan peran yang dapat menggunakan kunci dalam operasi kriptografi. Pilih Berikutnya.
14. Pada halaman Meninjau dan mengedit kebijakan kunci, tambahkan pernyataan berikut ini ke kebijakan kunci, dan kemudian pilih Selesai.

```
{
    "Sid": "Allow Amazon SNS to use this key",
    "Effect": "Allow",
    "Principal": {
        "Service": "sns.amazonaws.com"
    },
    "Action": [
        "kms:Decrypt",
        "kms:GenerateDataKey*"
    ],
    "Resource": "*"
}
```

Kunci terkelola pelanggan baru Anda muncul di daftar kunci.

Langkah 2: Buat topik Amazon terenkripsi SNS

1. Masuk ke SNSkonsol Amazon.
2. Pada panel navigasi, pilih Topik.
3. Pilih Buat topik.
4. Pada halaman Buat topik baru, untuk Nama, masukkan nama topik (sebagai contoh, `MyEncryptedTopic`) dan kemudian pilih Buat topik.
5. Perluas bagian Enkripsi dan lakukan hal berikut ini:

   a. Pilih Aktifkan enkripsi sisi server.

   b. Tentukan kunci yang dikelola pelanggan. Untuk informasi selengkapnya, lihat Istilah kunci.

      Untuk setiap jenis kunci yang dikelola pelanggan, kunci Deskripsi, Akun, dan yang dikelola pelanggan ARNditampilkan.

      > **Important**
      >
      > Jika Anda bukan pemilik kunci yang dikelola pelanggan, atau jika Anda masuk dengan akun yang tidak memiliki `kms:DescribeKey` izin `kms:ListAliases` dan, Anda tidak akan dapat melihat informasi tentang kunci yang dikelola pelanggan di SNS konsol Amazon.
      > Mintalah pemilik kunci yang dikelola pelanggan untuk memberi Anda izin ini. Untuk informasi selengkapnya, lihat AWS KMS APIIzin: Tindakan dan Referensi Sumber Daya di Panduan AWS Key Management Service Pengembang.

   c. Untuk kunci terkelola pelanggan, pilih MyCustomKeyyang Anda buat sebelumnya, lalu pilih Aktifkan enkripsi sisi server.
6. Pilih Simpan perubahan.

   SSEdiaktifkan untuk topik Anda dan MyTopichalaman ditampilkan.

   Status Enkripsi topik, AWS Akun, kunci terkelola pelanggan, kunci yang dikelola pelanggan ARN, dan Deskripsi ditampilkan di tab Enkripsi.

Topik terenkripsi baru Anda muncul dalam daftar topik.

Langkah 3: Buat dan berlangganan antrian Amazon terenkripsi SQS

1. Masuk ke SQSkonsol Amazon.
2. Pilih Buat Antrean Baru.
3. Pada halaman Buat Antrean Baru, lakukan hal berikut ini:
   a. Masukkan Nama Antrean (sebagai contoh, `MyEncryptedQueue1`).
   b. Pilih Antrean Standar, dan kemudian pilih Konfigurasi Antrean.
   c. Pilih Gunakan SSE.
   d. Untuk AWS KMS key, pilih MyCustomKeyyang Anda buat sebelumnya, lalu pilih Buat Antrian.
4. Ulangi proses untuk membuat antrean kedua (sebagai contoh, beri nama `MyEncryptedQueue2`).

   Antrean terenkripsi baru Anda muncul dalam daftar antrean.
5. Di SQS konsol Amazon, pilih `MyEncryptedQueue1` `MyEncryptedQueue2` dan kemudian pilih Tindakan Antrian, Berlangganan Antrian ke Topik. SNS
6. Dalam kotak dialog Subscribe to a Topic, untuk Pilih Topik pilih MyEncryptedTopic, lalu pilih Berlangganan.

   Langganan antrean terenkripsi Anda ke topik terenkripsi Anda ditampilkan di kotak dialog Hasil Berlangganan Topik.
7. Pilih OKE.

Langkah 4: Publikasikan pesan ke topik terenkripsi Anda

1. Masuk ke SNSkonsol Amazon.
2. Di panel navigasi, pilih Topik.
3. Dari daftar topik, pilih MyEncryptedTopiclalu pilih Publikasikan pesan.
4. Pada halaman Terbitkan pesan, lakukan hal berikut ini:
   a. (Opsional) Di bagian Detail pesan, masukkan Subjek (sebagai contoh, `Testing message publishing`).
   b. Di bagian Isi pesan, masukkan isi pesan (sebagai contoh, `My message body is encrypted at rest.`).

c. Pilih Terbitkan pesan.

Pesan Anda diterbitkan ke antrean terenkripsi berlangganan Anda.

Langkah 5: Verifikasi pengiriman pesan

1. Masuk ke SQSkonsol Amazon.
2. Dari daftar antrian, pilih MyEncryptedQueue1 lalu pilih Kirim dan terima pesan.
3. Pada halaman Kirim dan terima pesan dalam MyEncryptedQueue 1, pilih Poll untuk pesan.

   Pesan yang Anda kirim sebelumnya ditampilkan.
4. Pilih Detail Selengkapnya untuk melihat pesan Anda.
5. Setelah Anda selesai, pilih Tutup.
6. Ulangi proses ini untuk MyEncryptedQueue2.

Mengamankan SNS lalu lintas Amazon dengan titik akhir VPC

Titik akhir Amazon Virtual Private Cloud (AmazonVPC) untuk Amazon SNS adalah entitas logis dalam a VPC yang memungkinkan konektivitas hanya ke AmazonSNS. VPCRute meminta ke Amazon SNS dan merutekan tanggapan kembali keVPC. Bagian berikut memberikan informasi tentang bekerja dengan VPC titik akhir dan membuat kebijakan VPC titik akhir.

Jika Anda menggunakan Amazon Virtual Private Cloud (AmazonVPC) untuk meng-host AWS sumber daya Anda, Anda dapat membuat koneksi pribadi antara Anda VPC dan AmazonSNS. Dengan koneksi ini, Anda dapat mempublikasikan pesan ke SNS topik Amazon Anda tanpa mengirimnya melalui internet publik.

Amazon VPC adalah AWS layanan yang dapat Anda gunakan untuk meluncurkan AWS sumber daya di jaringan virtual yang Anda tentukan. DenganVPC, Anda memiliki kontrol atas pengaturan jaringan Anda, seperti rentang alamat IP, subnet, tabel rute, dan gateway jaringan. Untuk menghubungkan Anda VPC ke AmazonSNS, Anda menentukan VPCtitik akhir antarmuka. Jenis titik akhir ini memungkinkan Anda untuk menghubungkan Anda VPC ke AWS layanan. Titik akhir menyediakan konektivitas yang andal dan dapat diskalakan ke Amazon SNS tanpa memerlukan gateway internet, instance terjemahan alamat jaringan (NAT), atau VPN koneksi. Untuk informasi selengkapnya, lihat VPCTitik Akhir Antarmuka di Panduan VPC Pengguna Amazon.

Informasi di bagian ini adalah untuk pengguna AmazonVPC. Untuk informasi selengkapnya, dan untuk memulai membuatVPC, lihat Memulai Dengan Amazon VPC di Panduan VPC Pengguna Amazon.

> **Note**
>
> VPCendpoint tidak memungkinkan Anda untuk berlangganan SNS topik Amazon ke alamat IP pribadi.

Topik

- Membuat VPC titik akhir Amazon untuk Amazon SNS
- Membuat kebijakan VPC endpoint Amazon untuk Amazon SNS
- Menerbitkan SNS pesan Amazon dari Amazon VPC

Membuat VPC titik akhir Amazon untuk Amazon SNS

Untuk mempublikasikan pesan ke SNS topik Amazon Anda dari AmazonVPC, buat VPC titik akhir antarmuka. Kemudian, Anda dapat mempublikasikan pesan ke topik Anda sambil menjaga lalu lintas dalam jaringan yang Anda kelola denganVPC.

Gunakan informasi berikut untuk membuat titik akhir dan menguji koneksi antara Anda VPC dan AmazonSNS. Atau, untuk panduan yang membantu Anda memulai dari scratch, lihat Menerbitkan SNS pesan Amazon dari Amazon VPC.

Membuat titik akhir

Anda dapat membuat SNS endpoint Amazon di Anda VPC menggunakan AWS Management Console,, dan AWS SDK, Amazon SNSAPI, atau AWS CloudFormation. AWS CLI

Untuk informasi tentang membuat dan mengonfigurasi titik akhir menggunakan VPC konsol Amazon atau AWS CLI, lihat Membuat Titik Akhir Antarmuka di Panduan Pengguna Amazon VPC.

> **Important**
>
> Anda dapat menggunakan Amazon Virtual Private Cloud hanya dengan SNS endpoint HTTPS Amazon.

Saat Anda membuat titik akhir, tentukan Amazon SNS sebagai layanan yang VPC ingin Anda sambungkan. Di VPC konsol Amazon, nama layanan bervariasi berdasarkan wilayah. Sebagai contoh, jika Anda memilih US East (N. Virginia), nama layanan adalah com.amazonaws.us-east-1.sns.
Saat Anda mengonfigurasi Amazon SNS untuk mengirim pesan dari AmazonVPC, Anda harus mengaktifkan DNS privat dan menentukan titik akhir dalam formatsns.*us-east-2*.amazonaws.com.
Private DNS tidak mendukung titik akhir lama seperti queue.amazonaws.com atau. *us-east-2*.queue.amazonaws.com

Untuk informasi tentang membuat dan mengonfigurasi titik akhir menggunakan AWS CloudFormation, lihat AWS::EC2::VPCEndpointsumber daya di AWS CloudFormation Panduan Pengguna.

Menguji koneksi antara Anda VPC dan Amazon SNS

Setelah Anda membuat endpoint untuk AmazonSNS, Anda dapat mempublikasikan pesan dari SNS topik Amazon Anda VPC ke Anda. Untuk menguji koneksi ini, lakukan hal berikut ini:

1. Connect ke EC2 instans Amazon yang berada di instans AndaVPC. Untuk informasi tentang menghubungkan, lihat Menyambung ke Instans Linux Anda atau Menghubungkan ke Instans Windows Anda di EC2 dokumentasi Amazon.

   Misalnya, untuk terhubung ke instance Linux menggunakan SSH klien, jalankan perintah berikut dari terminal:

   ```
   $ ssh -i ec2-key-pair.pem ec2-user@instance-hostname
   ```

   Di mana:

   - ec2-key-pair.pem adalah file yang berisi key pair yang disediakan Amazon saat Anda membuat instance. EC2
   - instance-hostname adalah nama host publik dari instans. Untuk mendapatkan nama host di EC2konsol Amazon: Pilih Instans, pilih instance Anda, dan temukan nilai untuk Public DNS () IPv4.
2. Dari instans Anda, gunakan SNS publishperintah Amazon dengan file AWS CLI. Anda dapat mengirim pesan sederhana ke topik dengan perintah berikut ini:

```
$ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"
```

Di mana:

- aws-region adalah Wilayah tempat topik AWS tersebut berada.
- sns-topic-arnadalah Nama Sumber Daya Amazon (ARN) dari topik tersebut. Untuk mendapatkan ARN dari SNSkonsol Amazon: Pilih Topik, temukan topik Anda, dan temukan nilainya di ARNkolom.

Jika pesan berhasil diterima oleh AmazonSNS, terminal akan mencetak ID pesan, seperti berikut ini:

```
{
   "MessageId": "6c96dfff-0fdf-5b37-88d7-8cba910a8b64"
}
```

Membuat kebijakan VPC endpoint Amazon untuk Amazon SNS

Anda dapat membuat kebijakan untuk VPC titik akhir Amazon untuk Amazon SNS di mana Anda menentukan hal berikut:

- Prinsipal yang dapat melakukan tindakan.
- Tindakan yang dapat dilakukan.
- Sumber daya yang menjadi target tindakan.

Untuk informasi selengkapnya, lihat Mengontrol Akses ke Layanan dengan VPC Titik Akhir di Panduan VPC Pengguna Amazon.

Contoh kebijakan VPC endpoint berikut menentukan bahwa IAM pengguna `MyUser` diizinkan untuk mempublikasikan ke topik AmazonSNS. `MyTopic`

```
{
  "Statement": [{
    "Action": ["sns:Publish"],
    "Effect": "Allow",
    "Resource": "arn:aws:sns:us-east-2:123456789012:MyTopic",
```

```
    "Principal": {
      "AWS": "arn:aws:iam:123456789012:user/MyUser"
    }
  }]
}
```

Hal berikut ini ditolak:

- SNSAPITindakan Amazon lainnya, seperti `sns:Subscribe` dan`sns:Unsubscribe`.
- IAMPengguna dan aturan lain yang mencoba menggunakan VPC titik akhir ini.
- `MyUser`menerbitkan ke SNS topik Amazon yang berbeda.

> **Note**
>
> IAMPengguna masih dapat menggunakan SNS API tindakan Amazon lainnya dari luarVPC.

Menerbitkan SNS pesan Amazon dari Amazon VPC

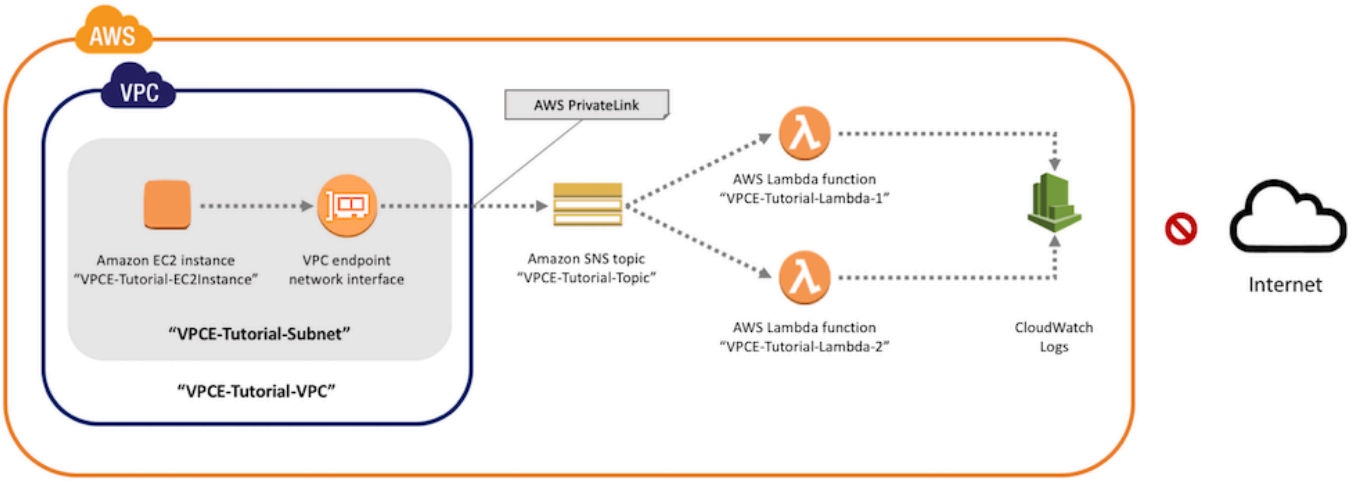
Bagian ini menjelaskan cara mempublikasikan ke SNS topik Amazon sambil menjaga pesan tetap aman di jaringan pribadi. Anda memublikasikan pesan dari EC2 instans Amazon yang di-host di Amazon Virtual Private Cloud (AmazonVPC). Pesan tetap berada dalam AWS jaringan tanpa bepergian ke internet publik. Dengan menerbitkan pesan secara pribadi dari aVPC, Anda dapat meningkatkan keamanan lalu lintas antara aplikasi Anda dan AmazonSNS. Keamanan ini penting ketika Anda mempublikasikan informasi identitas pribadi (PII) tentang pelanggan Anda, atau ketika aplikasi Anda tunduk pada peraturan pasar. Misalnya, menerbitkan secara pribadi sangat membantu jika Anda memiliki sistem perawatan kesehatan yang harus mematuhi Undang-Undang Portabilitas dan Akuntabilitas Asuransi Kesehatan (HIPAA), atau sistem keuangan yang harus mematuhi Standar Keamanan Data Industri Kartu Pembayaran (). PCI DSS

Langkah-langkah umumnya adalah sebagai berikut:

- Gunakan AWS CloudFormation template untuk secara otomatis membuat jaringan pribadi sementara di Anda Akun AWS.
- Buat VPC titik akhir yang menghubungkan VPC dengan AmazonSNS.
- Masuk ke EC2 instans Amazon dan publikasikan pesan secara pribadi ke SNS topik Amazon.
- Verifikasi bahwa pesan telah berhasil dikirimkan.

- Hapus sumber daya yang Anda buat selama proses ini sehingga tidak tetap ada di Anda Akun AWS.

Diagram berikut menggambarkan jaringan pribadi yang Anda buat di AWS akun Anda saat Anda menyelesaikan langkah-langkah ini:



Jaringan ini terdiri dari sebuah VPC yang berisi EC2 instance Amazon. Instans terhubung ke Amazon SNS melalui VPCtitik akhir antarmuka. Jenis titik akhir ini terhubung ke layanan yang didukung oleh AWS PrivateLink. Dengan koneksi ini dibuat, Anda dapat masuk ke EC2 instans Amazon dan mempublikasikan pesan ke SNS topik Amazon, meskipun jaringan terputus dari internet publik. Topik ini menggemari pesan yang diterimanya ke dua AWS Lambda fungsi berlangganan. Fungsi-fungsi ini mencatat pesan yang mereka terima di Amazon CloudWatch Logs.

Dibutuhkan sekitar 20 menit untuk menyelesaikan langkah-langkah tersebut.

Topik

- Sumber daya terkait

Sebelum Anda mulai

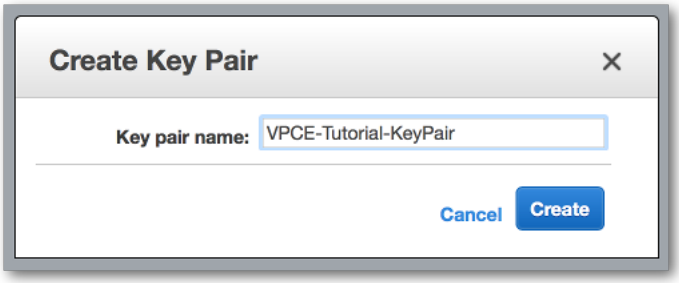
Sebelum Anda memulai, Anda memerlukan akun Amazon Web Services (AWS). Ketika Anda mendaftar, akun Anda secara otomatis mendaftar untuk semua layanan di AWS, termasuk Amazon SNS dan AmazonVPC. Jika Anda belum membuat akun, buka https://aws.amazon.com/, dan kemudian pilih Buat Akun Gratis.

Langkah 1: Buat EC2 key pair Amazon

Sebuah key pair digunakan untuk login ke EC2 instance Amazon. Pasangan kunci terdiri dari kunci publik yang digunakan untuk mengenkripsi informasi login Anda, dan kunci privat yang digunakan untuk mendekripsinya. Saat Anda membuat pasangan kunci, Anda harus mengunduh salinan kunci privat. Kemudian, Anda menggunakan key pair untuk masuk ke EC2 instance Amazon. Untuk login, Anda harus menentukan nama pasangan kunci, dan Anda harus memberikan kunci privat.

Untuk membuat pasangan kunci

1. Masuk ke AWS Management Console dan buka EC2 konsol Amazon di https://console.aws.amazon.com/ec2/.
2. Dalam menu navigasi di sebelah kiri, cari bagian Jaringan & Keamanan. Kemudian, pilih Pasangan Kunci.
3. Pilih Buat Pasangan Kunci.
4. Di jendela Buat Pasangan Kunci, untuk Nama pasangan kunci, ketik **`VPCE-Tutorial-KeyPair`**. Kemudian, pilih Buat.



5. File kunci privat tersebut akan secara otomatis diunduh oleh peramban Anda. Simpan di tempat yang aman. Amazon EC2 memberikan file ekstensi `.pem`.

6. (Opsional) Jika Anda menggunakan SSH klien di komputer Mac atau Linux untuk terhubung ke instans Anda, gunakan chmod perintah untuk mengatur izin file kunci pribadi Anda sehingga hanya Anda yang dapat membacanya:

   a. Buka terminal dan navigasikan ke direktori yang berisi kunci privat:

   ```
   $ cd /filepath_to_private_key/
   ```

   b. Mengatur izin menggunakan perintah berikut ini:

   ```
   $ chmod 400 VPCE-Tutorial-KeyPair.pem
   ```

Langkah 2: Buat AWS sumber daya

Untuk mengatur infrastruktur, Anda menggunakan AWS CloudFormation template. Template adalah file yang bertindak sebagai cetak biru untuk membangun AWS sumber daya, seperti EC2 contoh Amazon dan topik Amazon. SNS Template untuk proses ini disediakan GitHub untuk Anda unduh.

Anda menyediakan template untuk AWS CloudFormation, dan AWS CloudFormation menyediakan sumber daya yang Anda butuhkan sebagai tumpukan di Anda Akun AWS. Tumpukan adalah kumpulan sumber daya yang Anda kelola sebagai unit tunggal. Ketika Anda menyelesaikan langkah-langkah ini, Anda dapat menggunakan AWS CloudFormation untuk menghapus semua sumber daya dalam tumpukan sekaligus. Sumber daya ini tidak tetap ada di Anda Akun AWS, kecuali jika Anda menginginkannya.

Tumpukan untuk proses ini mencakup sumber daya berikut ini:

- A VPC dan sumber daya jaringan terkait, termasuk subnet, grup keamanan, gateway internet, dan tabel rute.
- EC2Instans Amazon yang diluncurkan ke subnet di. VPC
- SNSTopik Amazon.
- Dua AWS Lambda fungsi. Fungsi-fungsi ini menerima pesan yang dipublikasikan ke SNS topik Amazon, dan mereka mencatat peristiwa di CloudWatch Log.
- CloudWatch Metrik dan log Amazon.
- IAMPeran yang memungkinkan EC2 instans Amazon menggunakan AmazonSNS, dan IAM peran yang memungkinkan fungsi Lambda menulis ke CloudWatch log.

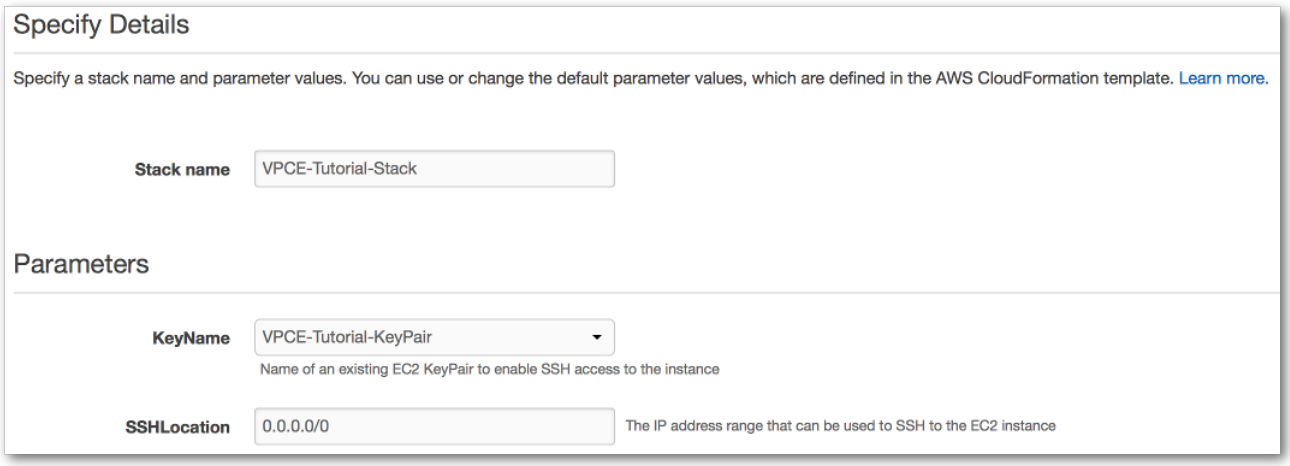
Untuk membuat sumber AWS daya

1. Unduh file template dari situs GitHub web.
2. Masuk ke konsol AWS CloudFormation tersebut.
3. Pilih Buat Tumpukan.
4. Pada halaman Pilih Templat, pilih Unggah templat ke Amazon S3, pilih file, dan pilih Berikutnya.
5. Pada halaman Tentukan Detail, tentukan nama tumpukan dan kunci:

   a. Untuk Nama tumpukan, ketik **`VPCE-Tutorial-Stack`**.

   b. Untuk KeyName, pilih VPCE-Tutorial- KeyPair.

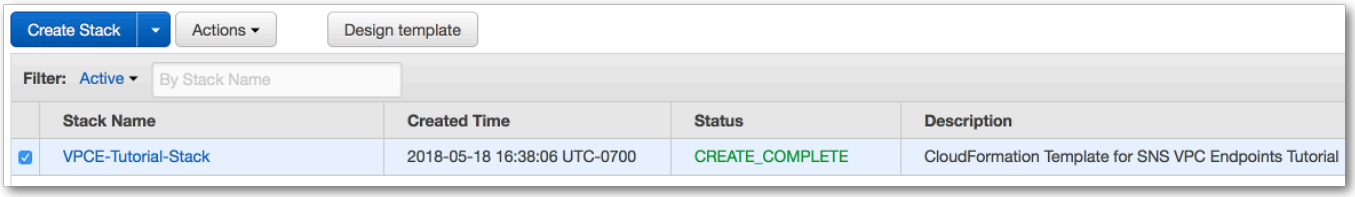
   c. Untuk SSHLocation, pertahankan nilai default**`0.0.0.0/0`**.



   d. Pilih Berikutnya.

6. Pada halaman Opsi, simpan semua nilai default, dan pilih Berikutnya.
7. Pada halaman Tinjau, verifikasi detail tumpukan.
8. Di bawah Kemampuan, akui bahwa AWS CloudFormation mungkin membuat IAM sumber daya dengan nama khusus.
9. Pilih Buat.

   AWS CloudFormation Konsol membuka halaman Stacks. VPCE-Tutorial-StackMemiliki status CREATE_IN_ PROGRESS. Dalam beberapa menit, setelah proses pembuatan selesai, status berubah menjadi CREATE_ COMPLETE.



> **Tip**
>
> Pilih tombol Segarkan untuk melihat status tumpukan terbaru.

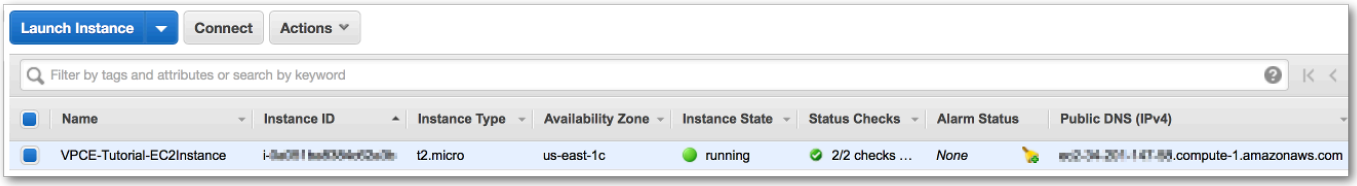
Langkah 3: Konfirmasikan bahwa EC2 instans Amazon Anda tidak memiliki akses internet

EC2Instans Amazon yang diluncurkan pada langkah sebelumnya tidak memiliki akses internet. VPC Ini melarang lalu lintas keluar, dan tidak dapat mempublikasikan pesan ke Amazon. SNS Verifikasikan ini dengan masuk ke instans. Kemudian, coba sambungkan ke titik akhir publik, dan coba kirim pesan ke AmazonSNS.

Pada titik ini, upaya menerbitkan gagal. Pada langkah selanjutnya, setelah Anda membuat VPC titik akhir untuk AmazonSNS, upaya publikasi Anda berhasil.

Untuk terhubung ke EC2 instans Amazon Anda

1. Buka EC2 konsol Amazon di https://console.aws.amazon.com/ec2/.
2. Dalam menu navigasi di sebelah kiri, cari bagian Instans. Kemudian, pilih Instans.
3. Dalam daftar contoh, pilih VPCE- Tutorial-EC2Instance.
4. Salin nama host yang disediakan di kolom Public DNS (IPv4).



5. Buka terminal. Dari direktori yang berisi key pair, sambungkan ke instance menggunakan perintah berikut, di mana *instance-hostname* adalah nama host yang Anda salin dari konsol AmazonEC2:

```
$ ssh -i VPCE-Tutorial-KeyPair.pem ec2-user@instance-hostname
```

Untuk memverifikasi bahwa instans tidak memiliki konektivitas internet

- Di terminal Anda, upayakan untuk menghubungkan ke titik akhir publik, seperti amazon.com:

  ```
  $ ping amazon.com
  ```

  Karena upaya koneksi gagal, Anda dapat membatalkan kapan saja (Ctrl + C pada Windows atau Command + C pada macOS).

Untuk memverifikasi bahwa instance tidak memiliki konektivitas ke Amazon SNS

1. Masuk ke SNSkonsol Amazon.
2. Di panel navigasi di sebelah kiri, pilih Topik.
3. Pada halaman Topik, salin Amazon Resource Name (ARN) untuk topik VPCE-Tutorial-Topic.
4. Di terminal Anda, upayakan untuk menerbitkan pesan ke topik:

   ```
   $ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"
   ```

   Karena upaya menerbitkan gagal, Anda dapat membatalkan setiap saat.

Langkah 4: Buat VPC titik akhir Amazon untuk Amazon SNS

Untuk menghubungkan VPC ke AmazonSNS, Anda menentukan VPC titik akhir antarmuka. Setelah menambahkan titik akhir, Anda dapat masuk ke EC2 instans Amazon di AndaVPC, dan dari sana Anda dapat menggunakan Amazon SNSAPI. Anda dapat menerbitkan pesan ke topik, dan pesan diterbitkan secara privat. Mereka tetap berada dalam AWS jaringan, dan mereka tidak melakukan perjalanan internet publik.

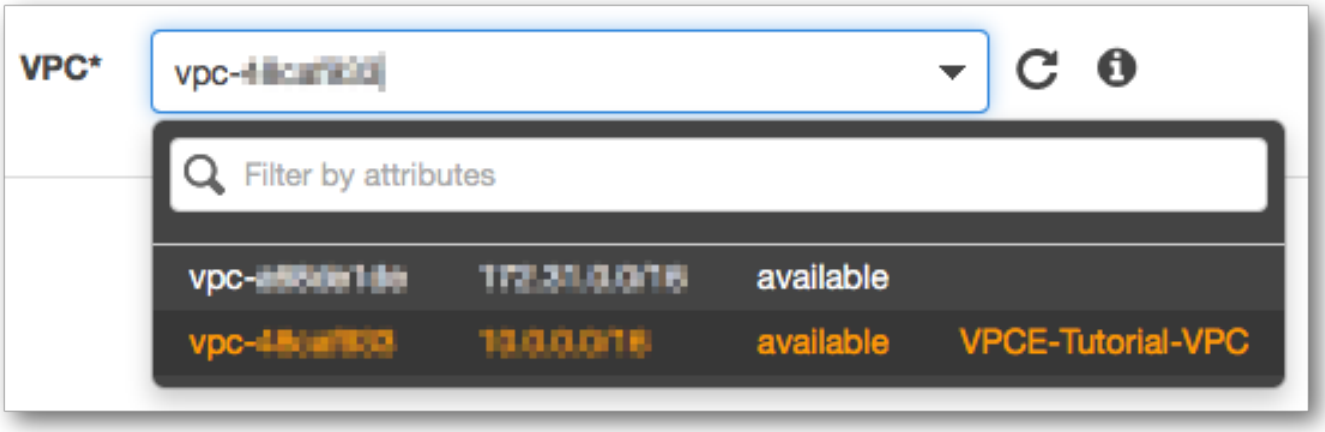
> **Note**
>
> Instans masih kekurangan akses ke AWS layanan lain dan titik akhir di internet.

Untuk membuat titik akhir

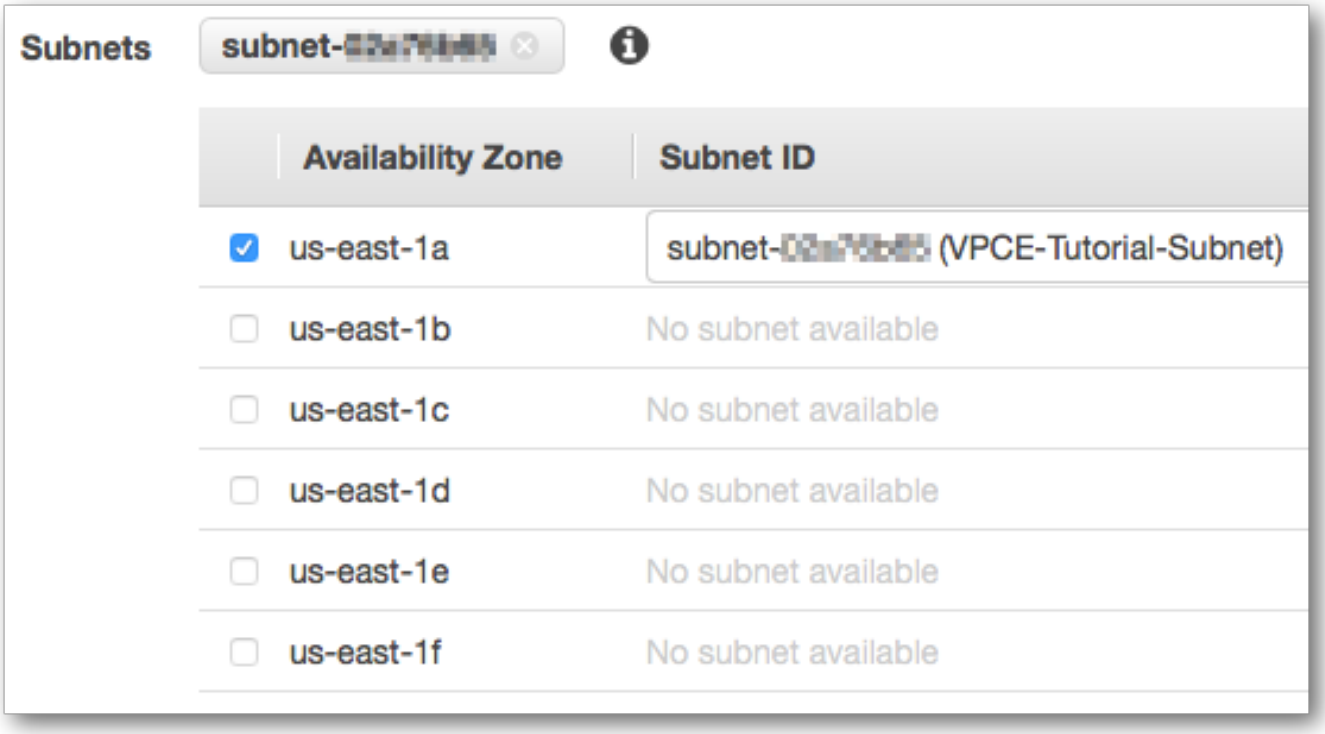
1. Buka VPC konsol Amazon di https://console.aws.amazon.com/vpc/.
2. Di menu navigasi di sebelah kiri, pilih Titik Akhir.

3. Pilih Buat Titik Akhir.
4. Pada halaman Buat Titik Akhir, untuk Kategori layanan, pilih layanan AWS .
5. Untuk Nama Layanan, pilih nama layanan untuk AmazonSNS.

   Nama layanan bervariasi berdasarkan wilayah yang dipilih. Misalnya, jika Anda memilih US East (Virginia N.), nama layanannya adalah com.amazonaws.***us-east-1***.sns.
6. Untuk VPC, pilih VPC yang memiliki nama VPCE-Tutorial- VPC.

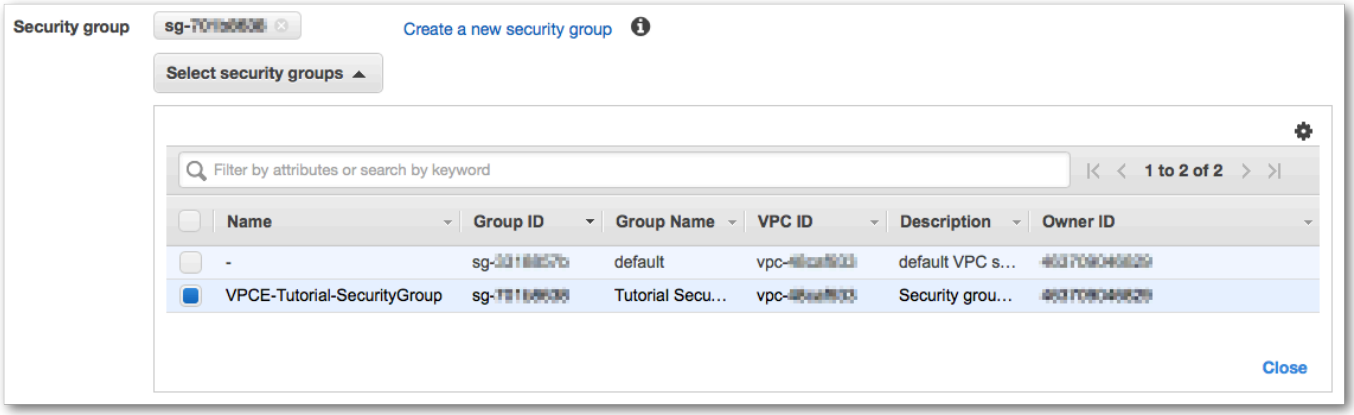


7. Untuk Subnet, pilih subnet yang memiliki VPCE-Tutorial-Subnet di subnet ID.

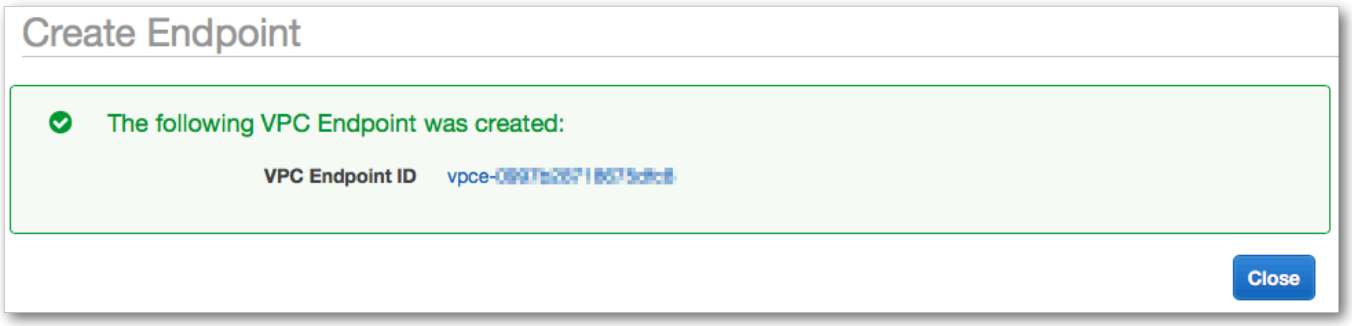


8. Untuk Aktifkan DNS Nama Pribadi, pilih Aktifkan untuk titik akhir ini.

9. Untuk grup Keamanan, pilih Pilih grup keamanan, dan pilih VPCE-Tutorial- SecurityGroup.

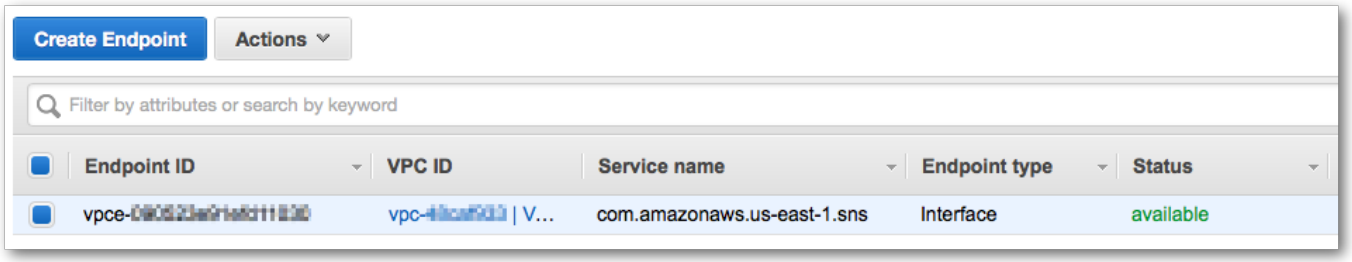


10. Pilih Buat titik akhir. VPCKonsol Amazon mengonfirmasi bahwa VPC titik akhir telah dibuat.



11. Pilih Tutup.

    VPCKonsol Amazon membuka halaman Endpoints. Titik akhir baru memiliki status tertunda. Dalam beberapa menit, setelah proses pembuatan selesai, status berubah menjadi tersedia.



Langkah 5: Publikasikan pesan ke SNS topik Amazon Anda

Sekarang setelah Anda VPC menyertakan titik akhir untuk AmazonSNS, Anda dapat masuk ke EC2 instans Amazon dan mempublikasikan pesan ke topik tersebut.

Untuk menerbitkan pesan

1. Jika terminal Anda tidak lagi terhubung ke EC2 instans Amazon Anda, sambungkan lagi:

```
$ ssh -i VPCE-Tutorial-KeyPair.pem ec2-user@instance-hostname
```

2. Jalankan perintah yang sama yang Anda lakukan sebelumnya untuk mempublikasikan pesan ke SNS topik Amazon Anda. Kali ini, upaya publikasi berhasil, dan Amazon SNS mengembalikan ID pesan:

```
$ aws sns publish --region aws-region --topic-arn sns-topic-arn --message "Hello"


{
    "MessageId": "5b111270-d169-5be6-9042-410dfc9e86de"
}
```

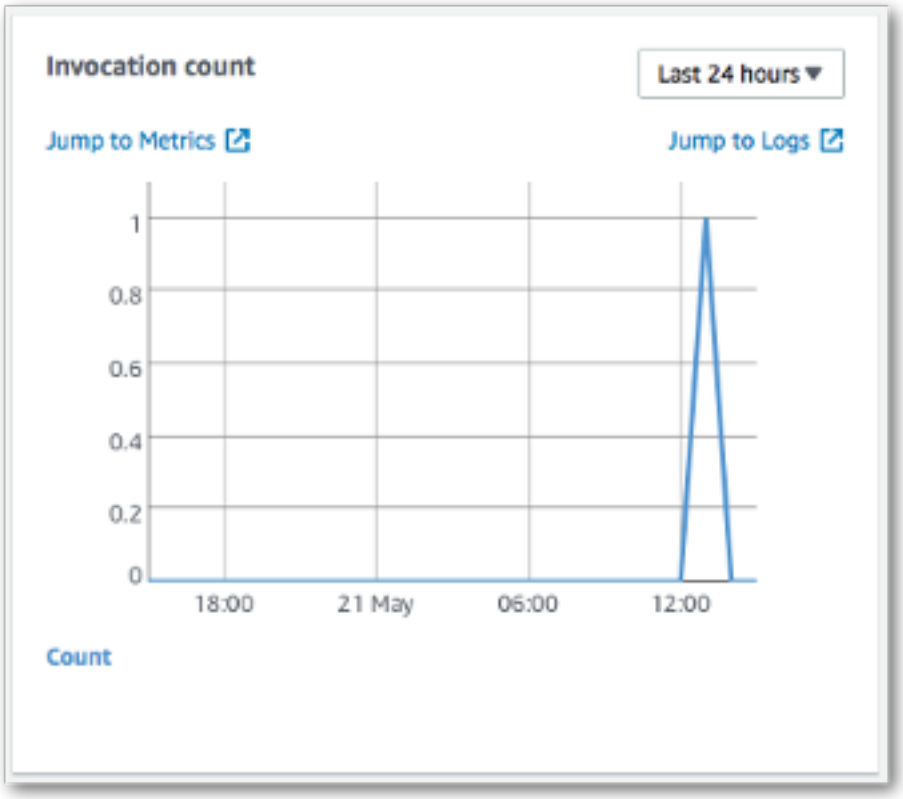
Langkah 6: Verifikasi pengiriman pesan Anda

Saat SNS topik Amazon menerima pesan, ia menggemari pesan tersebut dengan mengirimkannya ke dua fungsi Lambda berlangganan. Ketika fungsi-fungsi ini menerima pesan, mereka mencatat peristiwa ke CloudWatch log. Untuk memverifikasi bahwa pengiriman pesan Anda berhasil, periksa apakah fungsi telah dipanggil, dan periksa apakah CloudWatch log telah diperbarui.

Untuk memverifikasi bahwa fungsi Lambda dipanggil

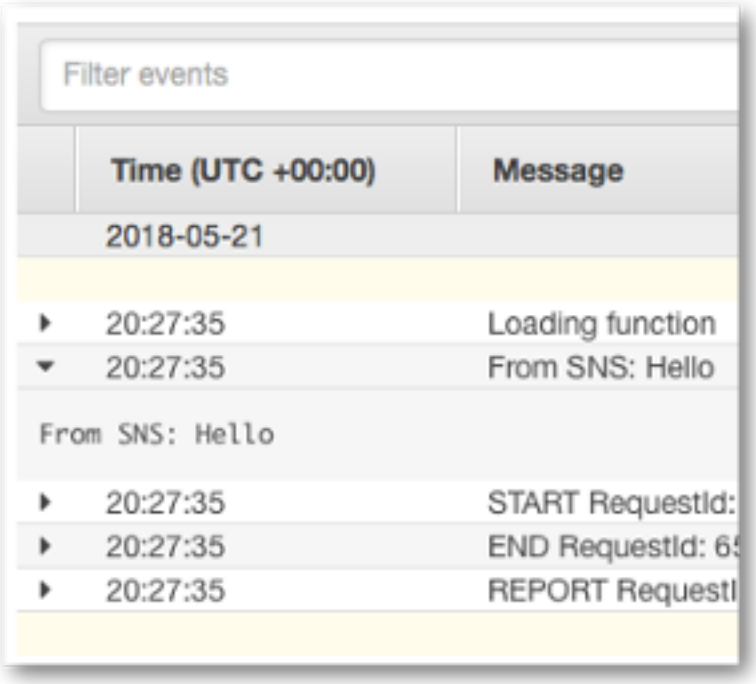
1. Buka AWS Lambda konsol di https://console.aws.amazon.com/lambda/.
2. Pada halaman Fungsi, pilih VPCE-Tutorial-Lambda-1.
3. Pilih Pemantauan.
4. Periksa grafik Hitungan invokasi. Grafik ini menunjukkan jumlah waktu saat fungsi Lambda telah dijalankan.

   Jumlah invokasi cocok dengan jumlah waktu saat Anda menerbitkan pesan ke topik.



Untuk memverifikasi bahwa CloudWatch log telah diperbarui

1. Buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Di panel navigasi di sebelah kiri, pilih Log.
3. Periksa log yang ditulis oleh fungsi Lambda:
    a. Pilih grup log/aws/lambda/VPCE-Tutorial-Lambda-1/.
    b. Pilih pengaliran log.
    c. Periksa bahwa log mencakup entri `From SNS: Hello`.



d. Pilih Grup Log pada bagian atas konsol untuk kembali ke halaman Grup Log. Kemudian, ulangi langkah-langkah sebelumnya untuk the /aws/lambda/VPCE -Tutorial-Lambda-2/log group.

Selamat! Dengan menambahkan titik akhir untuk Amazon SNS ke aVPC, Anda dapat mempublikasikan pesan ke topik dari dalam jaringan yang dikelola oleh. VPC Pesan diterbitkan secara privat tanpa dipaparkan ke internet publik.

Langkah 7: Membersihkan

Kecuali Anda ingin mempertahankan sumber daya yang Anda buat, Anda dapat menghapusnya sekarang. Dengan menghapus AWS sumber daya yang tidak lagi Anda gunakan, Anda mencegah tagihan yang tidak perlu ke Anda Akun AWS.

Pertama, hapus VPC titik akhir Anda menggunakan VPC konsol Amazon. Kemudian, hapus sumber daya lain yang Anda buat dengan menghapus tumpukan di AWS CloudFormation konsol. Saat Anda menghapus tumpukan, AWS CloudFormation hapus sumber daya tumpukan dari tumpukan Anda Akun AWS.

Untuk menghapus titik VPC akhir Anda

1. Buka VPC konsol Amazon di https://console.aws.amazon.com/vpc/.

2. Di menu navigasi di sebelah kiri, pilih Titik Akhir.
3. Pilih titik akhir yang Anda buat.
4. Pilih Tindakan, dan kemudian pilih Hapus Titik Akhir.
5. Di jendela Hapus Titik Akhir, pilih Ya, Hapus.

   Status titik akhir ubah ke menghapus. Setelah penghapusan selesai, titik akhir dihapus dari halaman.

Untuk menghapus AWS CloudFormation tumpukan Anda

1. Buka AWS CloudFormation konsol di https://console.aws.amazon.com/cloudformation.
2. Pilih tumpukan VPCE-Tutorial-Stack.
3. Pilih Tindakan, dan kemudian pilih Hapus Tumpukan.
4. Di jendela Hapus Tumpukan, pilih Ya, Hapus.

   Status tumpukan berubah menjadi DELETE_IN_ PROGRESS. Saat penghapusan selesai, tumpukan dihapus dari halaman.

Sumber daya terkait

Untuk informasi selengkapnya, lihat sumber daya berikut ini.

- AWS Blog Keamanan: Mengamankan pesan yang dipublikasikan ke Amazon SNS dengan AWS PrivateLink
- Apa itu AmazonVPC?
- VPCTitik akhir
- Apa itu AmazonEC2?
- AWS CloudFormation Konsep

Meningkatkan SNS keamanan Amazon dengan Perlindungan Data Pesan

- Perlindungan Data Pesan adalah fitur di Amazon yang SNS digunakan untuk menentukan aturan dan kebijakan Anda sendiri untuk mengaudit dan mengontrol konten untuk data yang bergerak, sebagai lawan dari data saat istirahat.

- Perlindungan Data Pesan menyediakan layanan tata kelola, kepatuhan, dan audit untuk aplikasi perusahaan yang berpusat pada pesan, sehingga masuknya dan keluar data dapat dikontrol oleh pemilik topik SNS Amazon, dan alur konten dapat dilacak dan dicatat.
- Anda dapat menulis aturan tata kelola berbasis payload untuk menghentikan konten payload yang tidak sah memasuki aliran pesan Anda.
- Anda dapat memberikan izin akses konten yang berbeda kepada pelanggan individu, dan mengaudit seluruh proses alur konten.

Manajemen identitas dan akses di Amazon SNS

Akses ke Amazon SNS memerlukan kredensi yang AWS dapat digunakan untuk mengautentikasi permintaan Anda. Kredensyal ini harus memiliki izin untuk mengakses AWS sumber daya, seperti SNS topik dan pesan Amazon. Bagian berikut memberikan detail tentang bagaimana Anda dapat menggunakan AWS Identity and Access Management (IAM) dan Amazon SNS untuk membantu mengamankan sumber daya Anda dengan mengontrol akses ke sumber daya tersebut.

AWS Identity and Access Management (IAM) adalah Layanan AWS yang membantu administrator mengontrol akses ke AWS sumber daya dengan aman. IAMadministrator mengontrol siapa yang dapat diautentikasi (masuk) dan diberi wewenang (memiliki izin) untuk menggunakan sumber daya Amazon. SNS IAMadalah Layanan AWS yang dapat Anda gunakan tanpa biaya tambahan.

Audiens

Cara Anda menggunakan AWS Identity and Access Management (IAM) berbeda, tergantung pada pekerjaan yang Anda lakukan di AmazonSNS.

Pengguna layanan — Jika Anda menggunakan SNS layanan Amazon untuk melakukan pekerjaan Anda, administrator Anda memberi Anda kredensyal dan izin yang Anda butuhkan. Saat Anda menggunakan lebih banyak SNS fitur Amazon untuk melakukan pekerjaan Anda, Anda mungkin memerlukan izin tambahan. Memahami cara akses dikelola dapat membantu Anda meminta izin yang tepat dari administrator Anda. Jika Anda tidak dapat mengakses fitur di AmazonSNS, lihatMemecahkan masalah identitas dan akses Amazon Simple Notification Service.

Administrator layanan - Jika Anda bertanggung jawab atas sumber SNS daya Amazon di perusahaan Anda, Anda mungkin memiliki akses penuh ke AmazonSNS. Tugas Anda adalah menentukan SNS fitur dan sumber daya Amazon mana yang harus diakses pengguna layanan Anda. Anda kemudian harus mengirimkan permintaan ke IAM administrator Anda untuk mengubah izin pengguna layanan

Anda. Tinjau informasi di halaman ini untuk memahami konsep dasarIAM. Untuk mempelajari lebih lanjut tentang bagaimana perusahaan Anda dapat menggunakan IAM AmazonSNS, lihatBagaimana Amazon SNS bekerja dengan IAM.

IAMadministrator - Jika Anda seorang IAM administrator, Anda mungkin ingin mempelajari detail tentang cara menulis kebijakan untuk mengelola akses ke AmazonSNS. Untuk melihat contoh kebijakan SNS berbasis identitas Amazon yang dapat Anda gunakan, lihat. IAM Contoh kebijakan berbasis identitas untuk Amazon Simple Notification Service

Mengautentikasi dengan identitas

Otentikasi adalah cara Anda masuk AWS menggunakan kredensi identitas Anda. Anda harus diautentikasi (masuk ke AWS) sebagai pengguna Akun AWS root, sebagai IAM pengguna, atau dengan mengambil peranIAM.

Anda dapat masuk AWS sebagai identitas federasi dengan menggunakan kredensil yang disediakan melalui sumber identitas. AWS IAM Identity Center Pengguna (Pusat IAM Identitas), autentikasi masuk tunggal perusahaan Anda, dan kredensyal Google atau Facebook Anda adalah contoh identitas federasi. Saat Anda masuk sebagai identitas federasi, administrator Anda sebelumnya menyiapkan federasi identitas menggunakan IAM peran. Ketika Anda mengakses AWS dengan menggunakan federasi, Anda secara tidak langsung mengambil peran.

Bergantung pada jenis pengguna Anda, Anda dapat masuk ke AWS Management Console atau portal AWS akses. Untuk informasi selengkapnya tentang masuk AWS, lihat Cara masuk ke Panduan AWS Sign-In Pengguna Anda Akun AWS.

Jika Anda mengakses AWS secara terprogram, AWS sediakan kit pengembangan perangkat lunak (SDK) dan antarmuka baris perintah (CLI) untuk menandatangani permintaan Anda secara kriptografis dengan menggunakan kredensil Anda. Jika Anda tidak menggunakan AWS alat, Anda harus menandatangani permintaan sendiri. Untuk informasi selengkapnya tentang menggunakan metode yang disarankan untuk menandatangani permintaan sendiri, lihat Versi AWS Tanda Tangan 4 untuk API permintaan di Panduan IAM Pengguna.

Apa pun metode autentikasi yang digunakan, Anda mungkin diminta untuk menyediakan informasi keamanan tambahan. Misalnya, AWS merekomendasikan agar Anda menggunakan otentikasi multi-faktor (MFA) untuk meningkatkan keamanan akun Anda. Untuk mempelajari selengkapnya, lihat Autentikasi multi-faktor di Panduan AWS IAM Identity Center Pengguna dan Autentikasi AWS multi-faktor IAM di Panduan Pengguna. IAM

Akun AWS pengguna root

Saat Anda membuat Akun AWS, Anda mulai dengan satu identitas masuk yang memiliki akses lengkap ke semua Layanan AWS dan sumber daya di akun. Identitas ini disebut pengguna Akun AWS root dan diakses dengan masuk dengan alamat email dan kata sandi yang Anda gunakan untuk membuat akun. Kami sangat menyarankan agar Anda tidak menggunakan pengguna root untuk tugas sehari-hari. Lindungi kredensial pengguna root Anda dan gunakan kredensial tersebut untuk melakukan tugas yang hanya dapat dilakukan pengguna root. Untuk daftar lengkap tugas yang mengharuskan Anda masuk sebagai pengguna root, lihat Tugas yang memerlukan kredensyal pengguna root di IAMPanduan Pengguna.

Identitas gabungan

Sebagai praktik terbaik, mewajibkan pengguna manusia, termasuk pengguna yang memerlukan akses administrator, untuk menggunakan federasi dengan penyedia identitas untuk mengakses Layanan AWS dengan menggunakan kredensyal sementara.

Identitas federasi adalah pengguna dari direktori pengguna perusahaan Anda, penyedia identitas web, direktori Pusat Identitas AWS Directory Service, atau pengguna mana pun yang mengakses Layanan AWS dengan menggunakan kredensyal yang disediakan melalui sumber identitas. Ketika identitas federasi mengakses Akun AWS, mereka mengambil peran, dan peran memberikan kredensi sementara.

Untuk manajemen akses terpusat, kami sarankan Anda menggunakan AWS IAM Identity Center. Anda dapat membuat pengguna dan grup di Pusat IAM Identitas, atau Anda dapat menghubungkan dan menyinkronkan ke sekumpulan pengguna dan grup di sumber identitas Anda sendiri untuk digunakan di semua aplikasi Akun AWS dan aplikasi Anda. Untuk informasi tentang Pusat IAM Identitas, lihat Apa itu Pusat IAM Identitas? dalam AWS IAM Identity Center User Guide.

Pengguna dan grup IAM

IAMPengguna adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus untuk satu orang atau aplikasi. Jika memungkinkan, sebaiknya mengandalkan kredensi sementara daripada membuat IAM pengguna yang memiliki kredensi jangka panjang seperti kata sandi dan kunci akses. Namun, jika Anda memiliki kasus penggunaan khusus yang memerlukan kredensi jangka panjang dengan IAM pengguna, kami sarankan Anda memutar kunci akses. Untuk informasi selengkapnya, lihat Memutar kunci akses secara teratur untuk kasus penggunaan yang memerlukan kredensi jangka panjang di IAMPanduan Pengguna.

IAMGrup adalah identitas yang menentukan kumpulan IAM pengguna. Anda tidak dapat masuk sebagai grup. Anda dapat menggunakan grup untuk menentukan izin bagi beberapa pengguna sekaligus. Grup mempermudah manajemen izin untuk sejumlah besar pengguna sekaligus. Misalnya, Anda dapat memiliki grup bernama IAMAdminsdan memberikan izin grup tersebut untuk mengelola sumber dayaIAM.

Pengguna berbeda dari peran. Pengguna secara unik terkait dengan satu orang atau aplikasi, tetapi peran dimaksudkan untuk dapat digunakan oleh siapa pun yang membutuhkannya. Pengguna memiliki kredensial jangka panjang permanen, tetapi peran memberikan kredensial sementara. Untuk mempelajari selengkapnya, lihat Kasus penggunaan untuk IAM pengguna di Panduan IAM Pengguna.

IAMperan

IAMPeran adalah identitas dalam diri Anda Akun AWS yang memiliki izin khusus. Ini mirip dengan IAM pengguna, tetapi tidak terkait dengan orang tertentu. Untuk mengambil IAM peran sementara di dalam AWS Management Console, Anda dapat beralih dari pengguna ke IAM peran (konsol). Anda dapat mengambil peran dengan memanggil AWS CLI atau AWS API operasi atau dengan menggunakan kustomURL. Untuk informasi selengkapnya tentang metode penggunaan peran, lihat Metode untuk mengambil peran dalam Panduan IAM Pengguna.

IAMperan dengan kredensi sementara berguna dalam situasi berikut:

- Akses pengguna terfederasi – Untuk menetapkan izin ke identitas terfederasi, Anda membuat peran dan menentukan izin untuk peran tersebut. Ketika identitas terfederasi mengautentikasi, identitas tersebut terhubung dengan peran dan diberi izin yang ditentukan oleh peran. Untuk informasi tentang peran untuk federasi, lihat Membuat peran untuk Penyedia Identitas pihak ketiga di Panduan IAM Pengguna. Jika Anda menggunakan Pusat IAM Identitas, Anda mengonfigurasi set izin. Untuk mengontrol apa yang dapat diakses identitas Anda setelah diautentikasi, Pusat IAM Identitas mengkorelasikan izin yang disetel ke peran. IAM Untuk informasi tentang set izin, lihat Set izin dalam Panduan Pengguna AWS IAM Identity Center .
- Izin IAM pengguna sementara — IAM Pengguna atau peran dapat mengambil IAM peran untuk sementara mengambil izin yang berbeda untuk tugas tertentu.
- Akses lintas akun — Anda dapat menggunakan IAM peran untuk memungkinkan seseorang (prinsipal tepercaya) di akun lain mengakses sumber daya di akun Anda. Peran adalah cara utama untuk memberikan akses lintas akun. Namun, dengan beberapa Layanan AWS, Anda dapat melampirkan kebijakan secara langsung ke sumber daya (alih-alih menggunakan peran sebagai

proxy). Untuk mempelajari perbedaan antara peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Akses sumber daya lintas akun di IAM Panduan Pengguna. IAM

- Akses lintas layanan — Beberapa Layanan AWS menggunakan fitur lain Layanan AWS. Misalnya, saat Anda melakukan panggilan dalam suatu layanan, biasanya layanan tersebut menjalankan aplikasi di Amazon EC2 atau menyimpan objek di Amazon S3. Sebuah layanan mungkin melakukannya menggunakan izin prinsipal yang memanggil, menggunakan peran layanan, atau peran terkait layanan.
  - Sesi akses teruskan (FAS) — Saat Anda menggunakan IAM pengguna atau peran untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan sebuah tindakan yang kemudian menginisiasi tindakan lain di layanan yang berbeda. FASmenggunakan izin dari pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. FASPermintaan hanya dibuat ketika layanan menerima permintaan yang memerlukan interaksi dengan orang lain Layanan AWS atau sumber daya untuk menyelesaikannya. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan saat membuat FAS permintaan, lihat Meneruskan sesi akses.
  - Peran layanan — Peran layanan adalah IAMperan yang diasumsikan layanan untuk melakukan tindakan atas nama Anda. IAMAdministrator dapat membuat, memodifikasi, dan menghapus peran layanan dari dalamIAM. Untuk informasi selengkapnya, lihat Membuat peran untuk mendelegasikan izin ke Layanan AWS dalam IAMPanduan Pengguna.
  - Peran terkait layanan — Peran terkait layanan adalah jenis peran layanan yang ditautkan ke peran layanan. Layanan AWS Layanan tersebut dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. IAMAdministrator dapat melihat, tetapi tidak mengedit izin untuk peran terkait layanan.
- Aplikasi yang berjalan di Amazon EC2 — Anda dapat menggunakan IAM peran untuk mengelola kredensyal sementara untuk aplikasi yang berjalan pada EC2 instance dan pembuatan AWS CLI atau AWS API permintaan. Ini lebih baik untuk menyimpan kunci akses dalam EC2 instance. Untuk menetapkan AWS peran ke EC2 instance dan membuatnya tersedia untuk semua aplikasinya, Anda membuat profil instance yang dilampirkan ke instance. Profil instance berisi peran dan memungkinkan program yang berjalan pada EC2 instance untuk mendapatkan kredensi sementara. Untuk informasi selengkapnya, lihat Menggunakan IAM peran untuk memberikan izin ke aplikasi yang berjalan di EC2 instans Amazon di IAMPanduan Pengguna.

Mengelola akses menggunakan kebijakan

Anda mengontrol akses AWS dengan membuat kebijakan dan melampirkannya ke AWS identitas atau sumber daya. Kebijakan adalah objek AWS yang, ketika dikaitkan dengan identitas atau sumber daya, menentukan izinnya. AWS mengevaluasi kebijakan ini ketika prinsipal (pengguna, pengguna root, atau sesi peran) membuat permintaan. Izin dalam kebijakan menentukan apakah permintaan diizinkan atau ditolak. Sebagian besar kebijakan disimpan AWS sebagai JSON dokumen. Untuk informasi selengkapnya tentang struktur dan isi dokumen JSON kebijakan, lihat Ringkasan JSON kebijakan di Panduan IAM Pengguna.

Administrator dapat menggunakan AWS JSON kebijakan untuk menentukan siapa yang memiliki akses ke apa. Yaitu, principal dapat melakukan tindakan pada suatu sumber daya, dan dalam suatu syarat.

Secara default, pengguna dan peran tidak memiliki izin. Untuk memberikan izin kepada pengguna untuk melakukan tindakan pada sumber daya yang mereka butuhkan, IAM administrator dapat membuat IAM kebijakan. Administrator kemudian dapat menambahkan IAM kebijakan ke peran, dan pengguna dapat mengambil peran.

IAMkebijakan menentukan izin untuk tindakan terlepas dari metode yang Anda gunakan untuk melakukan operasi. Misalnya, anggaplah Anda memiliki kebijakan yang mengizinkan tindakan `iam:GetRole`. Pengguna dengan kebijakan itu bisa mendapatkan informasi peran dari AWS Management Console, AWS CLI, atau AWS API.

Kebijakan berbasis identitas

Kebijakan berbasis identitas adalah dokumen kebijakan JSON izin yang dapat Anda lampirkan ke identitas, seperti pengguna, grup IAM pengguna, atau peran. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana, dan berdasarkan kondisi seperti apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Menentukan IAM izin khusus dengan kebijakan yang dikelola pelanggan di Panduan Pengguna. IAM

Kebijakan berbasis identitas dapat dikategorikan lebih lanjut sebagai kebijakan inline atau kebijakan yang dikelola. Kebijakan inline disematkan langsung ke satu pengguna, grup, atau peran. Kebijakan terkelola adalah kebijakan mandiri yang dapat dilampirkan ke beberapa pengguna, grup, dan peran dalam. Akun AWS Kebijakan AWS terkelola mencakup kebijakan terkelola dan kebijakan yang dikelola pelanggan. Untuk mempelajari cara memilih antara kebijakan terkelola atau kebijakan sebaris, lihat Memilih antara kebijakan terkelola dan kebijakan sebaris di IAMPanduan Pengguna.

Kebijakan berbasis sumber daya

Kebijakan berbasis sumber daya adalah dokumen JSON kebijakan yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan IAM peran dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan oleh prinsipal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus menentukan prinsipal dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

Kebijakan berbasis sumber daya merupakan kebijakan inline yang terletak di layanan tersebut. Anda tidak dapat menggunakan kebijakan AWS terkelola IAM dalam kebijakan berbasis sumber daya.

Daftar kontrol akses (ACLs)

Access control lists (ACLs) mengontrol prinsipal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACLsmirip dengan kebijakan berbasis sumber daya, meskipun mereka tidak menggunakan format dokumen kebijakan. JSON

Amazon S3, AWS WAF, dan Amazon VPC adalah contoh layanan yang mendukung. ACLs Untuk mempelajari selengkapnyaACLs, lihat Ikhtisar daftar kontrol akses (ACL) di Panduan Pengembang Layanan Penyimpanan Sederhana Amazon.

Jenis-jenis kebijakan lain

AWS mendukung jenis kebijakan tambahan yang kurang umum. Jenis-jenis kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda oleh jenis kebijakan yang lebih umum.

- Batas izin — Batas izin adalah fitur lanjutan tempat Anda menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas (pengguna atau peran). IAM IAM Anda dapat menetapkan batasan izin untuk suatu entitas. Izin yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas milik entitas dan batasan izinnya. Kebijakan berbasis sumber daya yang menentukan pengguna atau peran dalam bidang `Principal` tidak dibatasi oleh batasan izin. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya tentang batas izin, lihat Batas izin untuk IAM entitas di IAMPanduan Pengguna.
- Kebijakan kontrol layanan (SCPs) — SCPs adalah JSON kebijakan yang menentukan izin maksimum untuk organisasi atau unit organisasi (OU) di AWS Organizations. AWS Organizations

adalah layanan untuk mengelompokkan dan mengelola secara terpusat beberapa Akun AWS yang dimiliki bisnis Anda. Jika Anda mengaktifkan semua fitur dalam organisasi, Anda dapat menerapkan kebijakan kontrol layanan (SCPs) ke salah satu atau semua akun Anda. SCPMembatasi izin untuk entitas di akun anggota, termasuk setiap pengguna Akun AWS root. Untuk informasi selengkapnya tentang Organizations danSCPs, lihat Kebijakan kontrol layanan di Panduan AWS Organizations Pengguna.

- Kebijakan sesi – Kebijakan sesi adalah kebijakan lanjutan yang Anda berikan sebagai parameter ketika Anda membuat sesi sementara secara programatis untuk peran atau pengguna terfederasi. Izin sesi yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas pengguna atau peran dan kebijakan sesi. Izin juga bisa datang dari kebijakan berbasis sumber daya. Penolakan secara tegas dalam salah satu kebijakan ini membatalkan izin. Untuk informasi selengkapnya, lihat Kebijakan sesi di Panduan IAM Pengguna.

Berbagai jenis kebijakan

Ketika beberapa jenis kebijakan berlaku pada suatu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan saat beberapa jenis kebijakan terlibat, lihat Logika evaluasi kebijakan di Panduan IAM Pengguna.

Kontrol akses

Amazon SNS memiliki sistem izin berbasis sumber daya sendiri yang menggunakan kebijakan yang ditulis dalam bahasa yang sama yang digunakan untuk kebijakan (). AWS Identity and Access Management IAM Ini berarti Anda dapat mencapai hal serupa dengan SNS kebijakan dan IAM kebijakan Amazon.

> **Note**
>
> Penting untuk dipahami bahwa semua Akun AWS dapat mendelegasikan izin mereka kepada pengguna di bawah akun mereka. Akses lintas akun memungkinkan Anda berbagi akses ke sumber daya AWS tanpa harus mengelola pengguna tambahan. Untuk informasi tentang penggunaan akses lintas akun, lihat Mengaktifkan Akses Lintas Akun di Panduan Pengguna. IAM

Ikhtisar mengelola akses di Amazon SNS

Bagian ini menjelaskan konsep dasar yang perlu Anda pahami untuk menggunakan bahasa kebijakan akses untuk menulis kebijakan. Hal ini juga menjelaskan proses umum untuk bagaimana pengendalian akses bekerja dengan bahasa kebijakan akses, dan bagaimana kebijakan dievaluasi.

Topik

Kasus penggunaan kontrol SNS akses Amazon

Anda memiliki banyak fleksibilitas dalam cara Anda memberikan atau menolak akses ke sumber daya. Namun, kasus penggunaan yang umum cukup sederhana:

- Anda ingin memberi orang lain Akun AWS jenis tindakan topik tertentu (misalnya, Publikasikan). Untuk informasi selengkapnya, lihat Berikan Akun AWS akses ke suatu topik.
- Anda ingin membatasi langganan ke topik Anda hanya pada HTTPS protokol. Untuk informasi selengkapnya, lihat Batasi langganan ke HTTPS.
- Anda ingin mengizinkan Amazon SNS mempublikasikan pesan ke SQS antrian Amazon Anda. Untuk informasi selengkapnya, lihat Publikasikan pesan ke SQS antrian Amazon.

Konsep kebijakan SNS akses Amazon utama

Bagian berikut menjelaskan konsep yang perlu Anda pahami untuk menggunakan bahasa kebijakan akses. Konsep-konsep disajikan dalam urutan logis, dengan istilah pertama yang perlu Anda ketahui di bagian atas daftar.

Topik

- Kebijakan
- Penerbit
- Utama
- Tindakan
- Sumber Daya
- Syarat dan kunci
- Peminta
- Evaluasi
- Efek
- Blokir secara default
- Izinkan
- Penolakan eksplisit

Izin

Izinadalah konsep mengizinkan atau melarang beberapa jenis akses ke sumber daya tertentu. Izin pada dasarnya mengikuti bentuk ini: "A diizinkan/tidak diizinkan untuk melakukan B ke C jika D diterapkan." Misalnya, Jane (A) memiliki izin untuk menerbitkan (B) ke TopIca (C) selama dia menggunakan HTTP protokol (D). Setiap kali Jane memublikasikan TopicA, layanan memeriksa untuk melihat apakah dia memiliki izin dan apakah permintaan memenuhi persyaratan yang ditetapkan dalam izin.

Pernyataan

Pernyataan adalah deskripsi formal dari satu izin, yang ditulis dalam bahasa kebijakan akses. Anda selalu menulis pernyataan sebagai bagian dari dokumen kontainer yang lebih luas dikenal sebagai kebijakan (lihat konsep berikutnya).

Kebijakan

Kebijakan adalah dokumen (yang ditulis dalam bahasa kebijakan akses) yang bertindak sebagai kontainer untuk satu atau lebih pernyataan. Misalnya, kebijakan dapat memiliki dua pernyataan di dalamnya: satu yang menyatakan bahwa Jane dapat berlangganan menggunakan protokol email, dan satu lagi yang menyatakan bahwa Bob tidak dapat mempublikasikan ke Topik A. Seperti yang ditunjukkan pada gambar berikut, skenario yang setara adalah memiliki dua kebijakan, satu yang

menyatakan bahwa Jane dapat berlangganan menggunakan protokol email, dan satu lagi yang menyatakan bahwa Bob tidak dapat mempublikasikan ke Topik A.

Hanya ASCII karakter yang diizinkan dalam dokumen kebijakan. Anda dapat memanfaatkan `aws:SourceAccount` dan `aws:SourceOwner` mengatasi skenario di mana Anda perlu plug-in AWS layanan lain ARNs yang berisi non-karakter. ASCII Lihat perbedaan antara`aws:SourceAccount` versus `aws:SourceOwner`.

Penerbit

Penerbit adalah orang yang menulis kebijakan untuk memberikan izin untuk sumber daya. Penerbit (menurut definisi) selalu menjadi pemilik sumber daya. AWS tidak mengizinkan pengguna AWS layanan untuk membuat kebijakan untuk sumber daya yang tidak mereka miliki. Jika John adalah pemilik sumber daya, AWS mengautentikasi identitas John ketika dia mengirimkan kebijakan yang ditulisnya untuk memberikan izin untuk sumber daya tersebut.

Utama

Penanggung jawab adalah orang atau orang-orang yang menerima izin dalam kebijakan. Penanggung jawab adalah A dalam pernyataan "A memiliki izin untuk melakukan B ke C jika D diterapkan." Dalam kebijakan, Anda dapat mengatur penanggung jawab ke "siapa pun" (yaitu, Anda dapat menentukan wildcard untuk mewakili semua orang). Anda mungkin melakukan ini, misalnya, jika Anda tidak ingin membatasi akses berdasarkan identitas sebenarnya dari peminta, tetapi sebaliknya berdasarkan pada beberapa karakteristik lain yang mengidentifikasi seperti alamat IP peminta.

Tindakan

Tindakan adalah aktivitas yang izin untuk melakukannya dimiliki penanggung jawab. Tindakan adalah B dalam pernyataan "A memiliki izin untuk melakukan B ke C jika D diterapkan." Biasanya, tindakan

hanya operasi dalam permintaan untuk AWS. Misalnya, Jane mengirim permintaan ke Amazon SNS dengan `Action=Subscribe`. Anda dapat menentukan satu atau beberapa tindakan dalam kebijakan.

Sumber Daya

Sumber daya adalah obyek yang diminta aksesnya oleh penanggung jawab. Sumber daya adalah C dalam pernyataan "A memiliki izin untuk melakukan B ke C jika D diterapkan."

Syarat dan kunci

Syarat adalah pembatasan atau detail tentang izin. Syarat adalah D dalam pernyataan "A memiliki izin untuk melakukan B ke C jika D diterapkan." Bagian dari kebijakan yang menentukan syarat dapat menjadi yang paling rinci dan kompleks dari semua bagian. Syarat umum terkait dengan:

- Tanggal dan waktu (misalnya, permintaan harus tiba sebelum hari tertentu)
- Alamat IP (misalnya, alamat IP pemohon harus menjadi bagian dari CIDR rentang tertentu)

Kunci adalah karakteristik spesifik yang menjadi dasar pembatasan akses. Misalnya, tanggal dan waktu permintaan.

Anda menggunakan baik syarat maupun kunci bersama-sama untuk mengekspresikan pembatasan. Cara termudah untuk memahami bagaimana Anda benar-benar menerapkan pembatasan adalah dengan contoh: Jika Anda ingin membatasi akses sebelum 30 Mei 2010, Anda menggunakan syarat yang disebut `DateLessThan`. Anda menggunakan tombol yang disebut `aws:CurrentTime` dan mengaturnya ke nilai `2010-05-30T00:00:00Z`. AWS mendefinisikan syarat dan kunci yang dapat Anda gunakan. AWS Layanan itu sendiri (misalnya, Amazon SQS atau AmazonSNS) mungkin juga menentukan kunci khusus layanan. Untuk informasi selengkapnya, lihat SNSAPIIzin Amazon: Referensi tindakan dan sumber daya.

Peminta

Peminta adalah orang yang mengirimkan permintaan ke layanan AWS dan meminta akses ke sumber daya tertentu. Peminta mengirimkan permintaan ke AWS yang pada dasarnya mengatakan: "Apakah Anda mengizinkan saya untuk melakukan B ke C jika D diterapkan?"

Evaluasi

Evaluasi adalah proses yang digunakan AWS layanan untuk menentukan apakah permintaan yang masuk harus ditolak atau diizinkan berdasarkan kebijakan yang berlaku. Untuk informasi tentang logika evaluasi, lihat Logika evaluasi.

Efek

Efek adalah hasil yang Anda inginkan untuk dikembalikan pernyataan kebijakan xpada waktu evaluasi. Anda menentukan nilai ini ketika Anda menulis pernyataan dalam kebijakan, dan nilai-nilai yang mungkin adalah menolak dan mengizinkan.

Misalnya, Anda dapat menulis kebijakan yang memiliki pernyataan yang menyangkal semua permintaan yang berasal dari Antartika (effect=deny mengingat permintaan tersebut menggunakan alamat IP yang dialokasikan ke Antartika). Sebagai alternatif, Anda dapat menulis kebijakan yang memiliki pernyataan yang memungkinkan semua permintaan yang tidak berasal dari Antartika (effect=allow mengingat permintaan tersebut tidak berasal dari Antartika). Meskipun dua pernyataan tersebut tampak seperti melakukan hal yang sama, dalam logika bahasa kebijakan akses, keduanya berbeda. Untuk informasi selengkapnya, lihat Logika evaluasi.

Meskipun hanya ada dua nilai yang mungkin yang dapat Anda tentukan untuk efek (mengizinkan atau menolak), dapat ada tiga hasil yang berbeda pada waktu evaluasi kebijakan: blokir secara default, perizinan, atau penolakan eksplisit. Untuk informasi selengkapnya, lihat konsep berikut dan Logika evaluasi.

Blokir secara default

Blokir secara default adalah hasil default dari kebijakan jika tidak ada izin atau penolakan eksplisit.

Izinkan

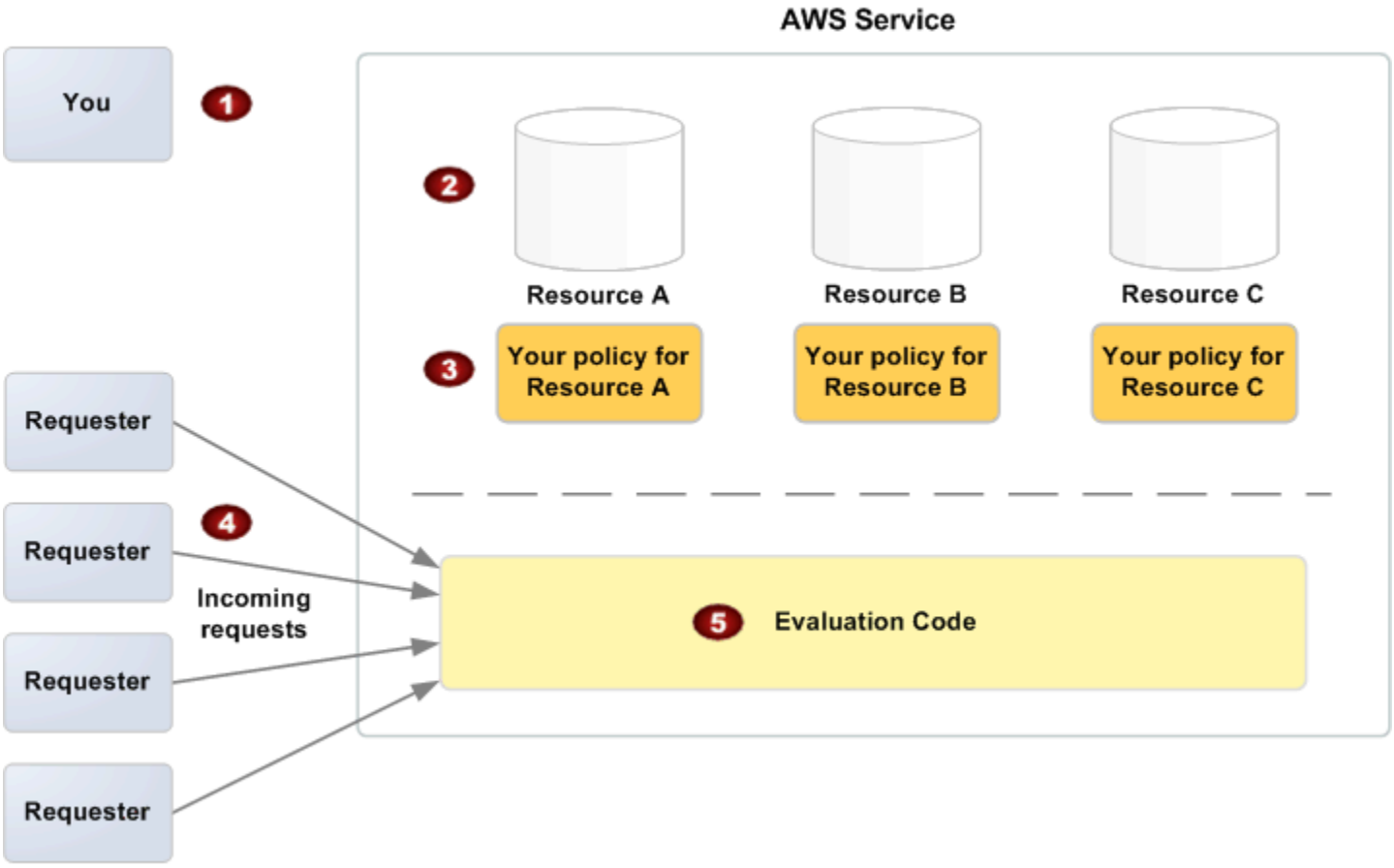
Perizinan dihasilkan dari pernyataan yang memiliki efek=mengizinkan, dengan asumsi syarat apa pun yang dinyatakan terpenuhi. Contoh: Mengizinkan permintaan jika diterima sebelum pukul 1:00 siang pada tanggal 30 April 2010. Perizinan mengabaikan semua blokir secara default, tetapi tidak mengabaikan penolakan eksplisit.

Penolakan eksplisit

Penolakan eksplisit dihasilkan dari pernyataan yang memiliki efek=menolak, dengan asumsi syarat apa pun yang dinyatakan terpenuhi. Contoh: Tolak semua permintaan jika berasal dari Antartika. Setiap permintaan yang berasal dari Antartika akan selalu ditolak tidak peduli apa yang mungkin diizinkan kebijakan lain apa pun.

Ikhtisar arsitektur kontrol SNS akses Amazon

Gambar dan tabel berikut menggambarkan komponen utama yang berinteraksi untuk memberikan pengendalian akses untuk sumber daya Anda.

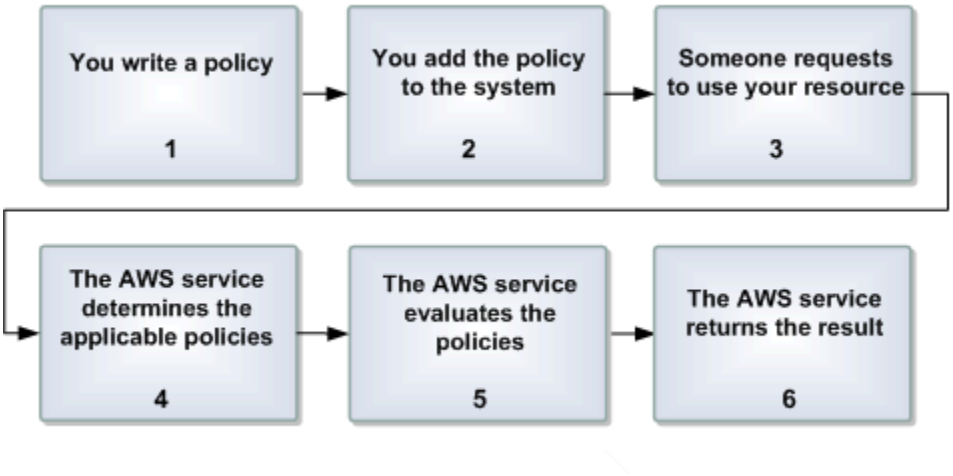


1	Anda, pemilik sumber daya.
2	Sumber daya Anda (terkandung dalam AWS layanan; misalnya, SQS antrian Amazon).
3	Kebijakan Anda. Biasanya Anda memiliki satu kebijakan per sumber daya, meskipun Anda bisa memiliki beberapa. AWS Layanan itu sendiri menyediakan API Anda gunakan untuk mengunggah dan mengelola kebijakan Anda.
4	Pemohon dan permintaan masuk mereka ke layanan. AWS
5	Kode evaluasi bahasa kebijakan akses.

	Ini adalah kumpulan kode dalam AWS layanan yang mengevaluasi permintaan masuk terhadap kebijakan yang berlaku dan menentukan apakah pemohon diizinkan mengakses sumber daya. Untuk informasi tentang cara membuat keputusan, lihat Logika evaluasi.

Menggunakan Bahasa Kebijakan Akses di Amazon SNS

Gambar dan tabel berikut mendeskripsikan proses umum bagaimana pengendalian akses bekerja dengan bahasa kebijakan akses.



Proses untuk menggunakan pengendalian akses dengan Bahasa Kebijakan Akses

1	Anda menulis kebijakan untuk sumber daya Anda. Misalnya, Anda menulis kebijakan untuk menentukan izin untuk SNS topik Amazon Anda.
2	Anda mengunggah kebijakan Anda ke AWS. AWS Layanan itu sendiri menyediakan API Anda gunakan untuk mengunggah kebijakan Anda. Misalnya, Anda menggunakan SNS `SetTopicAttributes` tindakan Amazon untuk mengunggah kebijakan untuk SNS topik Amazon tertentu.
3	Seseorang mengirimkan permintaan untuk menggunakan sumber daya Anda. Misalnya, pengguna mengirim permintaan ke Amazon SNS untuk menggunakan salah satu topik Anda.
4	AWS Layanan menentukan kebijakan mana yang berlaku untuk permintaan tersebut.

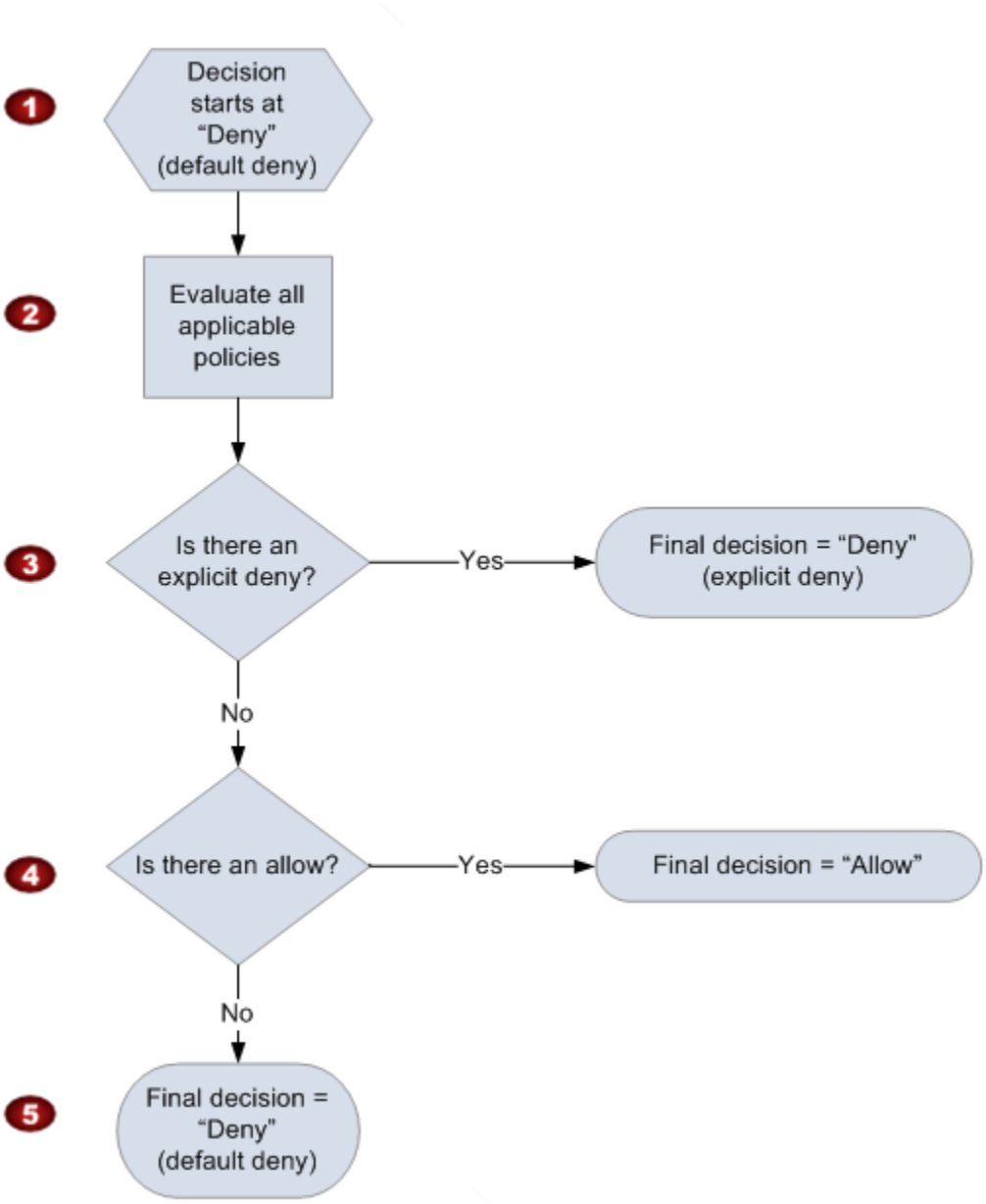
	Misalnya, Amazon SNS melihat semua SNS kebijakan Amazon yang tersedia dan menentukan mana yang berlaku (berdasarkan sumber dayanya, siapa pemohonnya, dll.).
5	AWS Layanan mengevaluasi kebijakan. Misalnya, Amazon SNS mengevaluasi kebijakan dan menentukan apakah pemohon diizinkan untuk menggunakan topik Anda atau tidak. Untuk informasi tentang logika keputusan, lihat Logika evaluasi.
6	AWS Layanan menolak permintaan atau terus memprosesnya. Sebagai contoh, berdasarkan hasil evaluasi kebijakan, layanan mengembalikan kesalahan "Akses ditolak" ke peminta atau melanjutkan untuk memproses permintaan.

Logika evaluasi

Tujuan pada waktu evaluasi adalah memutuskan apakah permintaan pemberian harus diizinkan atau ditolak. Logika evaluasi mengikuti beberapa aturan dasar:

- Secara default, semua permintaan untuk menggunakan sumber daya Anda yang berasal dari siapa pun selain Anda ditolak
- Perizinan mengabaikan penolakan default apa pun
- Penolakan eksplisit mengabaikan izin apa pun.
- Urutan evaluasi kebijakan tidak penting

Bagan alir dan diskusi berikut menjelaskan secara lebih rinci bagaimana keputusan dibuat.



1	Keputusan dimulai dengan penolakan default.
2	Kemudian, kode penerapan mengevaluasi semua kebijakan yang diterapkan pada permintaan (berdasarkan sumber daya, penanggung jawab, tindakan, dan syarat). Urutan yang digunakan kode penerapan untuk mengevaluasi kebijakan tidak penting.
3	Dalam semua kebijakan tersebut, kode penerapan mencari instruksi penolakan eksplisit yang akan diterapkan pada permintaan.

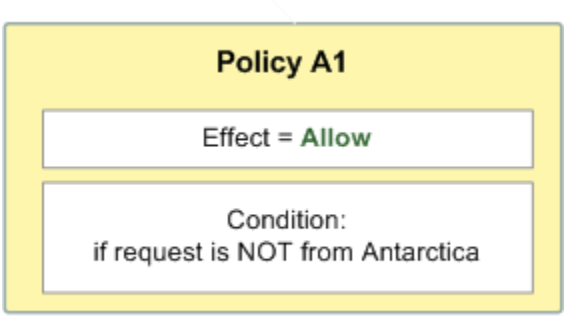
	Jika menemukan satu penolakan pun, kode penerapan akan mengembalikan keputusan "menolak" dan proses selesai (ini adalah penolakan eksplisit; untuk informasi lebih lanjut, lihat Penolakan eksplisit).
4	Jika penolakan eksplisit tidak ditemukan, kode penerapan mencari instruksi "mengizinkan" apa pun yang akan diterapkan pada permintaan. Jika menemukan satu instruksi pun, kode penerapan mengembalikan keputusan "mengizin kan" dan proses selesai (layanan melanjutkan untuk memproses permintaan).
5	Jika tidak ada perizinan ditemukan, maka keputusan akhir adalah "menolak" (karena tidak ada penolakan eksplisit atau memungkinkan, ini dianggap sebagai blokir secara default (untuk informasi selengkapnya, lihat Blokir secara default).

Interaksi penolakan eksplisit dan blokir secara default

Kebijakan menghasilkan blokir secara default jika tidak diterapkan secara langsung pada permintaan. Misalnya, jika pengguna meminta untuk menggunakan AmazonSNS, tetapi kebijakan tentang topik tersebut Akun AWS sama sekali tidak merujuk ke pengguna, maka kebijakan tersebut akan menghasilkan penolakan default.

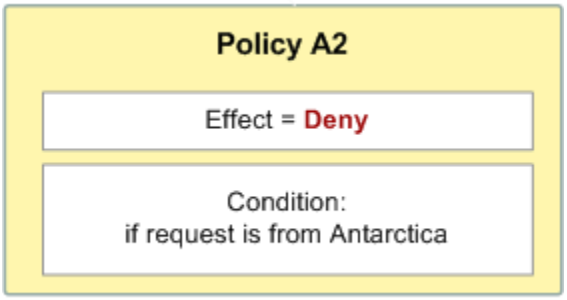
Kebijakan juga mengakibatkan blokir secara default jika suatu syarat dalam pernyataan tidak terpenuhi. Jika semua syarat dalam pernyataan terpenuhi, maka kebijakan menghasilkan perizinan atau penolakan eksplisit, berdasarkan nilai elemen Efek dalam kebijakan. Kebijakan tidak menentukan apa yang harus dilakukan jika syarat tidak terpenuhi, sehingga hasil default dalam kasus tersebut adalah blokir secara default.

Misalnya, katakanlah Anda ingin mencegah permintaan masuk dari Antartika. Anda menulis kebijakan (disebut Kebijakan A1) yang mengizinkan permintaan hanya jika tidak datang dari Antartika. Diagram berikut menggambarkan kebijakan.



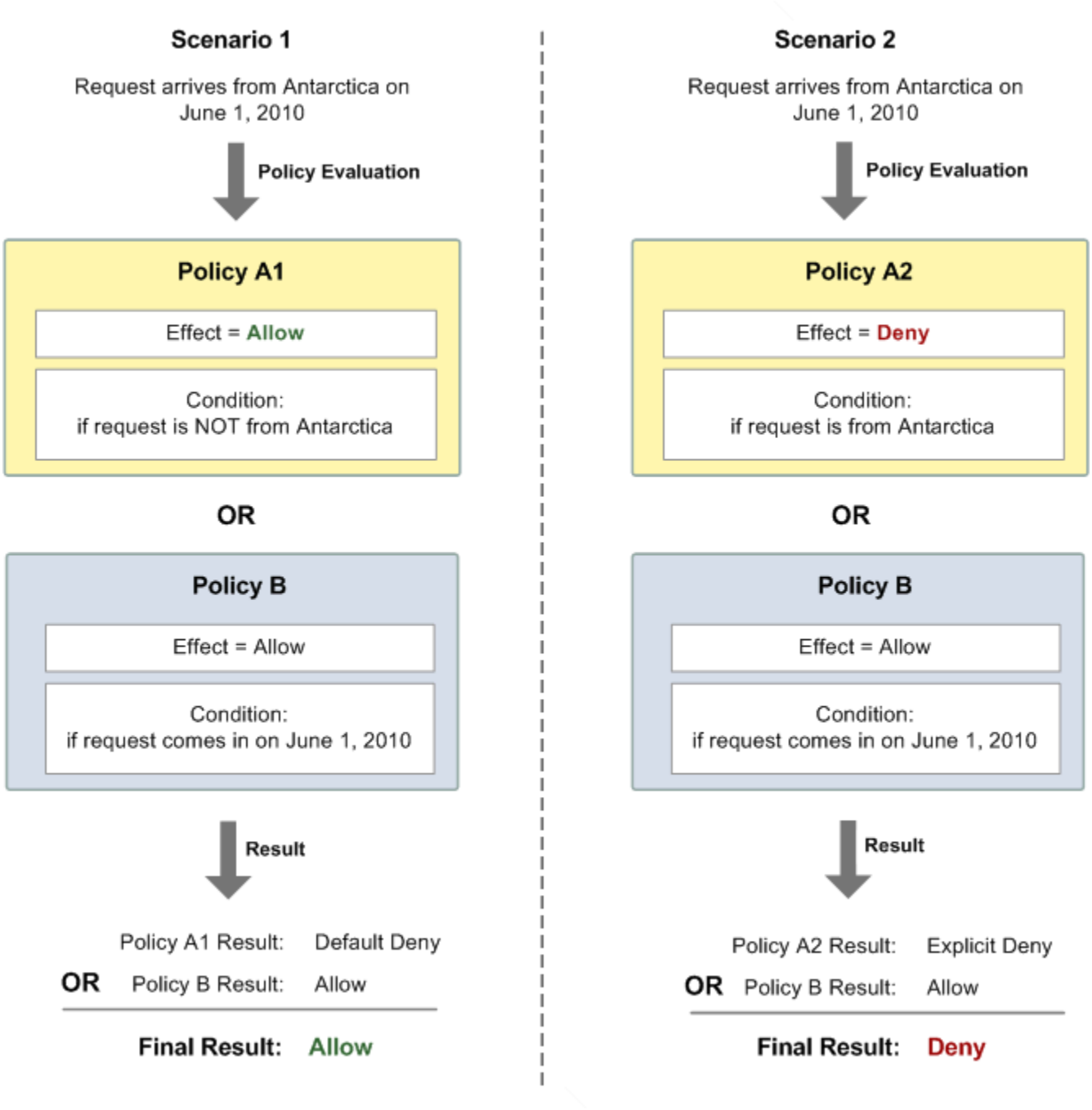
Jika seseorang mengirimkan permintaan dari AS, syaratnya terpenuhi (permintaannya bukan dari Antartika). Oleh karena itu, permintaan diizinkan. Namun, jika seseorang mengirimkan permintaan dari Antartika, syarat tidak terpenuhi, dan oleh karena itu hasil kebijakan adalah blokir secara default.

Anda dapat mengubah hasilnya menjadi penolakan eksplisit dengan menulis ulang kebijakan (bernama Kebijakan A2) seperti dalam diagram berikut. Di sini, kebijakan secara eksplisit menolak permintaan jika berasal dari Antartika.



Jika seseorang mengirimkan permintaan dari Antartika, syarat terpenuhi, dan karena itu hasil kebijakan adalah penolakan eksplisit.

Perbedaan antara blokir secara default dan penolakan eksplisit penting karena blokir secara default dapat diabaikan oleh perizinan, tetapi penolakan eksplisit tidak da[at diabaikan. Misalnya, ada kebijakan lain yang mengizinkan permintaan jika tiba pada tanggal 1 Juni 2010. Bagaimana kebijakan ini mempengaruhi hasil keseluruhan ketika digabungkan dengan kebijakan yang membatasi akses dari Antartika? Kami akan membandingkan hasil keseluruhan saat menggabungkan kebijakan berbasis tanggal (akan disebut Kebijakan B) dengan kebijakan A1 dan A2 sebelumnya. Skenario 1 menggabungkan Kebijakan A1 dengan Kebijakan B, dan Skenario 2 menggabungkan Kebijakan A2 dengan Kebijakan B. Gambar dan diskusi berikut menunjukkan hasil saat permintaan datang dari Antartika pada 1 Juni 2010.



Dalam Skenario 1, Kebijakan A1 mengembalikan blokir secara default, seperti yang dijelaskan sebelumnya dalam bagian ini. Kebijakan B mengembalikan perizinan karena kebijakan (menurut definisi) mengizinkan permintaan yang datang pada 1 Juni 2010. Perizinan dari Kebijakan B mengabaikan blokir secara default dari Kebijakan A1, dan oleh karena itu permintaan diperbolehkan.

Dalam Skenario 2, Kebijakan A2 mengembalikan penolakan eksplisit, seperti yang dijelaskan sebelumnya dalam bagian ini. Sekali lagi, Kebijakan B mengembalikan perizinan. Perizinan dari Kebijakan A2 mengabaikan perizinan dari Kebijakan B, dan oleh karena itu permintaan ditolak.

Contoh kasus untuk kontrol SNS akses Amazon

Bagian ini menjelaskan beberapa contoh kasus penggunaan umum untuk kontrol akses.

Topik

Berikan Akun AWS akses ke suatu topik

Katakanlah Anda memiliki topik di AmazonSNS, dan Anda ingin mengizinkan satu atau lebih Akun AWS untuk melakukan tindakan tertentu pada topik itu, seperti mempublikasikan pesan. Anda dapat melakukannya dengan menggunakan SNS API tindakan `AddPermission` Amazon.

`AddPermission`Tindakan ini memungkinkan Anda menentukan topik, daftar Akun AWS IDs, daftar tindakan, dan label. Amazon SNS kemudian secara otomatis membuat dan menambahkan pernyataan kebijakan baru ke kebijakan kontrol akses topik. Anda tidak perlu menulis pernyataan kebijakan sendiri—Amazon SNS menangani ini untuk Anda. Jika Anda perlu menghapus kebijakan nanti, Anda dapat melakukannya dengan menelepon `RemovePermission` dan memberikan label yang Anda gunakan saat menambahkan izin.

Misalnya, jika Anda memanggil `AddPermission` topik arn:aws:sns:us-east- 2:444455556666:, MyTopic tentukan ID Akun AWS 1111-2222-3333, tindakan, dan label, Amazon akan membuat dan menyisipkan pernyataan kebijakan berikut ke dalam kebijakan `Publish` kontrol akses topik: `grant-1234-publish` SNS

```
{
  "Statement": [{
    "Sid": "grant-1234-publish",
```

```
    "Effect": "Allow",
    "Principal": {
      "AWS": "111122223333"
    },
    "Action": ["sns:Publish"],
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic"
  }]
}
```

Setelah pernyataan ini ditambahkan, Akun AWS 1111-2222-3333 akan memiliki izin untuk mempublikasikan pesan ke topik tersebut.

Informasi tambahan:

- Manajemen kebijakan khusus: Meskipun `AddPermission` nyaman untuk memberikan izin, seringkali berguna untuk mengelola kebijakan kontrol akses topik secara manual untuk skenario yang lebih kompleks, seperti menambahkan kondisi atau memberikan izin ke peran atau layanan tertentuIAM. Anda dapat melakukannya dengan menggunakan atribut `SetTopicAttributes` API to update policy secara langsung.
- Praktik terbaik keamanan: Berhati-hatilah saat memberikan izin untuk memastikan bahwa hanya entitas tepercaya Akun AWS atau entitas yang memiliki akses ke topik Amazon Anda. SNS Secara teratur meninjau dan mengaudit kebijakan yang dilampirkan pada topik Anda untuk menjaga keamanan.
- Batas kebijakan: Perlu diingat bahwa ada batasan ukuran dan kompleksitas SNS kebijakan Amazon. Jika Anda perlu menambahkan banyak izin atau kondisi rumit, pastikan kebijakan Anda tetap berada dalam batas-batas ini.

Batasi langganan ke HTTPS

Untuk membatasi protokol pengiriman notifikasi untuk SNS topik Amazon AndaHTTPS, Anda harus membuat kebijakan khusus. `AddPermission`Tindakan di Amazon SNS tidak memungkinkan Anda menentukan batasan protokol saat memberikan akses ke topik Anda. Oleh karena itu, Anda perlu menulis kebijakan secara manual yang memberlakukan pembatasan ini dan kemudian menggunakan `SetTopicAttributes` tindakan tersebut untuk menerapkan kebijakan tersebut ke topik Anda.

Berikut cara membuat kebijakan yang membatasi langganan untukHTTPS:

1. Tulis kebijakan. Kebijakan harus menentukan Akun AWS ID yang ingin Anda berikan akses dan menerapkan kondisi bahwa hanya HTTPS langganan yang diizinkan. Di bawah ini adalah contoh

kebijakan yang memberikan izin Akun AWS ID 1111-2222-3333 untuk berlangganan topik, tetapi hanya jika protokol yang digunakan adalah. HTTPS

```
{
  "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"
      }
    }
  }]
}
```

2. Terapkan Kebijakan. Gunakan `SetTopicAttributes` tindakan di Amazon SNS API untuk menerapkan kebijakan ini ke topik Anda. Tetapkan `Policy` atribut topik ke JSON kebijakan yang Anda buat.

```
snsClient.setTopicAttributes(SetTopicAttributesRequest.builder()
        .topicArn("arn:aws:sns:us-east-2:444455556666:MyTopic")
        .attributeName("Policy")
        .attributeValue(jsonPolicyString)  // The JSON policy as a string
        .build());
```

Informasi tambahan:

- Menyesuaikan kontrol akses. Pendekatan ini memungkinkan Anda untuk menerapkan kontrol akses yang lebih terperinci, seperti membatasi protokol berlangganan, yang tidak mungkin dilakukan melalui tindakan saja. `AddPermission` Kebijakan khusus memberikan fleksibilitas untuk skenario yang memerlukan kondisi tertentu, seperti penegakan protokol atau pembatasan alamat IP.
- Praktik terbaik keamanan. Membatasi langganan untuk HTTPS meningkatkan keamanan notifikasi Anda dengan memastikan bahwa data dalam perjalanan dienkripsi. Tinjau kebijakan topik Anda secara teratur untuk memastikan mereka memenuhi persyaratan keamanan dan kepatuhan Anda.

- Pengujian kebijakan. Sebelum menerapkan kebijakan dalam lingkungan produksi, ujilah di lingkungan pengembangan untuk memastikannya berperilaku seperti yang diharapkan. Ini membantu mencegah masalah akses yang tidak disengaja atau pembatasan yang tidak diinginkan.

Publikasikan pesan ke SQS antrian Amazon

Untuk mempublikasikan pesan dari SNS topik Amazon Anda ke SQS antrean Amazon, Anda perlu mengonfigurasi izin yang benar pada antrian AmazonSQS. Meskipun Amazon SNS dan Amazon SQS menggunakan AWS bahasa kebijakan kontrol akses, Anda harus secara eksplisit menetapkan kebijakan pada SQS antrian Amazon untuk mengizinkan pesan dikirim dari topik Amazon. SNS

Anda dapat mencapai ini dengan menggunakan `SetQueueAttributes` tindakan untuk menerapkan kebijakan kustom ke SQS antrian Amazon. Tidak seperti AmazonSNS, Amazon SQS tidak mendukung `AddPermission` tindakan untuk membuat pernyataan kebijakan dengan kondisi. Karena itu, Anda harus menulis kebijakan secara manual.

Berikut ini adalah contoh SQS kebijakan Amazon yang memberikan SNS izin Amazon untuk mengirim pesan ke antrian Anda. Perhatikan bahwa kebijakan ini dikaitkan dengan SQS antrian Amazon, bukan SNS topik Amazon. Tindakan yang ditentukan adalah SQS tindakan Amazon, dan sumber daya adalah Amazon Resource Name (ARN) dari antrean. Anda dapat mengambil antrean ARN dengan menggunakan tindakan. `GetQueueAttributes`

```
{
  "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"
      }
    }
  }]
}
```

Kebijakan ini menggunakan `aws:SourceArn` kondisi untuk membatasi akses ke SQS antrian berdasarkan sumber pesan yang dikirim. Ini memastikan bahwa hanya pesan yang berasal dari SNS topik tertentu (dalam hal ini, arn:aws:sns:us-east- 2:444455556666:) yang diizinkan untuk dikirim ke antrian. MyTopic

Informasi tambahan:

- AntrianARN. Pastikan Anda mengambil ARN SQS antrean Amazon yang benar menggunakan tindakan. `GetQueueAttributes` Ini ARN penting untuk mengatur izin yang benar.
- Praktik terbaik keamanan. Saat membuat kebijakan, selalu ikuti prinsip hak istimewa paling sedikit. Berikan hanya izin yang diperlukan ke SNS topik Amazon untuk berinteraksi dengan SQS antrian Amazon, dan tinjau kebijakan Anda secara teratur untuk memastikan kebijakan tersebut aman dan aman up-to-date
- Kebijakan default di AmazonSNS. Bertentangan dengan beberapa kesalahpahaman, Amazon SNS tidak secara otomatis memberikan kebijakan default yang memungkinkan Layanan AWS akses lain ke topik yang baru dibuat. Anda harus secara eksplisit menentukan dan melampirkan kebijakan untuk mengontrol akses ke topik Amazon SNS Anda.
- Pengujian dan validasi. Setelah menyetel kebijakan, uji integrasi dengan menerbitkan pesan ke SNS topik Amazon dan memverifikasi bahwa pesan tersebut berhasil dikirim ke SQS antrean Amazon. Ini membantu mengonfirmasi bahwa kebijakan telah dikonfigurasi dengan benar.

Izinkan pemberitahuan acara Amazon S3 untuk mempublikasikan ke suatu topik

Untuk mengizinkan bucket Amazon S3 dari yang lain Akun AWS untuk mempublikasikan pemberitahuan peristiwa ke SNS topik Amazon Anda, Anda perlu mengonfigurasi kebijakan akses topik yang sesuai. Ini melibatkan penulisan kebijakan khusus yang memberikan izin ke layanan Amazon S3 dari yang Akun AWS spesifik dan kemudian menerapkan kebijakan ini ke topik Amazon SNS Anda.

Inilah cara Anda dapat mengaturnya:

1. Tulis kebijakan. Kebijakan tersebut harus memberikan layanan Amazon S3 (s3.amazonaws.com) izin yang diperlukan untuk mempublikasikan ke SNS topik Amazon Anda. Anda akan menggunakan `SourceAccount` kondisi ini untuk memastikan bahwa hanya yang ditentukan Akun AWS, yang memiliki bucket Amazon S3, yang dapat mempublikasikan pemberitahuan ke topik Anda.

   Berikut ini adalah contoh kebijakan.

```
{
  "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"
      }
    }
  }]
}
```

- Pemilik topik — 111122223333 adalah ID Akun AWS yang memiliki topik Amazon. SNS
- Pemilik bucket Amazon S3 — 444455556666 adalah ID yang Akun AWS memiliki bucket Amazon S3 mengirimkan notifikasi.

2. Terapkan Kebijakan. Gunakan `SetTopicAttributes` tindakan untuk menetapkan kebijakan ini pada SNS topik Amazon Anda. Ini akan memperbarui kontrol akses topik untuk menyertakan izin yang ditentukan dalam kebijakan kustom Anda.

```
snsClient.setTopicAttributes(SetTopicAttributesRequest.builder()
        .topicArn("arn:aws:sns:us-east-2:111122223333:MyTopic")
        .attributeName("Policy")
        .attributeValue(jsonPolicyString)  // The JSON policy as a string
        .build());
```

Informasi tambahan:

- Menggunakan **`SourceAccount`** kondisi. `SourceAccount`Kondisi ini memastikan bahwa hanya peristiwa yang berasal dari yang ditentukan Akun AWS (444455556666 dalam kasus ini) yang dapat memicu topik Amazon. SNS Ini adalah langkah keamanan untuk mencegah akun yang tidak sah mengirim pemberitahuan ke topik Anda.
- Layanan lain yang mendukung**`SourceAccount`**. `SourceAccount`Kondisi ini didukung oleh layanan berikut. Sangat penting untuk menggunakan kondisi ini ketika Anda ingin membatasi akses ke SNS topik Amazon Anda berdasarkan akun asal.

- APIGerbang Amazon
- Amazon CloudWatch
- DevOpsGuru Amazon
- Amazon EventBridge
- Amazon GameLift
- Amazon Pinpoint SMS dan Suara API
- Amazon RDS
- Amazon Redshift
- Amazon S3 Glacier
- Amazon SES
- Amazon Simple Storage Service
- AWS CodeCommit
- AWS Directory Service
- AWS Lambda
- AWS Systems Manager Incident Manager

- Pengujian dan validasi. Setelah menerapkan kebijakan, uji penyiapan dengan memicu peristiwa di bucket Amazon S3 dan mengonfirmasi bahwa itu berhasil dipublikasikan ke topik Amazon Anda. SNS Ini akan membantu memastikan bahwa kebijakan Anda dikonfigurasi dengan benar.
- Praktik terbaik keamanan. Tinjau dan audit kebijakan SNS topik Amazon Anda secara teratur untuk memastikan kebijakan tersebut mematuhi persyaratan keamanan Anda. Membatasi akses hanya ke akun dan layanan tepercaya sangat penting untuk menjaga operasi yang aman.

SESIzinkan Amazon mempublikasikan ke topik yang dimiliki oleh akun lain

Anda dapat mengizinkan orang lain Layanan AWS untuk mempublikasikan ke topik yang dimiliki oleh orang lain Akun AWS. Misalkan Anda masuk ke akun 111122223333, membuka SES Amazon, dan membuat email. Untuk mempublikasikan pemberitahuan tentang email ini ke SNS topik Amazon yang dimiliki akun 444455556666, Anda harus membuat kebijakan seperti berikut ini. Untuk melakukannya, Anda perlu memberikan informasi tentang penanggung jawab (layanan lainnya) dan kepemilikan setiap sumber daya. `Resource`Pernyataan tersebut memberikan topikARN, yang mencakup ID akun pemilik topik, 444455556666. Pernyataan `"aws:SourceOwner": "111122223333"` menentukan bahwa akun Anda memiliki email.

```
{
  "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"
        }
      }
    }
  ]
}
```

Saat memublikasikan acara ke AmazonSNS, layanan berikut mendukung`aws:SourceOwner`:

- APIGerbang Amazon
- Amazon CloudWatch
- DevOpsGuru Amazon
- Amazon GameLift
- Amazon Pinpoint SMS dan Suara API
- Amazon RDS
- Amazon Redshift
- Amazon SES
- AWS CodeCommit
- AWS Directory Service
- AWS Lambda
- AWS Systems Manager Incident Manager

`aws:SourceAccount` versus `aws:SourceOwner`

> **Important**
>
> `aws:SourceOwner`tidak digunakan lagi dan layanan baru dapat diintegrasikan dengan Amazon SNS hanya melalui dan. `aws:SourceArn` `aws:SourceAccount` Amazon SNS masih mempertahankan kompatibilitas mundur untuk layanan yang ada yang saat ini mendukung`aws:SourceOwner`.

Kunci `aws:SourceAccount` dan `aws:SourceOwner` kondisi masing-masing ditetapkan oleh beberapa orang Layanan AWS ketika mereka mempublikasikan ke SNS topik Amazon. Ketika didukung, nilainya akan menjadi ID AWS akun 12 digit yang atas nama layanan tersebut menerbitkan data. Beberapa layanan mendukung satu, dan beberapa mendukung yang lain.

- Lihat Izinkan pemberitahuan acara Amazon S3 untuk mempublikasikan ke suatu topik bagaimana notifikasi Amazon S3 digunakan `aws:SourceAccount` dan daftar AWS layanan yang mendukung kondisi tersebut.
- Lihat SESIzinkan Amazon mempublikasikan ke topik yang dimiliki oleh akun lain bagaimana Amazon SES menggunakan `aws:SourceOwner` dan daftar AWS layanan yang mendukung kondisi tersebut.

Izinkan akun di organisasi AWS Organizations untuk mempublikasikan ke topik di akun yang berbeda

AWS Organizations Layanan ini membantu Anda mengelola penagihan secara terpusat, mengontrol akses dan keamanan, dan berbagi sumber daya di seluruh Anda. Akun AWS

Anda dapat menemukan ID organisasi Anda di konsol Organizations. Untuk informasi selengkapnya, lihat Melihat detail organisasi dari akun manajemen.

Dalam contoh ini, setiap Akun AWS organisasi `myOrgId` dapat mempublikasikan ke SNS topik Amazon `MyTopic` di akun`444455556666`. Kebijakan memeriksa nilai ID organisasi menggunakan kunci syarat global `aws:PrincipalOrgID`.

```
{
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
```

```
                "AWS": "*"
            },
            "Action": "SNS:Publish",
            "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
            "Condition": {
                "StringEquals": {
                    "aws:PrincipalOrgID": "myOrgId"
                }
            }
        }
    ]
}
```

Izinkan CloudWatch alarm apa pun untuk mempublikasikan ke topik di akun yang berbeda

Dalam hal ini, CloudWatch alarm apa pun di akun 111122223333 diizinkan untuk dipublikasikan ke SNS topik Amazon di akun444455556666.

```
{
    "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:*"
                }
            }
        }
    ]
}
```

Batasi publikasi ke SNS topik Amazon hanya dari titik akhir tertentu VPC

Dalam hal ini, topik di akun 444455556666 diizinkan untuk dipublikasikan hanya dari titik akhir dengan ID. VPC vpce-1ab2c34d

```
{
```

```
  "Statement": [{
    "Effect": "Deny",
    "Principal": "*",
    "Action": "SNS:Publish",
    "Resource": "arn:aws:sns:us-east-2:444455556666:MyTopic",
    "Condition": {
      "StringNotEquals": {
        "aws:sourceVpce": "vpce-1ab2c34d"
      }
    }
  }]
}
```

Bagaimana Amazon SNS bekerja dengan IAM

Sebelum Anda menggunakan IAM untuk mengelola akses ke AmazonSNS, pelajari IAM fitur apa saja yang tersedia untuk digunakan dengan AmazonSNS.

IAMfitur yang dapat Anda gunakan dengan Amazon Simple Notification Service

IAMfitur	SNSDukungan Amazon
Kebijakan berbasis identitas	Ya
Kebijakan berbasis sumber daya	Ya
Tindakan kebijakan	Ya
Sumber daya kebijakan	Ya
kunci-kunci persyaratan kebijakan (spesifik layanan)	Ya
ACLs	Tidak
ABAC(tag dalam kebijakan)	Parsial
Kredensial sementara	Ya
Izin prinsipal	Ya

IAMfitur	SNSDukungan Amazon
Peran layanan	Ya
Peran terkait layanan	Tidak

Untuk mendapatkan tampilan tingkat tinggi tentang cara kerja Amazon SNS dan AWS layanan lainnya dengan sebagian besar IAM fitur, lihat AWS layanan yang berfungsi IAM di Panduan IAM Pengguna.

AWS kebijakan terkelola untuk Amazon Simple Notification Service

Kebijakan AWS terkelola adalah kebijakan mandiri yang dibuat dan dikelola oleh AWS. AWS Kebijakan terkelola dirancang untuk memberikan izin bagi banyak kasus penggunaan umum sehingga Anda dapat mulai menetapkan izin kepada pengguna, grup, dan peran.

Perlu diingat bahwa kebijakan AWS terkelola mungkin tidak memberikan izin hak istimewa paling sedikit untuk kasus penggunaan spesifik Anda karena tersedia untuk digunakan semua pelanggan. AWS Kami menyarankan Anda untuk mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola pelanggan yang khusus untuk kasus penggunaan Anda.

Anda tidak dapat mengubah izin yang ditentukan dalam kebijakan AWS terkelola. Jika AWS memperbarui izin yang ditentukan dalam kebijakan AWS terkelola, pembaruan akan memengaruhi semua identitas utama (pengguna, grup, dan peran) yang dilampirkan kebijakan tersebut. AWS kemungkinan besar akan memperbarui kebijakan AWS terkelola saat baru Layanan AWS diluncurkan atau API operasi baru tersedia untuk layanan yang ada.

Untuk informasi selengkapnya, lihat kebijakan AWS terkelola di Panduan IAM Pengguna.

AWS kebijakan terkelola: mazonSNSFull Akses

`AmazonSNSFullAccess`menyediakan akses penuh ke Amazon SNS menggunakan file AWS Management Console. Kebijakan ini juga mencakup tindakan baca dan tulis berikut AWS Olah Pesan Pengguna Akhir SMS saat dipanggil menggunakan AmazonSNS. Anda dapat melampirkan kebijakan ini ke pengguna, grup, atau peran Anda.

Detail izin

Izin berikut hanya berlaku saat menggunakan Amazon SNSAPIs:

- `sns:*`— Memungkinkan izin penuh untuk melakukan tindakan apa pun yang terkait dengan AmazonSNS. Wildcard ini (*) berarti bahwa pengguna dapat menjalankan semua SNS tindakan Amazon yang mungkin.
- `sms-voice:DescribeVerifiedDestinationNumbers`— Memungkinkan Anda untuk mengambil daftar nomor telepon yang telah diverifikasi untuk mengirim SMS pesan dalam Akun AWS.
- `sms-voice:CreateVerifiedDestinationNumber`— Memungkinkan Anda memverifikasi nomor telepon baru untuk digunakan dengan layanan SMS pesan di dalamnya AWS.
- `sms-voice:SendDestinationNumberVerificationCode`— Memungkinkan Anda mengirim kode verifikasi ke nomor telepon yang sedang dalam proses diverifikasi untuk SMS pengiriman pesan di dalamnya AWS.
- `sms-voice:SendTextMessage`— Memungkinkan Anda membuat pesan teks baru dan mengirimkannya ke nomor telepon penerima. `SendTextMessage`hanya mengirim SMS pesan ke satu penerima setiap kali dipanggil.
- `sms-voice:DeleteVerifiedDestinationNumber`— Memungkinkan Anda untuk menghapus nomor telepon dari daftar nomor yang diverifikasi dalam Akun AWS
- `sms-voice:VerifyDestinationNumber`— Memungkinkan Anda untuk memulai dan menyelesaikan proses verifikasi untuk nomor telepon yang akan digunakan untuk layanan SMS pesan di dalamnya AWS.
- `sms-voice:DescribeAccountAttributes`— Memungkinkan Anda untuk mengambil informasi rinci tentang atribut tingkat akun yang terkait dengan SMS layanan pesan di dalamnya. AWS
- `sms-voice:DescribeSpendLimits`— Memungkinkan Anda untuk mengambil informasi tentang batas pengeluaran yang terkait dengan layanan SMS pesan dalam Akun AWS
- `sms-voice:DescribePhoneNumbers`— Memungkinkan Anda untuk mengambil informasi rinci tentang nomor telepon yang terkait dengan layanan SMS pesan dalam Akun AWS
- `sms-voice:SetTextMessageSpendLimitOverride`— Memungkinkan Anda untuk mengatur atau mengganti batas pengeluaran untuk pesan SMS teks dalam Akun AWS
- `sms-voice:DescribeOptedOutNumbers`— Memungkinkan Anda untuk mengambil daftar nomor telepon yang telah memilih untuk tidak menerima SMS pesan dari akun Anda AWS .
- `sms-voice:DeleteOptedOutNumber`— Memungkinkan Anda untuk menghapus nomor telepon dari daftar nomor opted-out dalam Akun AWS

AmazonSNSFullAccesscontoh kebijakan

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SNSFullAccess",
            "Effect": "Allow",
            "Action": "sns:*",
            "Resource": "*"
        },
        {
            "Sid": "SMSAccessViaSNS",
            "Effect": "Allow",
            "Action": [
                "sms-voice:DescribeVerifiedDestinationNumbers",
                "sms-voice:CreateVerifiedDestinationNumber",
                "sms-voice:SendDestinationNumberVerificationCode",
                "sms-voice:SendTextMessage",
                "sms-voice:DeleteVerifiedDestinationNumber",
                "sms-voice:VerifyDestinationNumber",
                "sms-voice:DescribeAccountAttributes",
                "sms-voice:DescribeSpendLimits",
                "sms-voice:DescribePhoneNumbers",
                "sms-voice:SetTextMessageSpendLimitOverride",
                "sms-voice:DescribeOptedOutNumbers",
                "sms-voice:DeleteOptedOutNumber"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaLast": "sns.amazonaws.com"
                }
            }
        }
    ]
}
```

Untuk melihat izin kebijakan ini, lihat mazonSNSFullAkses di Referensi Kebijakan AWS Terkelola.

AWS kebijakan terkelola: A mazonSNSRead OnlyAccess

`AmazonSNSReadOnlyAccess`menyediakan akses hanya-baca ke Amazon SNS menggunakan file. AWS Management Console Kebijakan ini juga mencakup tindakan hanya-baca berikut AWS Olah Pesan Pengguna Akhir SMS saat dipanggil menggunakan Amazon. SNS Anda dapat melampirkan kebijakan ini ke pengguna, grup, dan peran Anda.

Detail izin

Izin berikut hanya berlaku saat menggunakan Amazon SNSAPIs:

- `sns:GetTopicAttributes`— Memungkinkan Anda untuk mengambil atribut dari SNS topik Amazon. Ini termasuk informasi seperti topik ARN (Nama Sumber Daya Amazon), daftar pelanggan, kebijakan pengiriman, kebijakan kontrol akses, dan metadata lain yang terkait dengan topik tersebut.
- `sns:List*`— Memungkinkan Anda melakukan operasi apa pun yang dimulai dengan `List` untuk SNS sumber daya Amazon. Ini termasuk izin untuk mencantumkan berbagai elemen yang terkait dengan AmazonSNS, seperti:
  - `sns:ListTopics`— Memungkinkan Anda untuk mengambil daftar semua SNS topik Amazon di. Akun AWS
  - `sns:ListSubscriptions`— Memungkinkan Anda untuk mengambil daftar semua langganan ke topik AmazonSNS.
  - `sns:ListSubscriptionsByTopic`— Memungkinkan Anda membuat daftar semua langganan untuk SNS topik Amazon tertentu.
  - `sns:ListPlatformApplications`— Memungkinkan Anda untuk membuat daftar semua aplikasi platform yang dibuat untuk pemberitahuan push seluler.
  - `sns:ListEndpointsByPlatformApplication`— Memungkinkan Anda untuk membuat daftar semua titik akhir yang terkait dengan aplikasi platform.
- `sns:CheckIfPhoneNumberIsOptedOut`— Memungkinkan Anda memeriksa apakah nomor telepon tertentu telah memilih untuk tidak menerima SMS pesan melalui AmazonSNS.
- `sns:GetEndpointAttributes`— Memungkinkan Anda mengambil atribut titik akhir yang terkait dengan aplikasi SNS platform Amazon. Ini dapat mencakup atribut seperti status diaktifkan titik akhir, data pengguna kustom, dan metadata lain yang terkait dengan titik akhir.
- `sns:GetDataProtectionPolicy`— Memungkinkan Anda mengambil kebijakan perlindungan data yang terkait dengan SNS topik Amazon.

- `sns:GetPlatformApplicationAttributes`— Memungkinkan Anda untuk mengambil atribut dari aplikasi SNS platform Amazon. Aplikasi platform digunakan di Amazon SNS untuk mengirim pemberitahuan push ke perangkat seluler melalui layanan seperti Apple Push Notification Service (APNS) atau Firebase Cloud Messaging (FCM).
- `sns:GetSMSAttributes`— Memungkinkan Anda untuk mengambil SMS pengaturan default untuk. Akun AWS
- `sns:GetSMSSandboxAccountStatus`— Memungkinkan Anda untuk mengambil status SMS kotak pasir saat ini untuk Anda. Akun AWS
- `sns:GetSubscriptionAttributes`— Memungkinkan Anda mengambil atribut langganan tertentu ke SNS topik Amazon.
- `sms-voice:DescribeVerifiedDestinationNumbers`— Memungkinkan Anda untuk melihat atau mengambil daftar nomor telepon yang telah diverifikasi untuk mengirim SMS pesan dalam Akun AWS
- `sms-voice:DescribeAccountAttributes`— Memungkinkan Anda untuk melihat atau mengambil informasi tentang atribut tingkat akun yang terkait dengan SMS layanan pesan di dalamnya. AWS
- `sms-voice:DescribeSpendLimits`— Memungkinkan Anda untuk melihat atau mengambil informasi tentang batas pengeluaran yang terkait dengan layanan SMS pesan dalam Akun AWS
- `sms-voice:DescribePhoneNumbers`— Memungkinkan Anda untuk melihat atau mengambil informasi tentang nomor telepon yang digunakan untuk layanan SMS pesan dalam Akun AWS
- `sms-voice:DescribeOptedOutNumbers`— Memungkinkan Anda untuk melihat atau mengambil daftar nomor telepon yang telah memilih untuk tidak menerima SMS pesan dari Anda Akun AWS

`AmazonSNSReadOnlyAccess`contoh kebijakan

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "SNSReadOnlyAccess",
            "Effect": "Allow",
            "Action": [
                "sns:GetTopicAttributes",
                "sns:List*",
                "sns:CheckIfPhoneNumberIsOptedOut",
```

```
                "sns:GetEndpointAttributes",
                "sns:GetDataProtectionPolicy",
                "sns:GetPlatformApplicationAttributes",
                "sns:GetSMSAttributes",
                "sns:GetSMSSandboxAccountStatus",
                "sns:GetSubscriptionAttributes"
            ],
            "Resource": "*"
        },
        {
            "Sid": "SMSAccessViaSNS",
            "Effect": "Allow",
            "Action": [
                "sms-voice:DescribeVerifiedDestinationNumbers",
                "sms-voice:DescribeAccountAttributes",
                "sms-voice:DescribeSpendLimits",
                "sms-voice:DescribePhoneNumbers",
                "sms-voice:DescribeOptedOutNumbers"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaLast": "sns.amazonaws.com"
                }
            }
        }
    ]
}
```

Untuk melihat izin kebijakan ini, lihat mazonSNSFullAkses di Referensi Kebijakan AWS Terkelola.

Amazon SNS memperbarui kebijakan AWS terkelola

Lihat detail tentang pembaruan kebijakan AWS terkelola untuk Amazon SNS sejak layanan ini mulai melacak perubahan ini. Untuk peringatan otomatis tentang perubahan pada halaman ini, berlangganan RSS umpan di halaman riwayat SNS Dokumen Amazon.

Perubahan	Deskripsi	Tanggal
AmazonSNSFullAccess — Permbaruan ke kebijakan yang sudah ada	Amazon SNS menambahkan izin baru untuk memungkinkan akses penuh ke Amazon SNS menggunakan file. AWS Management Console	09/24/2024
A mazonSNSRead OnlyAccess — Perbarui ke kebijakan yang ada	Amazon SNS menambahkan izin baru untuk memungkinkan akses hanya-baca ke Amazon SNS menggunakan file. AWS Management Console	09/24/2024
Amazon SNS mulai melacak perubahan	Amazon SNS mulai melacak perubahan untuk kebijakan yang AWS dikelola.	08/27/2024

Tindakan kebijakan untuk Amazon SNS

Mendukung tindakan kebijakan: Ya

Administrator dapat menggunakan AWS JSON kebijakan untuk menentukan siapa yang memiliki akses ke apa. Yaitu, principal dapat melakukan tindakan pada suatu sumber daya, dan dalam suatu syarat.

`Action`Elemen JSON kebijakan menjelaskan tindakan yang dapat Anda gunakan untuk mengizinkan atau menolak akses dalam kebijakan. Tindakan kebijakan biasanya memiliki nama yang sama dengan AWS API operasi terkait. Ada beberapa pengecualian, seperti tindakan khusus izin yang tidak memiliki operasi yang cocok. API Ada juga beberapa operasi yang memerlukan beberapa tindakan dalam suatu kebijakan. Tindakan tambahan ini disebut tindakan dependen.

Menyertakan tindakan dalam kebijakan untuk memberikan izin untuk melakukan operasi terkait.

Untuk melihat daftar SNS tindakan Amazon, lihat Sumber Daya yang Ditentukan oleh Layanan Pemberitahuan Sederhana Amazon di Referensi Otorisasi Layanan.

Tindakan kebijakan di Amazon SNS menggunakan awalan berikut sebelum tindakan:

```
sns
```

Untuk menetapkan secara spesifik beberapa tindakan dalam satu pernyataan, pisahkan tindakan tersebut dengan koma.

```
"Action": [
      "sns:action1",
      "sns:action2"
         ]
```

Untuk melihat contoh kebijakan SNS berbasis identitas Amazon, lihat. Contoh kebijakan berbasis identitas untuk Amazon Simple Notification Service

Sumber daya kebijakan untuk Amazon SNS

Mendukung sumber daya kebijakan: Ya

Administrator dapat menggunakan AWS JSON kebijakan untuk menentukan siapa yang memiliki akses ke apa. Yaitu, principal dapat melakukan tindakan pada suatu sumber daya, dan dalam suatu syarat.

Elemen `Resource` JSON kebijakan menentukan objek atau objek yang tindakan tersebut berlaku. Pernyataan harus menyertakan elemen `Resource` atau `NotResource`. Sebagai praktik terbaik, tentukan sumber daya menggunakan Amazon Resource Name (ARN). Anda dapat melakukan ini untuk tindakan yang mendukung jenis sumber daya tertentu, yang dikenal sebagai izin tingkat sumber daya.

Untuk tindakan yang tidak mendukung izin di tingkat sumber daya, misalnya operasi pencantuman, gunakan wildcard (*) untuk menunjukkan bahwa pernyataan tersebut berlaku untuk semua sumber daya.

```
"Resource": "*"
```

Untuk melihat daftar jenis SNS sumber daya Amazon dan jenisnyaARNs, lihat Tindakan yang Ditentukan oleh Layanan Pemberitahuan Sederhana Amazon di Referensi Otorisasi Layanan. Untuk mempelajari tindakan yang dapat Anda tentukan ARN dari setiap sumber daya, lihat Sumber Daya yang Ditentukan oleh Amazon Simple Notification Service.

Untuk melihat contoh kebijakan SNS berbasis identitas Amazon, lihat. Contoh kebijakan berbasis identitas untuk Amazon Simple Notification Service

Kunci kondisi kebijakan untuk Amazon SNS

Mendukung kunci kondisi kebijakan khusus layanan: Ya

Administrator dapat menggunakan AWS JSON kebijakan untuk menentukan siapa yang memiliki akses ke apa. Yaitu, di mana utama dapat melakukan tindakan pada sumber daya, dan dalam kondisi apa.

Elemen `Condition` (atau blok `Condition`) akan memungkinkan Anda menentukan kondisi yang menjadi dasar suatu pernyataan berlaku. Elemen `Condition` bersifat opsional. Anda dapat membuat ekspresi bersyarat yang menggunakan operator kondisi, misalnya sama dengan atau kurang dari, untuk mencocokkan kondisi dalam kebijakan dengan nilai-nilai yang diminta.

Jika Anda menentukan beberapa elemen `Condition` dalam sebuah pernyataan, atau beberapa kunci dalam elemen `Condition` tunggal, maka AWS akan mengevaluasinya menggunakan operasi `AND` logis. Jika Anda menentukan beberapa nilai untuk satu kunci kondisi, AWS mengevaluasi kondisi menggunakan `OR` operasi logis. Semua kondisi harus dipenuhi sebelum izin pernyataan diberikan.

Anda juga dapat menggunakan variabel placeholder saat menentukan kondisi. Misalnya, Anda dapat memberikan izin IAM pengguna untuk mengakses sumber daya hanya jika ditandai dengan nama IAM pengguna mereka. Untuk informasi selengkapnya, lihat elemen IAM kebijakan: variabel dan tag di Panduan IAM Pengguna.

AWS mendukung kunci kondisi global dan kunci kondisi khusus layanan. Untuk melihat semua kunci kondisi AWS global, lihat kunci konteks kondisi AWS global di Panduan IAM Pengguna.

Untuk melihat daftar kunci SNS kondisi Amazon, lihat Kunci Kondisi untuk Layanan Pemberitahuan Sederhana Amazon di Referensi Otorisasi Layanan. Untuk mempelajari tindakan dan sumber daya yang dapat Anda gunakan kunci kondisi, lihat Sumber Daya yang Ditentukan oleh Amazon Simple Notification Service.

Untuk melihat contoh kebijakan SNS berbasis identitas Amazon, lihat. Contoh kebijakan berbasis identitas untuk Amazon Simple Notification Service

ACLsdi Amazon SNS

MendukungACLs: Tidak

Access control lists (ACLs) mengontrol prinsipal mana (anggota akun, pengguna, atau peran) yang memiliki izin untuk mengakses sumber daya. ACLsmirip dengan kebijakan berbasis sumber daya, meskipun mereka tidak menggunakan format dokumen kebijakan. JSON

ABACdengan Amazon SNS

Mendukung ABAC (tag dalam kebijakan): Sebagian

Attribute-based access control (ABAC) adalah strategi otorisasi yang mendefinisikan izin berdasarkan atribut. Dalam AWS, atribut ini disebut tag. Anda dapat melampirkan tag ke IAM entitas (pengguna atau peran) dan ke banyak AWS sumber daya. Menandai entitas dan sumber daya adalah langkah pertama dari. ABAC Kemudian Anda merancang ABAC kebijakan untuk mengizinkan operasi ketika tag prinsipal cocok dengan tag pada sumber daya yang mereka coba akses.

ABACmembantu dalam lingkungan yang berkembang pesat dan membantu dengan situasi di mana manajemen kebijakan menjadi rumit.

Untuk mengendalikan akses berdasarkan tag, berikan informasi tentang tag di elemen kondisi dari kebijakan menggunakan kunci kondisi `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, atau `aws:TagKeys`.

Jika sebuah layanan mendukung ketiga kunci kondisi untuk setiap jenis sumber daya, nilainya adalah Ya untuk layanan tersebut. Jika suatu layanan mendukung ketiga kunci kondisi untuk hanya beberapa jenis sumber daya, nilainya adalah Parsial.

Untuk informasi selengkapnyaABAC, lihat Menentukan izin dengan ABAC otorisasi di IAMPanduan Pengguna. Untuk melihat tutorial dengan langkah-langkah penyiapanABAC, lihat Menggunakan kontrol akses berbasis atribut (ABAC) di IAMPanduan Pengguna.

Menggunakan kredensi sementara dengan Amazon SNS

Mendukung kredensi sementara: Ya

Beberapa Layanan AWS tidak berfungsi saat Anda masuk menggunakan kredensi sementara. Untuk informasi tambahan, termasuk yang Layanan AWS bekerja dengan kredensyal sementara, lihat Layanan AWS yang berfungsi IAM di IAMPanduan Pengguna.

Anda menggunakan kredensyal sementara jika Anda masuk AWS Management Console menggunakan metode apa pun kecuali nama pengguna dan kata sandi. Misalnya, ketika Anda

mengakses AWS menggunakan link sign-on (SSO) tunggal perusahaan Anda, proses tersebut secara otomatis membuat kredensi sementara. Anda juga akan secara otomatis membuat kredensial sementara ketika Anda masuk ke konsol sebagai seorang pengguna lalu beralih peran. Untuk informasi selengkapnya tentang beralih peran, lihat Beralih dari pengguna ke IAM peran (konsol) di Panduan IAM Pengguna.

Anda dapat secara manual membuat kredensi sementara menggunakan atau. AWS CLI AWS API Anda kemudian dapat menggunakan kredensi sementara tersebut untuk mengakses. AWS AWS merekomendasikan agar Anda secara dinamis menghasilkan kredensi sementara alih-alih menggunakan kunci akses jangka panjang. Untuk informasi selengkapnya, lihat Kredensi keamanan sementara di. IAM

Izin utama lintas layanan untuk Amazon SNS

Mendukung sesi akses maju (FAS): Ya

Saat Anda menggunakan IAM pengguna atau peran untuk melakukan tindakan AWS, Anda dianggap sebagai prinsipal. Ketika Anda menggunakan beberapa layanan, Anda mungkin melakukan sebuah tindakan yang kemudian menginisiasi tindakan lain di layanan yang berbeda. FASmenggunakan izin dari pemanggilan utama Layanan AWS, dikombinasikan dengan permintaan Layanan AWS untuk membuat permintaan ke layanan hilir. FASPermintaan hanya dibuat ketika layanan menerima permintaan yang memerlukan interaksi dengan orang lain Layanan AWS atau sumber daya untuk menyelesaikannya. Dalam hal ini, Anda harus memiliki izin untuk melakukan kedua tindakan tersebut. Untuk detail kebijakan saat membuat FAS permintaan, lihat Meneruskan sesi akses.

Peran layanan untuk Amazon SNS

Mendukung peran layanan: Ya

Peran layanan adalah IAMperan yang diasumsikan layanan untuk melakukan tindakan atas nama Anda. IAMAdministrator dapat membuat, memodifikasi, dan menghapus peran layanan dari dalamIAM. Untuk informasi selengkapnya, lihat Membuat peran untuk mendelegasikan izin ke Layanan AWS dalam IAMPanduan Pengguna.

> **Warning**
>
> Mengubah izin untuk peran layanan dapat merusak SNS fungsionalitas Amazon. Edit peran layanan hanya jika Amazon SNS memberikan panduan untuk melakukannya.

Peran terkait layanan untuk Amazon SNS

Mendukung peran terkait layanan: Tidak

Peran terkait layanan adalah jenis peran layanan yang ditautkan ke. Layanan AWS Layanan tersebut dapat menjalankan peran untuk melakukan tindakan atas nama Anda. Peran terkait layanan muncul di Anda Akun AWS dan dimiliki oleh layanan. IAMAdministrator dapat melihat, tetapi tidak mengedit izin untuk peran terkait layanan.

Untuk detail tentang membuat atau mengelola peran terkait layanan, lihat AWS layanan yang berfungsi dengannya. IAM Cari layanan dalam tabel yang memiliki `Yes` di kolom Peran terkait layanan. Pilih tautan Ya untuk melihat dokumentasi peran terkait layanan untuk layanan tersebut.

Contoh kebijakan berbasis identitas untuk Amazon Simple Notification Service

Secara default, pengguna dan peran tidak memiliki izin untuk membuat atau memodifikasi SNS sumber daya Amazon. Mereka juga tidak dapat melakukan tugas dengan menggunakan AWS Management Console, AWS Command Line Interface (AWS CLI), atau AWS API. Untuk memberikan izin kepada pengguna untuk melakukan tindakan pada sumber daya yang mereka butuhkan, IAM administrator dapat membuat IAM kebijakan. Administrator kemudian dapat menambahkan IAM kebijakan ke peran, dan pengguna dapat mengambil peran.

Untuk mempelajari cara membuat kebijakan IAM berbasis identitas menggunakan contoh dokumen kebijakan ini, lihat Membuat JSON IAM kebijakan (konsol) di Panduan Pengguna. IAM

Untuk detail tentang tindakan dan jenis sumber daya yang ditentukan oleh AmazonSNS, termasuk format ARNs untuk setiap jenis sumber daya, lihat Tindakan, Sumber Daya, dan Kunci Kondisi untuk Layanan Pemberitahuan Sederhana Amazon di Referensi Otorisasi Layanan.

Topik

- Praktik terbaik kebijakan
- Menggunakan SNS konsol Amazon
- Jenis-jenis kebijakan lain
- Berbagai jenis kebijakan
- Mengizinkan pengguna melihat izin mereka sendiri

Praktik terbaik kebijakan

Kebijakan berbasis identitas menentukan apakah seseorang dapat membuat, mengakses, atau menghapus SNS sumber daya Amazon di akun Anda. Tindakan ini membuat Akun AWS Anda dikenai biaya. Ketika Anda membuat atau mengedit kebijakan berbasis identitas, ikuti panduan dan rekomendasi ini:

- Mulailah dengan kebijakan AWS terkelola dan beralih ke izin hak istimewa paling sedikit — Untuk mulai memberikan izin kepada pengguna dan beban kerja Anda, gunakan kebijakan AWS terkelola yang memberikan izin untuk banyak kasus penggunaan umum. Mereka tersedia di Anda Akun AWS. Kami menyarankan Anda mengurangi izin lebih lanjut dengan menentukan kebijakan yang dikelola AWS pelanggan yang khusus untuk kasus penggunaan Anda. Untuk informasi selengkapnya, lihat kebijakan AWSAWS terkelola atau kebijakan terkelola untuk fungsi pekerjaan di Panduan IAM Pengguna.
- Menerapkan izin hak istimewa paling sedikit — Saat Anda menetapkan izin dengan IAM kebijakan, berikan hanya izin yang diperlukan untuk melakukan tugas. Anda melakukannya dengan mendefinisikan tindakan yang dapat diambil pada sumber daya tertentu dalam kondisi tertentu, yang juga dikenal sebagai izin dengan hak akses paling rendah. Untuk informasi selengkapnya tentang penggunaan IAM untuk menerapkan izin, lihat Kebijakan dan izin IAM di IAM Panduan Pengguna.
- Gunakan ketentuan dalam IAM kebijakan untuk membatasi akses lebih lanjut — Anda dapat menambahkan kondisi ke kebijakan Anda untuk membatasi akses ke tindakan dan sumber daya. Misalnya, Anda dapat menulis kondisi kebijakan untuk menentukan bahwa semua permintaan harus dikirim menggunakanSSL. Anda juga dapat menggunakan ketentuan untuk memberikan akses ke tindakan layanan jika digunakan melalui yang spesifik Layanan AWS, seperti AWS CloudFormation. Untuk informasi selengkapnya, lihat elemen IAM JSON kebijakan: Kondisi dalam Panduan IAM Pengguna.
- Gunakan IAM Access Analyzer untuk memvalidasi IAM kebijakan Anda guna memastikan izin yang aman dan fungsional — IAM Access Analyzer memvalidasi kebijakan baru dan yang sudah ada sehingga kebijakan tersebut mematuhi bahasa IAM kebijakan () JSON dan praktik terbaik. IAM IAMAccess Analyzer menyediakan lebih dari 100 pemeriksaan kebijakan dan rekomendasi yang dapat ditindaklanjuti untuk membantu Anda membuat kebijakan yang aman dan fungsional. Untuk informasi selengkapnya, lihat Memvalidasi kebijakan dengan IAM Access Analyzer di IAMPanduan Pengguna.
- Memerlukan otentikasi multi-faktor (MFA) - Jika Anda memiliki skenario yang mengharuskan IAM pengguna atau pengguna root di Anda Akun AWS, aktifkan MFA untuk keamanan tambahan.

Untuk meminta MFA kapan API operasi dipanggil, tambahkan MFA kondisi ke kebijakan Anda. Untuk informasi selengkapnya, lihat APIAkses aman dengan MFA di Panduan IAM Pengguna.

Untuk informasi selengkapnya tentang praktik terbaik diIAM, lihat Praktik terbaik keamanan IAM di Panduan IAM Pengguna.

Menggunakan SNS konsol Amazon

Untuk mengakses konsol Amazon Simple Notification Service, Anda harus memiliki set izin minimum. Izin ini harus memungkinkan Anda untuk membuat daftar dan melihat detail tentang SNS sumber daya Amazon di Anda Akun AWS. Jika Anda membuat kebijakan berbasis identitas yang lebih ketat daripada izin minimum yang diperlukan, konsol tidak akan berfungsi sebagaimana mestinya untuk entitas (pengguna atau peran) dengan kebijakan tersebut.

Anda tidak perlu mengizinkan izin konsol minimum untuk pengguna yang melakukan panggilan hanya ke AWS CLI atau. AWS API Sebagai gantinya, izinkan akses hanya ke tindakan yang cocok dengan API operasi yang mereka coba lakukan.

Untuk memastikan bahwa pengguna dan peran masih dapat menggunakan SNS konsol Amazon, lampirkan juga Amazon SNS *`ConsoleAccess`* atau kebijakan *`ReadOnly`* AWS terkelola ke entitas. Untuk informasi selengkapnya, lihat Menambahkan izin ke pengguna di Panduan IAM Pengguna.

Jenis-jenis kebijakan lain

AWS mendukung jenis kebijakan tambahan yang kurang umum. Jenis-jenis kebijakan ini dapat mengatur izin maksimum yang diberikan kepada Anda oleh jenis kebijakan yang lebih umum.

- Batas izin — Batas izin adalah fitur lanjutan tempat Anda menetapkan izin maksimum yang dapat diberikan oleh kebijakan berbasis identitas kepada entitas (pengguna atau peran). IAM IAM Anda dapat menetapkan batasan izin untuk suatu entitas. Izin yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas milik entitas dan batasan izinnya. Kebijakan berbasis sumber daya yang menentukan pengguna atau peran dalam bidang `Principal` tidak dibatasi oleh batasan izin. Penolakan eksplisit dalam salah satu kebijakan ini akan menggantikan pemberian izin. Untuk informasi selengkapnya tentang batas izin, lihat Batas izin untuk IAM entitas di IAMPanduan Pengguna.
- Kebijakan kontrol layanan (SCPs) — SCPs adalah JSON kebijakan yang menentukan izin maksimum untuk organisasi atau unit organisasi (OU) di AWS Organizations. AWS Organizations adalah layanan untuk mengelompokkan dan mengelola secara terpusat beberapa Akun

AWS yang dimiliki bisnis Anda. Jika Anda mengaktifkan semua fitur dalam organisasi, Anda dapat menerapkan kebijakan kontrol layanan (SCPs) ke salah satu atau semua akun Anda. SCPMembatasi izin untuk entitas di akun anggota, termasuk setiap pengguna Akun AWS root. Untuk informasi selengkapnya tentang Organizations danSCPs, lihat Kebijakan kontrol layanan di Panduan AWS Organizations Pengguna.

- Kebijakan sesi – Kebijakan sesi adalah kebijakan lanjutan yang Anda berikan sebagai parameter ketika Anda membuat sesi sementara secara programatis untuk peran atau pengguna terfederasi. Izin sesi yang dihasilkan adalah perpotongan antara kebijakan berbasis identitas pengguna atau peran dan kebijakan sesi. Izin juga bisa datang dari kebijakan berbasis sumber daya. Penolakan secara tegas dalam salah satu kebijakan ini membatalkan izin. Untuk informasi selengkapnya, lihat Kebijakan sesi di Panduan IAM Pengguna.

Berbagai jenis kebijakan

Ketika beberapa jenis kebijakan berlaku pada suatu permintaan, izin yang dihasilkan lebih rumit untuk dipahami. Untuk mempelajari cara AWS menentukan apakah akan mengizinkan permintaan saat beberapa jenis kebijakan terlibat, lihat Logika evaluasi kebijakan di Panduan IAM Pengguna.

Mengizinkan pengguna melihat izin mereka sendiri

Contoh ini menunjukkan cara Anda membuat kebijakan yang memungkinkan IAM pengguna melihat kebijakan sebaris dan terkelola yang dilampirkan pada identitas pengguna mereka. Kebijakan ini mencakup izin untuk menyelesaikan tindakan ini di konsol atau secara terprogram menggunakan atau. AWS CLI AWS API

```
{
    "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": "*"
        }
    ]
}
```

Kebijakan berbasis identitas untuk Amazon SNS

Mendukung kebijakan berbasis identitas: Ya

Kebijakan berbasis identitas adalah dokumen kebijakan JSON izin yang dapat Anda lampirkan ke identitas, seperti pengguna, grup IAM pengguna, atau peran. Kebijakan ini mengontrol jenis tindakan yang dapat dilakukan oleh pengguna dan peran, di sumber daya mana, dan berdasarkan kondisi seperti apa. Untuk mempelajari cara membuat kebijakan berbasis identitas, lihat Menentukan IAM izin khusus dengan kebijakan yang dikelola pelanggan di Panduan Pengguna. IAM

Dengan kebijakan IAM berbasis identitas, Anda dapat menentukan tindakan dan sumber daya yang diizinkan atau ditolak serta kondisi di mana tindakan diizinkan atau ditolak. Anda tidak dapat menentukan secara spesifik prinsipal dalam sebuah kebijakan berbasis identitas karena prinsipal berlaku bagi pengguna atau peran yang melekat kepadanya. Untuk mempelajari semua elemen yang dapat Anda gunakan dalam JSON kebijakan, lihat referensi elemen IAM JSON kebijakan di Panduan IAM Pengguna.

Contoh kebijakan berbasis identitas untuk Amazon SNS

Untuk melihat contoh kebijakan SNS berbasis identitas Amazon, lihat. Contoh kebijakan berbasis identitas untuk Amazon Simple Notification Service

Kebijakan berbasis sumber daya di Amazon SNS

Mendukung kebijakan berbasis sumber daya	Ya

Kebijakan berbasis sumber daya adalah dokumen JSON kebijakan yang Anda lampirkan ke sumber daya. Contoh kebijakan berbasis sumber daya adalah kebijakan kepercayaan IAM peran dan kebijakan bucket Amazon S3. Dalam layanan yang mendukung kebijakan berbasis sumber daya, administrator layanan dapat menggunakannya untuk mengontrol akses ke sumber daya tertentu. Untuk sumber daya tempat kebijakan dilampirkan, kebijakan menentukan tindakan apa yang dapat dilakukan oleh prinsipal tertentu pada sumber daya tersebut dan dalam kondisi apa. Anda harus menentukan prinsipal dalam kebijakan berbasis sumber daya. Prinsipal dapat mencakup akun, pengguna, peran, pengguna federasi, atau. Layanan AWS

Untuk mengaktifkan akses lintas akun, Anda dapat menentukan seluruh akun atau IAM entitas di akun lain sebagai prinsipal dalam kebijakan berbasis sumber daya. Menambahkan prinsipal akun silang ke kebijakan berbasis sumber daya hanya setengah dari membangun hubungan kepercayaan. Ketika prinsipal dan sumber daya berbeda Akun AWS, IAM administrator di akun tepercaya juga harus memberikan izin entitas utama (pengguna atau peran) untuk mengakses sumber daya. Mereka memberikan izin dengan melampirkan kebijakan berbasis identitas kepada entitas. Namun, jika kebijakan berbasis sumber daya memberikan akses ke prinsipal dalam akun yang sama, tidak diperlukan kebijakan berbasis identitas tambahan. Untuk informasi selengkapnya, lihat Akses sumber daya lintas akun IAM di Panduan IAM Pengguna.

Menggunakan kebijakan berbasis identitas dengan Amazon SNS

Topik

- IAMdan SNS kebijakan Amazon bersama-sama
- ARNFormat SNS sumber daya Amazon
- SNSAPITindakan Amazon
- Kunci SNS kebijakan Amazon
- Contoh kebijakan untuk Amazon SNS

Amazon Simple Notification Service terintegrasi dengan AWS Identity and Access Management (IAM) sehingga Anda dapat menentukan SNS tindakan Amazon mana yang Akun AWS dapat

dilakukan pengguna dengan SNS sumber daya Amazon. Anda dapat menentukan topik tertentu dalam kebijakan. Misalnya, Anda dapat menggunakan variabel saat membuat IAM kebijakan yang memberikan izin kepada pengguna tertentu di organisasi Anda untuk menggunakan `Publish` tindakan dengan topik tertentu di organisasi Anda Akun AWS. Untuk informasi selengkapnya, lihat Variabel Kebijakan dalam IAM panduan Menggunakan.

> **Important**
>
> Menggunakan Amazon SNS dengan IAM tidak mengubah cara Anda menggunakan AmazonSNS. Tidak ada perubahan pada SNS tindakan Amazon, dan tidak ada SNS tindakan Amazon baru yang terkait dengan pengguna dan kontrol akses.

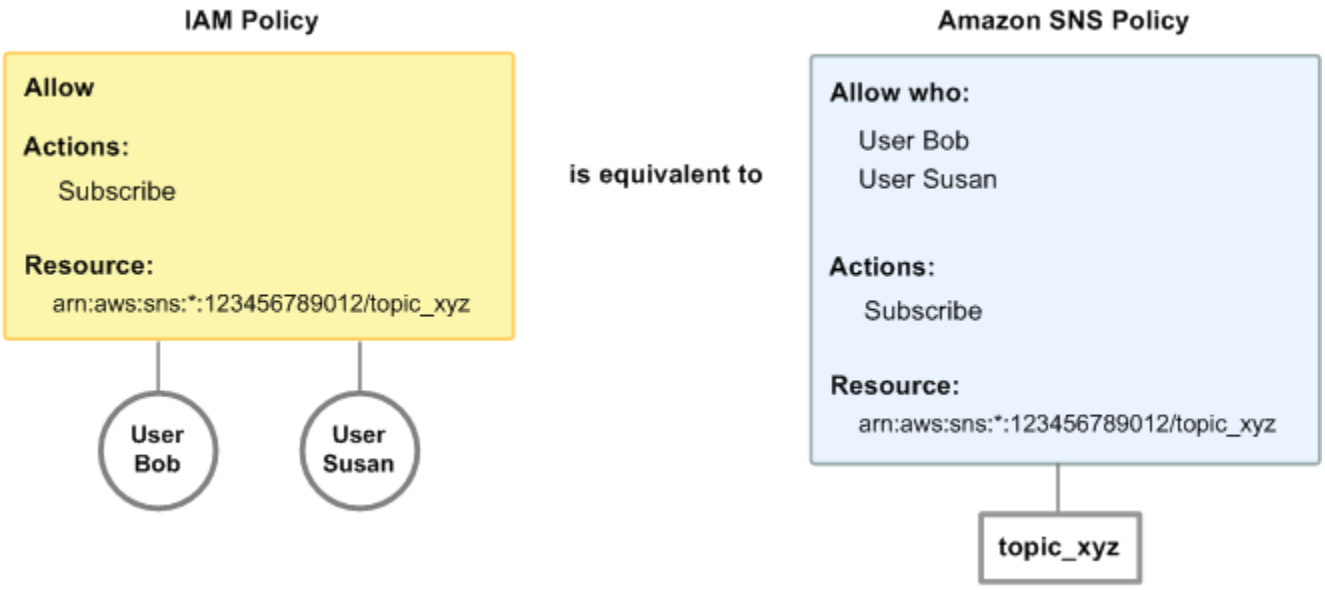
Untuk contoh kebijakan yang mencakup SNS tindakan dan sumber daya Amazon, lihatContoh kebijakan untuk Amazon SNS.

IAMdan SNS kebijakan Amazon bersama-sama

Anda menggunakan IAM kebijakan untuk membatasi akses pengguna ke SNS tindakan dan topik Amazon. IAMKebijakan dapat membatasi akses hanya untuk pengguna dalam AWS akun Anda, bukan untuk yang lain Akun AWS.

Anda menggunakan SNS kebijakan Amazon dengan topik tertentu untuk membatasi siapa yang dapat bekerja dengan topik itu (misalnya, siapa yang dapat mempublikasikan pesan ke sana, siapa yang dapat berlangganan, dll.). SNSKebijakan Amazon dapat memberikan akses ke orang lain Akun AWS, atau kepada pengguna di dalam milik Anda Akun AWS.

Untuk memberikan izin kepada pengguna untuk SNS topik Amazon, Anda dapat menggunakan IAM kebijakan, SNS kebijakan Amazon, atau keduanya. Umumnya, Anda dapat mencapai hasil yang sama dengan baik. Misalnya, diagram berikut menunjukkan IAM kebijakan dan SNS kebijakan Amazon yang setara. IAMKebijakan ini memungkinkan SNS `Subscribe` tindakan Amazon untuk topik yang disebut topic_xyz dalam IAM Kebijakan Anda Akun AWS dilampirkan ke pengguna Bob dan Susan (yang berarti bahwa Bob dan Susan memiliki izin yang dinyatakan dalam kebijakan). SNSKebijakan Amazon juga memberikan izin kepada Bob dan Susan `Subscribe` untuk mengakses topic_xyz.



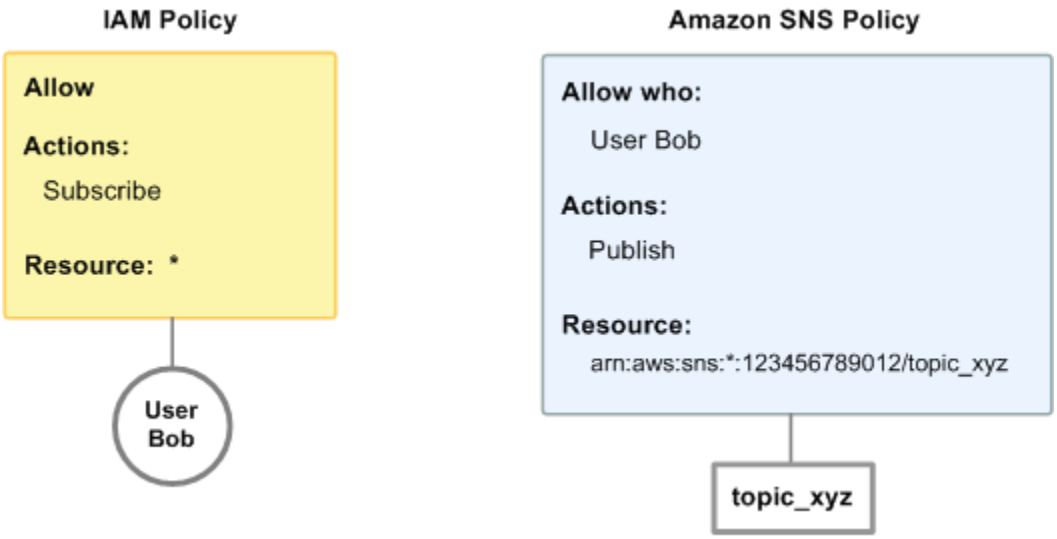
> **Note**
>
> Contoh sebelumnya menunjukkan kebijakan sederhana tanpa syarat. Anda bisa menentukan syarat tertentu dalam kebijakan yang mana pun dan mendapatkan hasil yang sama.

Ada satu perbedaan antara AWS IAM dan SNS kebijakan Amazon: Sistem SNS kebijakan Amazon memungkinkan Anda memberikan izin kepada yang lain Akun AWS, sedangkan IAM kebijakan tidak.

Terserah Anda bagaimana Anda menggunakan kedua sistem bersama-sama untuk mengelola izin Anda, berdasarkan kebutuhan Anda. Contoh berikut menunjukkan cara sistem dua kebijakan bekerja sama.

Example 1

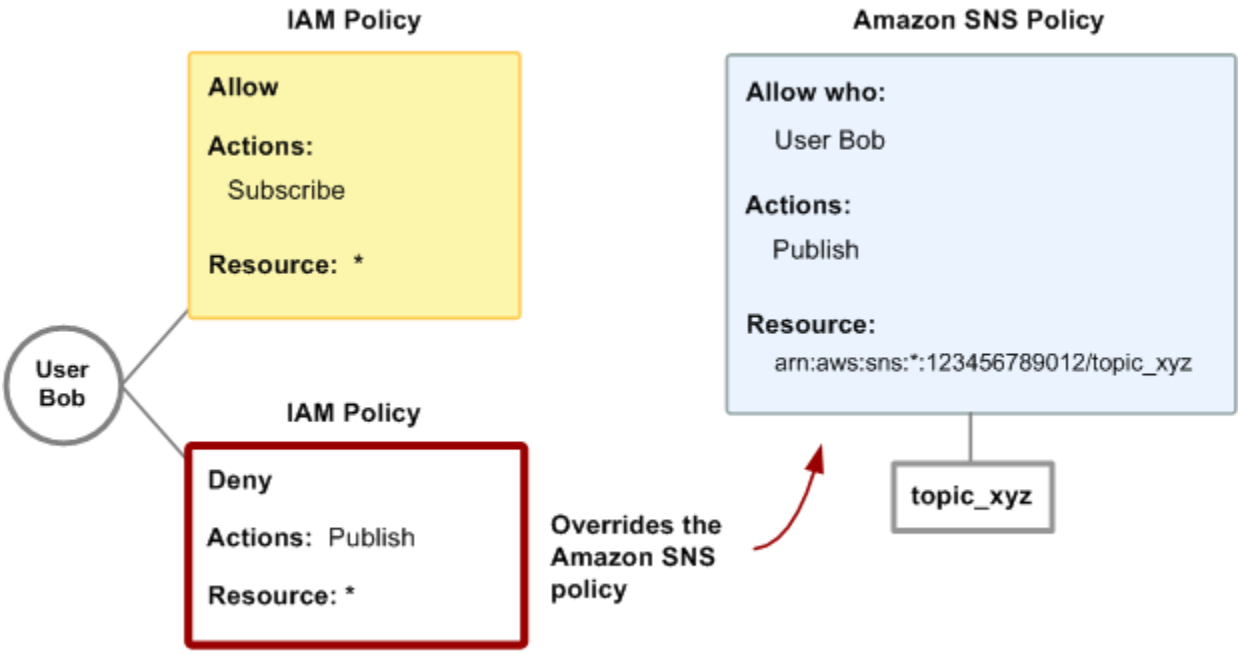
Dalam contoh ini, IAM kebijakan dan SNS kebijakan Amazon berlaku untuk Bob. IAMKebijakan tersebut memberinya izin untuk `Subscribe` topik apa pun, sedangkan SNS kebijakan Amazon memberinya izin untuk menggunakan `Publish` topik tertentu (topic_xyz). Akun AWS Diagram berikut menggambarkan konsep.



Jika Bob mengirim permintaan untuk berlangganan topik apa pun di AWS akun, IAM kebijakan tersebut akan memungkinkan tindakan tersebut. Jika Bob mengirim permintaan untuk mempublikasikan pesan ke topic_xyz, SNS kebijakan Amazon akan mengizinkan tindakan tersebut.

Example 2

Dalam contoh ini, kita mengembangkan contoh 1 (ada dua kebijakan yang diterapkan pada Bob). Katakanlah bahwa Bob memublikasikan pesan ke topic_xyz yang seharusnya tidak dia lakukan, jadi Anda ingin sepenuhnya menghapus kemampuannya memublikasikan ke topik. Hal termudah untuk dilakukan adalah menambahkan IAM kebijakan yang menolaknya mengakses `Publish` tindakan pada semua topik. Kebijakan ketiga ini mengesampingkan SNS kebijakan Amazon yang awalnya memberinya izin untuk mempublikasikan ke topic_xyz, karena penolakan eksplisit selalu mengesampingkan izin (untuk informasi lebih lanjut tentang logika evaluasi kebijakan, lihat). Logika evaluasi Diagram berikut menggambarkan konsep.



Untuk contoh kebijakan yang mencakup SNS tindakan dan sumber daya Amazon, lihatContoh kebijakan untuk Amazon SNS. Untuk informasi lebih lanjut tentang menulis SNS kebijakan Amazon, buka dokumentasi teknis untuk Amazon SNS.

ARNFormat SNS sumber daya Amazon

Untuk AmazonSNS, topik adalah satu-satunya jenis sumber daya yang dapat Anda tentukan dalam kebijakan. Berikut ini adalah format Amazon Resource Name (ARN) untuk topik.

```
arn:aws:sns:region:account_ID:topic_name
```

Untuk informasi lebih lanjut tentangARNs, buka Panduan IAM Pengguna. ARNs

Example

Berikut ini adalah ARN untuk topik yang dinamai MyTopic di wilayah us-east-2, milik 123456789012. Akun AWS

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

Example

Jika Anda memiliki topik yang disebutkan MyTopic di setiap Wilayah berbeda yang SNS didukung Amazon, Anda dapat menentukan topik dengan yang berikut iniARN.

```
arn:aws:sns:*:123456789012:MyTopic
```

Anda dapat menggunakan wildcard * dan ? dalam nama topik. Sebagai contoh, yang berikut ini dapat merujuk pada semua topik yang dibuat oleh Bob yang telah dia awali dengan bob_.

```
arn:aws:sns:*:123456789012:bob_*
```

Sebagai kenyamanan bagi Anda, saat Anda membuat topik, Amazon SNS mengembalikan topik sebagai tanggapan. ARN

SNSAPITindakan Amazon

Dalam IAM kebijakan, Anda dapat menentukan tindakan apa pun yang SNS ditawarkan Amazon. Namun, `Unsubscribe` tindakan `ConfirmSubscription` dan tindakan tidak memerlukan otentikasi, yang berarti bahwa meskipun Anda menentukan tindakan tersebut dalam kebijakan, tidak IAM akan membatasi akses pengguna ke tindakan tersebut.

Setiap tindakan yang Anda tentukan dalam kebijakan harus diawali dengan string huruf kecil `sns:`. Untuk menentukan semua SNS tindakan Amazon, misalnya, Anda akan menggunakan`sns:*`. Untuk daftar tindakan, buka APIReferensi Layanan Pemberitahuan Sederhana Amazon.

Kunci SNS kebijakan Amazon

Amazon SNS menerapkan kunci kebijakan AWS lebar berikut, ditambah beberapa kunci khusus layanan.

Untuk daftar kunci kondisi yang didukung oleh masing-masing Layanan AWS, lihat Tindakan, sumber daya, dan kunci kondisi Layanan AWS di Panduan IAM Pengguna. Untuk daftar kunci kondisi yang dapat digunakan dalam beberapa Layanan AWS, lihat kunci konteks kondisi AWS global di Panduan IAM Pengguna.

Amazon SNS menggunakan kunci khusus layanan berikut. Gunakan kunci ini dalam kebijakan yang membatasi akses ke permintaan `Subscribe`.

- sns:endpoint— Alamat emailURL, atau ARN dari `Subscribe` permintaan atau langganan yang dikonfirmasi sebelumnya. Gunakan dengan syarat string (lihat Contoh kebijakan untuk Amazon SNS) untuk membatasi akses ke titik akhir tertentu (misalnya, *@yourcompany.com).
- sns:protocol—Nilai `protocol` dari permintaan `Subscribe` atau langganan yang telah dikonfirmasi sebelumnya. Gunakan dengan syarat string (lihat Contoh kebijakan untuk Amazon SNS) untuk membatasi publikasi ke protokol pengiriman tertentu (misalnya, https).

Contoh kebijakan untuk Amazon SNS

Bagian ini menunjukkan beberapa kebijakan sederhana untuk mengontrol akses pengguna ke AmazonSNS.

> **Note**
>
> Di masa depan, Amazon SNS mungkin menambahkan tindakan baru yang secara logis harus dimasukkan dalam salah satu kebijakan berikut, berdasarkan tujuan yang dinyatakan kebijakan tersebut.

Example 1: Memungkinkan grup untuk membuat dan mengelola topik

Dalam contoh ini, kami membuat kebijakan yang memberikan akses ke `CreateTopic`, `ListTopics` `SetTopicAttributes`, dan `DeleteTopic`.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": ["sns:CreateTopic", "sns:ListTopics", "sns:SetTopicAttributes",
 "sns:DeleteTopic"],
    "Resource": "*"
  }]
}
```

Example 2: Memungkinkan grup IT untuk memublikasikan pesan ke topik tertentu

Dalam contoh ini, kami membuat grup untuk IT, dan menetapkan kebijakan yang memberikan akses ke `Publish` pada topik tertentu yang diinginkan.

```
{
```

```
  "Statement": [{
    "Effect": "Allow",
    "Action": "sns:Publish",
    "Resource": "arn:aws:sns:*:123456789012:MyTopic"
  }]
}
```

Example 3: Berikan pengguna Akun AWS kemampuan untuk berlangganan topik

Dalam contoh ini, kami membuat kebijakan yang memberikan akses ke tindakan `Subscribe`, dengan syarat pencocokan string untuk kunci kebijakan `sns:Protocol` dan `sns:Endpoint`.

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action": ["sns:Subscribe"],
    "Resource": "*",
    "Condition": {
      "StringLike": {
        "SNS:Endpoint": "*@example.com"
      },
      "StringEquals": {
        "sns:Protocol": "email"
      }
    }
  }]
}
```

Example 4: Mengizinkan mitra memublikasikan pesan ke topik tertentu

Anda dapat menggunakan SNS kebijakan Amazon atau IAM kebijakan untuk mengizinkan mitra mempublikasikan ke topik tertentu. Jika pasangan Anda memiliki Akun AWS, mungkin lebih mudah untuk menggunakan SNS kebijakan Amazon. Namun, siapa pun di perusahaan mitra yang memiliki kredensyal AWS keamanan dapat mempublikasikan pesan ke topik tersebut. Contoh ini mengasumsikan bahwa Anda ingin membatasi akses ke orang (atau aplikasi) tertentu. Untuk melakukan ini, Anda perlu memperlakukan mitra seperti pengguna di dalam perusahaan Anda sendiri, dan menggunakan IAM kebijakan alih-alih SNS kebijakan Amazon.

Untuk contoh ini, kami membuat grup bernama WidgetCo yang mewakili perusahaan mitra; kami membuat pengguna untuk orang tertentu (atau aplikasi) di perusahaan mitra yang membutuhkan akses; dan kemudian kami menempatkan pengguna dalam grup.

Kami kemudian melampirkan kebijakan yang memberikan `Publish` akses grup pada topik tertentu bernama WidgetPartnerTopic.

Kami juga ingin mencegah WidgetCo grup melakukan hal lain dengan topik, jadi kami menambahkan pernyataan yang menolak izin untuk SNS tindakan Amazon selain topik apa pun selain `Publish` topik apa pun selain WidgetPartnerTopic. Ini diperlukan hanya jika ada kebijakan luas di tempat lain dalam sistem yang memberi pengguna akses luas ke AmazonSNS.

```
{
  "Statement": [{
      "Effect": "Allow",
      "Action": "sns:Publish",
      "Resource": "arn:aws:sns:*:123456789012:WidgetPartnerTopic"
    },
    {
      "Effect": "Deny",
      "NotAction": "sns:Publish",
      "NotResource": "arn:aws:sns:*:123456789012:WidgetPartnerTopic"
    }
  ]
}
```

Menggunakan kredensi keamanan sementara dengan Amazon SNS

AWS Identity and Access Management (IAM) memungkinkan Anda untuk memberikan kredensi keamanan sementara kepada pengguna dan aplikasi yang membutuhkan akses ke sumber daya Anda AWS . Kredensyal keamanan sementara ini terutama digunakan untuk IAM peran dan akses federasi melalui protokol standar industri seperti dan SAML OpenID Connect (). OIDC

Untuk mengelola akses ke AWS sumber daya secara efektif, penting untuk memahami konsep-konsep kunci berikut:

- IAMPeran — Peran digunakan untuk mendelegasikan akses ke AWS sumber daya. Peran dapat diasumsikan oleh entitas seperti EC2 instans Amazon, fungsi Lambda, atau pengguna dari yang lain. Akun AWS
- Pengguna Federasi — Ini adalah pengguna yang diautentikasi melalui penyedia identitas eksternal (IdPs) menggunakan SAML atau. OIDC Akses federasi direkomendasikan untuk pengguna manusia, sedangkan IAM peran harus digunakan untuk aplikasi perangkat lunak.

- Peran Di Mana Saja — Untuk aplikasi eksternal yang memerlukan AWS akses, Anda dapat menggunakan IAM Peran Di Mana Saja untuk mengelola akses dengan aman tanpa membuat kredensyal jangka panjang.

Anda dapat menggunakan kredensi keamanan sementara untuk mengajukan permintaan ke Amazon. SNS APIPustaka SDKs dan menghitung tanda tangan yang diperlukan menggunakan kredensyal ini untuk mengautentikasi permintaan Anda. Permintaan dengan kredensi kedaluwarsa akan ditolak oleh Amazon. SNS

Untuk informasi selengkapnya tentang kredensyal keamanan sementara, lihat Menggunakan IAM peran dan Menyediakan akses ke pengguna yang diautentikasi secara eksternal (federasi identitas) dalam Panduan Pengguna. IAM

Example HTTPScontoh permintaan

Contoh berikut menunjukkan cara mengautentikasi SNS permintaan Amazon menggunakan kredensil keamanan sementara yang diperoleh dari (). AWS Security Token Service STS

```
https://sns.us-east-2.amazonaws.com/
?Action=CreateTopic
&Name=My-Topic
&SignatureVersion=4
&SignatureMethod=AWS4-HMAC-SHA256
&Timestamp=2023-07-05T12:00:00Z
&X-Amz-Security-Token=SecurityTokenValue
&X-Amz-Date=20230705T120000Z
&X-Amz-Credential=<your-access-key-id>/20230705/us-east-2/sns/aws4_request
&X-Amz-SignedHeaders=host
&X-Amz-Signature=<signature-value>
```

Langkah-langkah untuk mengautentikasi permintaan

1. Dapatkan Kredensyal Keamanan Sementara — Gunakan AWS STS untuk mengambil peran atau mendapatkan kredensyal pengguna gabungan. Ini akan memberi Anda ID kunci akses, kunci akses rahasia, dan token keamanan.
2. Buat Permintaan — Sertakan parameter yang diperlukan untuk SNS tindakan Amazon Anda (misalnya, CreateTopic), dan pastikan Anda menggunakannya HTTPS untuk komunikasi yang aman.

3. Tanda tangani Permintaan — Gunakan proses AWS Signature Version 4 untuk menandatangani permintaan Anda. Ini melibatkan pembuatan permintaan kanonik, string-to-sign, dan kemudian menghitung tanda tangan. Untuk selengkapnya tentang AWS Tanda Tangan Versi 4, lihat Menggunakan Tanda Tangan Versi 4 yang masuk di Panduan EBS Pengguna Amazon.
4. Kirim Permintaan - Sertakan X-Amz-Security-Token di header permintaan Anda untuk meneruskan kredensi keamanan sementara ke Amazon. SNS

SNSAPIIzin Amazon: Referensi tindakan dan sumber daya

Daftar berikut memberikan informasi khusus untuk SNS implementasi kontrol akses Amazon:

- Setiap kebijakan harus mencakup hanya satu topik (ketika menulis kebijakan, jangan sertakan pernyataan yang mencakup topik yang berbeda)
- Setiap kebijakan harus memiliki `Id` kebijakan yang unik
- Setiap pernyataan dalam kebijakan harus memiliki `sid` pernyataan unik

Kuota kebijakan

Tabel berikut mencantumkan kuota maksimum untuk pernyataan kebijakan.

Nama	Kuota maksimum
Byte	30 kb
Pernyataan	100
Penanggung jawab	1 hingga 200 (0 tidak valid.)
Sumber Daya	1 (0 tidak valid. Nilai harus sesuai dengan topik kebijakan.) ARN

Tindakan SNS kebijakan Amazon yang valid

Amazon SNS mendukung tindakan yang ditunjukkan pada tabel berikut.

Tindakan	Deskripsi
SNS: AddPermission	Memberikan izin untuk menambahkan izin ke kebijakan topik.
SNS: DeleteTopic	Memberikan izin untuk menghapus topik.
SNS: GetDataProtectionPolicy	Memberikan izin untuk mengambil kebijakan perlindungan data topik.
SNS: GetTopicAttributes	Memberikan izin untuk menerima semua atribut topik.
SNS: ListSubscriptionsBy Topic	Memberikan izin untuk mengambil semua langganan untuk topik tertentu.
SNS: ListTagsForResource	Memberikan izin untuk mencantumkan semua tag yang ditambahkan ke topik tertentu.
SNS:Publish	Memberikan izin untuk mempublikasikan dan menerbitkan batch ke topik atau titik akhir. Untuk informasi selengkapnya, lihat Menerbitkan dan PublishBatchdi API Referensi Layanan Pemberitahuan Sederhana Amazon.
SNS: PutDataProtectionPolicy	Memberikan izin untuk menetapkan kebijakan perlindungan data topik.
SNS: RemovePermission	Memberikan izin untuk menghapus izin apa pun dalam kebijakan topik.
SNS: SetTopicAttributes	Memberikan izin untuk mengatur atribut topik.
sns:Subscribe	Memberikan izin untuk berlangganan topik.

Kunci khusus layanan

Amazon SNS menggunakan kunci khusus layanan berikut. Anda dapat menggunakan kunci ini dalam kebijakan yang membatasi akses ke permintaan `Subscribe`.

- sns:endpoint— Alamat emailURL, atau ARN dari `Subscribe` permintaan atau langganan yang dikonfirmasi sebelumnya. Gunakan dengan syarat string (lihat Contoh kebijakan untuk Amazon SNS) untuk membatasi akses ke titik akhir tertentu (misalnya, *@example.com).
- sns:protocol—Nilai `protocol` dari permintaan `Subscribe` atau langganan yang telah dikonfirmasi sebelumnya. Gunakan dengan syarat string (lihat Contoh kebijakan untuk Amazon SNS) untuk membatasi publikasi ke protokol pengiriman tertentu (misalnya, https).

> **Important**
>
> Saat Anda menggunakan kebijakan untuk mengontrol akses oleh SNS:Endpoint, ketahuilah bahwa DNS masalah dapat memengaruhi resolusi nama titik akhir di masa mendatang.

Memecahkan masalah identitas dan akses Amazon Simple Notification Service

Gunakan informasi berikut untuk membantu Anda mendiagnosis dan memperbaiki masalah umum yang mungkin Anda temui saat bekerja dengan Amazon SNS danIAM.

Topik

- Saya tidak berwenang untuk melakukan tindakan di Amazon SNS
- Saya tidak berwenang untuk melakukan iam: PassRole
- Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses SNS sumber daya Amazon saya

Saya tidak berwenang untuk melakukan tindakan di Amazon SNS

Jika Anda menerima pesan kesalahan bahwa Anda tidak memiliki otorisasi untuk melakukan tindakan, kebijakan Anda harus diperbarui agar Anda dapat melakukan tindakan tersebut.

Contoh kesalahan berikut terjadi ketika pengguna `mateojackson` mencoba menggunakan konsol untuk melihat detail tentang suatu sumber daya *`my-example-widget`* fiktif, tetapi tidak memiliki izin `sns:`*`GetWidget`* fiktif.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 sns:GetWidget on resource: my-example-widget
```

Dalam hal ini, kebijakan Mateo harus diperbarui untuk memungkinkannya mengakses *my-example-widget* sumber daya menggunakan `sns:GetWidget` tindakan.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya tidak berwenang untuk melakukan iam: PassRole

Jika Anda menerima kesalahan bahwa Anda tidak diizinkan untuk melakukan `iam:PassRole` tindakan, kebijakan Anda harus diperbarui agar Anda dapat meneruskan peran ke AmazonSNS.

Beberapa Layanan AWS memungkinkan Anda untuk meneruskan peran yang ada ke layanan tersebut alih-alih membuat peran layanan baru atau peran terkait layanan. Untuk melakukannya, Anda harus memiliki izin untuk meneruskan peran ke layanan.

Contoh kesalahan berikut terjadi ketika IAM pengguna bernama `marymajor` mencoba menggunakan konsol untuk melakukan tindakan di AmazonSNS. Namun, tindakan tersebut memerlukan layanan untuk mendapatkan izin yang diberikan oleh peran layanan. Mary tidak memiliki izin untuk meneruskan peran tersebut pada layanan.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

Dalam kasus ini, kebijakan Mary harus diperbarui agar dia mendapatkan izin untuk melakukan tindakan `iam:PassRole` tersebut.

Jika Anda memerlukan bantuan, hubungi AWS administrator Anda. Administrator Anda adalah orang yang memberi Anda kredensial masuk.

Saya ingin mengizinkan orang di luar saya Akun AWS untuk mengakses SNS sumber daya Amazon saya

Anda dapat membuat peran yang dapat digunakan pengguna di akun lain atau orang-orang di luar organisasi Anda untuk mengakses sumber daya Anda. Anda dapat menentukan siapa saja yang dipercaya untuk mengambil peran tersebut. Untuk layanan yang mendukung kebijakan berbasis sumber daya atau daftar kontrol akses (ACLs), Anda dapat menggunakan kebijakan tersebut untuk memberi orang akses ke sumber daya Anda.

Untuk mempelajari selengkapnya, periksa referensi berikut:

- Untuk mengetahui apakah Amazon SNS mendukung fitur-fitur ini, lihatBagaimana Amazon SNS bekerja dengan IAM.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda di seluruh sumber daya Akun AWS yang Anda miliki, lihat Menyediakan akses ke IAM pengguna lain Akun AWS yang Anda miliki di Panduan IAM Pengguna.
- Untuk mempelajari cara menyediakan akses ke sumber daya Anda kepada pihak ketiga Akun AWS, lihat Menyediakan akses yang Akun AWS dimiliki oleh pihak ketiga dalam Panduan IAM Pengguna.
- Untuk mempelajari cara menyediakan akses melalui federasi identitas, lihat Menyediakan akses ke pengguna yang diautentikasi secara eksternal (federasi identitas) di Panduan Pengguna. IAM
- Untuk mempelajari perbedaan antara menggunakan peran dan kebijakan berbasis sumber daya untuk akses lintas akun, lihat Akses sumber daya lintas akun di IAM Panduan Pengguna. IAM

Pencatatan dan pemantauan di Amazon SNS

Amazon SNS memungkinkan Anda untuk melacak dan memantau aktivitas pesan dengan mencatat API panggilan dengan CloudTrail dan memantau topik dengan CloudWatch. Alat ini membantu Anda mendapatkan wawasan tentang pengiriman pesan, memecahkan masalah, dan memastikan kesehatan alur kerja pesan Anda. Topik ini mencakup hal-hal berikut:

- Pencatatan SNS API panggilan Amazon menggunakan CloudTrail. Pencatatan ini memungkinkan Anda melacak tindakan yang dilakukan pada SNS topik Amazon Anda, seperti pembuatan topik, manajemen langganan, dan penerbitan pesan. Dengan menganalisis CloudTrail log, Anda dapat mengidentifikasi siapa yang membuat API permintaan spesifik dan kapan permintaan tersebut dibuat, membantu Anda mengaudit dan memecahkan masalah penggunaan Amazon SNS Anda.
- Memantau SNS topik Amazon menggunakan CloudWatch. CloudWatch menyediakan metrik yang memungkinkan Anda mengamati kinerja dan kesehatan SNS topik Amazon Anda secara real time. Siapkan alarm berdasarkan metrik ini, sehingga Anda dapat segera merespons anomali apa pun, seperti kegagalan pengiriman atau latensi pesan tinggi. Kemampuan pemantauan ini memastikan bahwa Anda dapat mempertahankan keandalan sistem pesan SNS berbasis Anda dengan secara proaktif menangani masalah potensial.

Pencatatan SNS API panggilan Amazon menggunakan CloudTrail

Amazon SNS terintegrasi dengan AWS CloudTrail, layanan yang menyediakan catatan tindakan yang diambil oleh pengguna, peran, atau AWS layanan di AmazonSNS. CloudTrail menangkap API panggilan untuk Amazon SNS sebagai acara. Panggilan yang diambil termasuk panggilan dari SNS konsol Amazon dan panggilan kode ke SNS API operasi Amazon. Jika Anda membuat jejak, Anda dapat mengaktifkan pengiriman CloudTrail acara secara berkelanjutan ke bucket Amazon S3, termasuk acara untuk Amazon. SNS Jika Anda tidak mengonfigurasi jejak, Anda masih dapat melihat peristiwa terbaru di CloudTrail konsol dalam Riwayat acara. Dengan menggunakan informasi yang dikumpulkan oleh CloudTrail, Anda dapat menentukan permintaan yang dibuat ke AmazonSNS, alamat IP dari mana permintaan itu dibuat, siapa yang membuat permintaan, kapan dibuat, dan detail tambahan.

Untuk mempelajari selengkapnya CloudTrail, termasuk cara mengonfigurasi dan mengaktifkannya, lihat Panduan AWS CloudTrail Pengguna.

SNSInformasi Amazon di CloudTrail

CloudTrail diaktifkan pada Akun AWS saat Anda membuat akun. Ketika aktivitas acara yang didukung terjadi di AmazonSNS, aktivitas tersebut direkam dalam suatu CloudTrail peristiwa bersama dengan peristiwa AWS layanan lainnya dalam riwayat Acara. Anda dapat melihat, mencari, dan mengunduh acara terbaru di situs Anda Akun AWS. Untuk informasi selengkapnya, lihat Melihat Acara dengan Riwayat CloudTrail Acara.

Untuk catatan acara yang sedang berlangsung di Anda Akun AWS, termasuk acara untuk AmazonSNS, buat jejak. Jejak memungkinkan CloudTrail untuk mengirimkan file log ke bucket Amazon S3. Secara default, saat Anda membuat jejak di konsol, jejak tersebut berlaku untuk semua AWS Wilayah. Jejak mencatat peristiwa dari semua Wilayah di AWS partisi dan mengirimkan file log ke bucket Amazon S3 yang Anda tentukan. Selain itu, Anda dapat mengonfigurasi AWS layanan lain untuk menganalisis lebih lanjut dan menindaklanjuti data peristiwa yang dikumpulkan dalam CloudTrail log. Untuk informasi selengkapnya, lihat berikut:

- Gambaran Umum untuk Membuat Jejak
- CloudTrail Layanan dan Integrasi yang Didukung
- Mengkonfigurasi SNS Pemberitahuan Amazon untuk CloudTrail
- Menerima File CloudTrail Log dari Beberapa Wilayah dan Menerima File CloudTrail Log dari Beberapa Akun

Mengontrol peristiwa pesawat di CloudTrail

Amazon SNS mendukung pencatatan tindakan berikut sebagai peristiwa dalam file CloudTrail log:

- 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**
>
> Ketika Anda tidak masuk ke Amazon Web Services (mode tidak diautentikasi) dan tindakan ConfirmSubscriptionatau Berhenti berlangganan dipanggil, maka tindakan tersebut tidak akan masuk ke log. CloudTrail Misalnya, ketika Anda memilih tautan yang disediakan dalam notifikasi email untuk mengonfirmasi langganan tertunda ke topik, tindakan `ConfirmSubscription` dipanggil dalam mode tidak terautentikasi. Dalam contoh ini, `ConfirmSubscription` tindakan tidak akan dicatat CloudTrail.

Setiap entri peristiwa atau log berisi informasi tentang siapa yang membuat permintaan tersebut. Informasi identitas membantu Anda menentukan berikut ini:

- Apakah permintaan dibuat dengan root atau AWS Identity and Access Management (IAM) kredensional pengguna.
- Apakah permintaan tersebut dibuat dengan kredensial keamanan sementara untuk satu peran atau pengguna terfederasi.
- Apakah permintaan itu dibuat oleh AWS layanan lain.

Untuk informasi selengkapnya, lihat CloudTrail userIdentity Elemen.

Peristiwa pesawat data di CloudTrail

Untuk mengaktifkan pencatatan API tindakan berikut dalam CloudTrail file, Anda harus mengaktifkan pencatatan API aktivitas bidang data CloudTrail. Untuk informasi selengkapnya, lihat Mencatat peristiwa data dalam AWS CloudTrail Panduan Pengguna.

Peristiwa bidang data juga dapat difilter berdasarkan jenis sumber daya, untuk kontrol terperinci atas SNS API panggilan Amazon yang ingin Anda log dan bayar secara selektif. CloudTrail Misalnya, dengan menentukan `AWS::SNS::Topic` sebagai jenis sumber daya, Anda dapat mencatat panggilan `Publish` dan `PublishBatch` API tindakan untuk topik. Demikian juga, dengan menentukan `AWS::SNS::PlatformEndpoint` sebagai tipe sumber daya, Anda dapat mencatat panggilan ke API tindakan Publikasikan untuk titik akhir platform. Untuk informasi lebih lanjut, lihat `AdvancedEventSelector`di AWS CloudTrail API Referensi.

> **Note**
>
> Jenis SNS sumber daya Amazon `AWS::SNS::PhoneNumber` tidak dicatat oleh CloudTrail.

Pesawat SNS data Amazon APIs

- `Publish`
- `PublishBatch`

Contoh: Entri file SNS log Amazon

Trail adalah konfigurasi yang memungkinkan pengiriman peristiwa sebagai file log ke bucket Amazon S3 yang Anda tentukan. CloudTrail file log berisi satu atau lebih entri log. Peristiwa merepresentasikan satu permintaan dari sumber apa pun dan menyertakan informasi tentang tindakan yang diminta, tanggal dan waktu tindakan, parameter permintaan, dan sebagainya. CloudTrail file log bukanlah jejak tumpukan yang diurutkan dari API panggilan publik, sehingga tidak muncul dalam urutan tertentu.

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan`ListTopics`,`CreateTopic`, dan `DeleteTopic` tindakan.

```
{
```

```
  "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"
    },
  ]
}
```

Contoh berikut menunjukkan entri CloudTrail log yang menunjukkan `Publish` dan `PublishBatch` tindakan.

Publikasikan

```
{
```

```
 "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"
 }
```

```
}
```

Memantau SNS topik Amazon menggunakan CloudWatch

Amazon SNS dan Amazon CloudWatch terintegrasi sehingga Anda dapat mengumpulkan, melihat, dan menganalisis metrik untuk setiap SNS notifikasi Amazon yang aktif. Setelah Anda mengonfigurasi CloudWatch untuk AmazonSNS, Anda dapat memperoleh wawasan yang lebih baik tentang kinerja SNS topik Amazon, pemberitahuan push, dan SMS pengiriman Anda. Misalnya, Anda dapat menyetel alarm untuk mengirimi Anda pemberitahuan email jika ambang batas yang ditentukan terpenuhi untuk SNS metrik Amazon, sepertiNumberOfNotificationsFailed. Untuk daftar semua metrik yang SNS dikirimkan Amazon CloudWatch, lihatMetrik Amazon SNS. Untuk informasi selengkapnya tentang notifikasi SNS push Amazon, lihatMengirim notifikasi push seluler dengan Amazon SNS.

> **Note**
>
> Metrik yang Anda konfigurasikan CloudWatch untuk SNS topik Amazon secara otomatis dikumpulkan dan didorong ke CloudWatch interval 1 menit. Metrik ini dikumpulkan pada semua topik yang memenuhi CloudWatch pedoman untuk aktif. Sebuah topik dianggap aktif hingga CloudWatch enam jam dari aktivitas terakhir (yaitu, API panggilan apa pun) pada topik tersebut.
>
> Tidak ada biaya untuk SNS metrik Amazon yang dilaporkan CloudWatch; mereka disediakan sebagai bagian dari SNS layanan Amazon.

Lihat CloudWatch metrik untuk Amazon SNS

Anda dapat memantau metrik untuk Amazon SNS menggunakan CloudWatch konsol, CloudWatch antarmuka baris perintah sendiri (CLI), atau secara terprogram menggunakan. CloudWatch API Prosedur berikut menunjukkan cara mengakses metrik menggunakan AWS Management Console.

Untuk melihat metrik menggunakan konsol CloudWatch

1. Masuk ke konsol CloudWatch tersebut.
2. Di panel navigasi, pilih Metrics (Metrik).
3. Pada tab Semua metrik, pilih SNS, lalu pilih salah satu dimensi berikut:

   - Negara, SMS Tipe

- PhoneNumber
- Metrik Topik
- Metrik tanpa dimensi

4. Untuk melihat detail lebih lanjut, pilih item tertentu. Misalnya, jika Anda memilih Metrik Topik dan kemudian memilih NumberOfMessagesPublished, jumlah rata-rata SNS pesan Amazon yang dipublikasikan untuk periode 1 menit selama rentang waktu 6 jam akan ditampilkan.
5. Untuk melihat metrik SNS penggunaan Amazon, pada tab Semua metrik, pilih Penggunaan, dan pilih metrik SNS penggunaan Amazon target (misalnya,`NumberOfMessagesPublishedPerAccount`).

Setel CloudWatch alarm untuk metrik Amazon SNS

CloudWatch juga memungkinkan Anda untuk mengatur alarm ketika ambang batas terpenuhi untuk metrik. Misalnya, Anda dapat mengatur alarm untuk metrik NumberOfNotificationsFailed, sehingga ketika nomor ambang batas yang Anda tentukan terpenuhi dalam periode pengambilan sampel, maka pemberitahuan email akan dikirim untuk memberi tahu Anda tentang peristiwa tersebut.

Untuk mengatur alarm menggunakan konsol CloudWatch

1. Masuk ke AWS Management Console dan buka CloudWatch konsol di https://console.aws.amazon.com/cloudwatch/.
2. Pilih Alarms (Alarm) lalu pilih tombol Create Alarm (Buat Alarm). Proses ini meluncurkan wizard Create Alarm (Buat Alarm).
3. Gulir SNS metrik Amazon untuk menemukan metrik yang ingin Anda gunakan alarm. Pilih metrik untuk membuat alarm aktif dan pilih Continue (Lanjutkan).
4. Isi nilai Name (Nama), Description (Deskripsi), Threshold (Ambang), dan Time (Waktu) untuk metrik, dan kemudian pilih Continue (Lanjutkan).
5. Pilih Alarm (Alarm) sebagai status alarm. Jika Anda CloudWatch ingin mengirimi Anda email saat status alarm tercapai, pilih salah satu SNS topik Amazon yang ada atau pilih Buat Topik Email Baru. Jika memilih Create New Email Topic (Buat Topik Email Baru), Anda dapat mengatur nama dan alamat email untuk topik baru. Daftar ini akan disimpan dan muncul di kotak drop-down untuk alarm di masa mendatang. Pilih Continue (Lanjutkan).

> **Note**
>
> Jika Anda menggunakan Buat Topik Email Baru untuk membuat SNS topik Amazon baru, alamat email harus diverifikasi sebelum menerima pemberitahuan. Email hanya dikirim saat alarm memasuki status alarm. Jika perubahan status alarm ini terjadi sebelum alamat email diverifikasi, alamat tidak akan menerima notifikasi.

6. Pada proses ini, wizard Create Alarm (Buat Alarm) memberi Anda kesempatan untuk meninjau alarm yang akan Anda buat. Jika Anda perlu melakukan perubahan, Anda dapat menggunakan tautan Edit di sebelah kanan. Setelah Anda puas, pilih Create Alarm (Buat Alarm).

Untuk informasi selengkapnya tentang penggunaan CloudWatch dan alarm, lihat CloudWatchDokumentasi.

Metrik Amazon SNS

Amazon SNS mengirimkan metrik berikut ke CloudWatch.

Namespace	Metrik	Deskripsi
AWS/SNS	`NumberOfMessagesPublished`	Jumlah pesan yang dipublikasikan ke SNS topik Amazon Anda. Unit: Hitung Dimensi yang valid: Aplikasi PhoneNumber, Platform, dan TopicName Statistik yang valid: Jumlah
AWS/SNS	`NumberOfNotificationsDelivered`	Jumlah pesan yang berhasil dikirim dari SNS topik Amazon Anda ke titik akhir berlangganan. Agar upaya pengiriman berhasil, langganan dari endpoint harus menerima pesan tersebut.

Namespace	Metrik	Deskripsi
		Langganan menerima pesan jika.) Tidak memiliki kebijakan filter atau b.) kebijakan filter mencakup atribut yang cocok dengan yang ditetapkan ke pesan. Jika langganan menolak pesan, upaya pengiriman tidak dihitung untuk metrik ini. Unit: Hitung Dimensi yang valid: Aplikasi PhoneNumber, Platform, dan TopicName Statistik yang valid: Jumlah

Namespace	Metrik	Deskripsi
AWS/SNS	`NumberOfNotificationsFailed`	Jumlah pesan yang SNS gagal dikirimkan Amazon. Untuk AmazonSQS, emailSMS, atau titik akhir push seluler, metrik bertambah 1 saat Amazon SNS berhenti mencoba pengiriman pesan. Untuk HTTP atau HTTPS titik akhir, metrik mencakup setiap upaya pengiriman yang gagal, termasuk percobaan ulang yang mengikuti upaya awal. Untuk semua endpoint lainnya, jumlah bertambah 1 ketika pesan gagal terkrim (terlepas dari jumlah upaya). Metrik ini tidak menyertakan pesan yang ditolak oleh kebijakan filter langganan. Anda dapat mengontrol jumlah percobaan ulang untuk titik HTTP akhir. Untuk informasi selengkap nya, lihat Mencoba lagi pengiriman SNS pesan Amazon. Unit: Hitung Dimensi yang valid: Aplikasi PhoneNumber, Platform, dan TopicName Statistik yang valid: Jumlah, Rata-rata

Namespace	Metrik	Deskripsi
AWS/SNS	`NumberOfNotificationsFilteredOut`	Jumlah pesan yang ditolak oleh kebijakan filter langganan. Kebijakan filter menolak pesan bila atribut pesan tidak cocok dengan atribut kebijakan. Unit: Hitung Dimensi yang valid: Aplikasi PhoneNumber, Platform, dan TopicName Statistik yang valid: Jumlah, Rata-rata
AWS/SNS	`NumberOfNotificationsFilteredOut-MessageAttributes`	Jumlah pesan yang ditolak oleh kebijakan filter langganan untuk pemfilteran berbasis atribut. Unit: Hitung Dimensi yang valid: Aplikasi PhoneNumber, Platform, dan TopicName Statistik yang valid: Jumlah, Rata-rata

Namespace	Metrik	Deskripsi
AWS/SNS	`NumberOfNotificationsFilteredOut-MessageBody`	Jumlah pesan yang ditolak oleh kebijakan filter langganan untuk pemfilteran berbasis muatan. Unit: Hitung Dimensi yang valid: Aplikasi PhoneNumber, Platform, dan TopicName Statistik yang valid: Jumlah, Rata-rata
AWS/SNS	`NumberOfNotificationsFilteredOut-InvalidAttributes`	Jumlah pesan yang ditolak oleh kebijakan filter langganan karena atribut pesan tidak valid — misalnya, karena atribut JSON salah diformat. Unit: Hitung Dimensi yang valid: Aplikasi PhoneNumber, Platform, dan TopicName Statistik yang valid: Jumlah, Rata-rata

Namespace	Metrik	Deskripsi
AWS/SNS	`NumberOfNotificationsFilteredOut-NoMessageAttributes`	Jumlah pesan yang ditolak oleh kebijakan filter langganan karena pesan tidak memiliki atribut. Unit: Hitung Dimensi yang valid: Aplikasi PhoneNumber, Platform, dan TopicName Statistik yang valid: Jumlah, Rata-rata
AWS/SNS	`NumberOfNotificationsFilteredOut-InvalidMessageBody`	Jumlah pesan yang ditolak oleh kebijakan filter langganan karena isi pesan tidak valid untuk pemfilteran — misalnya, badan pesan tidak validJSON. Unit: Hitung Dimensi yang valid: Aplikasi PhoneNumber, Platform, dan TopicName Statistik yang valid: Jumlah, Rata-rata

Namespace	Metrik	Deskripsi
AWS/SNS	`NumberOfNotificationsRedrivenToDlq`	Jumlah pesan yang telah dipindahkan ke antrean surat mati. Unit: Hitung Dimensi yang valid: Aplikasi PhoneNumber, Platform, dan TopicName Statistik yang valid: Jumlah, Rata-rata
AWS/SNS	`NumberOfNotificationsFailedToRedriveToDlq`	Jumlah pesan yang tidak dapat dipindahkan ke antrean surat mati. Unit: Hitung Dimensi yang valid: Aplikasi PhoneNumber, Platform, dan TopicName Statistik yang valid: Jumlah, Rata-rata
AWS/SNS	`PublishSize`	Ukuran pesan yang diterbitkan. Unit: Bytes Dimensi yang valid: Aplikasi PhoneNumber, Platform, dan TopicName Statistik yang valid: Minimum, Maksimum, Rata-rata dan Hitungan

Namespace	Metrik	Deskripsi
AWS/SNS	`SMSMonthToDateSpentUSD`	Biaya yang telah Anda bayar sejak awal bulan kalender saat ini untuk mengirim pesanSMS. Anda dapat mengatur alarm untuk metrik ini untuk mengetahui kapan month-to-date tagihan Anda mendekati kuota SMS belanja bulanan untuk akun Anda. Ketika Amazon SNS menentukan bahwa mengirim SMS pesan akan dikenakan biaya yang melebihi kuota ini, Amazon berhenti menerbitkan SMS pesan dalam beberapa menit. Untuk informasi tentang pengaturan kuota SMS belanja bulanan Anda, atau untuk informasi tentang meminta kenaikan kuota belanja dengan AWS, lihat. Mengatur preferensi SMS pesan di Amazon SNS Unit: USD Dimensi yang valid: Tidak ada Statistik yang valid: Jumlah

Namespace	Metrik	Deskripsi
AWS/SNS	`SMSSuccessRate`	Tingkat pengiriman SMS pesan yang berhasil. Unit: Hitung Dimensi yang valid: PhoneNumber Statistik yang valid: Jumlah, Rata-rata, Sampel Data

Dimensi untuk SNS metrik Amazon

Amazon Simple Notification Service mengirimkan dimensi berikut ke CloudWatch.

Dimensi	Deskripsi
`Application`	Filter pada objek aplikasi, yang mewakili aplikasi dan perangkat yang terdaftar di salah satu layanan pemberitahuan push yang didukung, seperti APNs danFCM.
`Application,Platform`	Filter pada objek aplikasi dan platform, di mana objek platform adalah untuk layanan pemberitahuan push yang didukung, seperti APNs danFCM.
`Country`	Filter pada negara tujuan atau wilayah SMS pesan. Negara atau wilayah diwakili oleh kode ISO alfa-2 3166-1.
`PhoneNumber`	Filter pada nomor telepon saat Anda mempublikasikan SMS langsung ke nomor telepon (tanpa topik).
`Platform`	Filter pada objek platform untuk layanan pemberitahuan push, seperti APNs danFCM.
`TopicName`	Filter pada nama SNS topik Amazon.
`SMSType`	Filter pada jenis pesan SMS pesan. Bisa promotional (promosi) atau transactional (transaksional).

Metrik SNS penggunaan Amazon

Amazon Simple Notification Service mengirimkan metrik penggunaan berikut ke CloudWatch.

Namespace	Layanan	Metrik	Sumber Daya	Tipe	Deskripsi
AWS/Penggunaan	SNS	`ResourceCount`	`NumberOfMessagesPublishedPerAccount`	Sumber Daya	• Jumlah pesan yang dipublikasikan ke SNS topik Amazon Anda di seluruh AWS akun Anda. • Satuan: Tidak ada • Statistik Valid: Sum
AWS/Penggunaan	SNS	`ResourceCount`	`ApproximateNumberOfTopics`	Sumber Daya	• Perkiraan jumlah topik di seluruh AWS akun Anda. • Satuan: Tidak ada • Statistik yang Valid: Rata-rata, Minimum, Maksimum, Jumlah

Namespace	Layanan	Metrik	Sumber Daya	Tipe	Deskripsi
AWS/Penggunaan	SNS	ResourceCount	ApproximateNumberOfFilterPolicies	Sumber Daya	• Perkiraan jumlah kebijakan filter di seluruh AWS akun Anda. • Satuan: Tidak ada • Statistik yang Valid: Rata-rata, Minimum, Maksimum, Jumlah
AWS/Penggunaan	SNS	ResourceCount	ApproximateNumberOfPendingSubscriptions	Sumber Daya	• Perkiraan jumlah langganan yang tertunda di seluruh AWS akun Anda. • Satuan: Tidak ada • Statistik yang Valid: Rata-rata, Minimum, Maksimum, Jumlah

Namespace	Layanan	Metrik	Sumber Daya	Tipe	Deskripsi
AWS/Penggunaan	SNS	`CallCount`	• `AddPermission` • `CheckIfPhoneNumberIsOptedOut` • `CreatePlatformApplication` • `CreatePlatformEndpoint` • `ConfirmSubscription` • `CreateSMSSandboxPhoneNumber` • `CreateTopic` • `DeleteEndpoint` • `DeletePlatformApplication` • `DeleteSMSSandboxPhoneNumber` • `DeleteTopic`	API	• Jumlah API panggilan untuk Amazon yang dipilih SNS API di seluruh AWS akun Anda. • Satuan: Tidak ada • Statistik Valid: Sum

Namespace	Layanan	Metrik	Sumber Daya	Tipe	Deskripsi
			• GetEndpointAttributes • GetPlatformApplicationAttributes • GetSMSAttributes • GetSMSSandboxAccountStatus • GetSubscriptionAttributes • GetTopicAttributes • ListEndpointsByPlatformApplication • ListOriginationNumbers • ListPhoneNumbersOptedOut • ListPlatformApplications		

Namespace	Layanan	Metrik	Sumber Daya	Tipe	Deskripsi
			• ListSMSSandboxPhoneNumbers • ListSubscriptions • ListSubscriptionsByTopic • ListTagsForResource • ListTopics • OptInPhoneNumber • RemovePermission • SetEndpointAttributes • SetPlatformApplicationAttributes • SetSMSAttributes • SetSubscriptionAttributes • SetTopicAttributes		

Namespace	Layanan	Metrik	Sumber Daya	Tipe	Deskripsi
			• Subscribe • Unsubscribe • UntagResource • VerifySMSSandboxPhoneNumber		

Validasi kepatuhan untuk Amazon SNS

Auditor pihak ketiga menilai keamanan dan kepatuhan Amazon SNS sebagai bagian dari beberapa program AWS kepatuhan, termasuk Undang-Undang Portabilitas dan Akuntabilitas Asuransi Kesehatan (). HIPAA

Untuk daftar AWS layanan dalam lingkup program kepatuhan tertentu, lihat AWS Layanan dalam Lingkup oleh AWS Layanan Program Kepatuhan . Untuk informasi umum, lihat Program AWS Kepatuhan Program AWS .

Anda dapat mengunduh laporan audit pihak ketiga menggunakan AWS Artifact. Untuk informasi selengkapnya, lihat Mengunduh Laporan di AWS Artifact .

Tanggung jawab kepatuhan Anda saat menggunakan Amazon SNS ditentukan oleh sensitivitas data Anda, tujuan kepatuhan perusahaan Anda, serta undang-undang dan peraturan yang berlaku. AWS menyediakan sumber daya berikut untuk membantu kepatuhan:

- Panduan Memulai Cepat Keamanan dan Kepatuhan Panduan Keamanan dan Kepatuhan — Panduan penerapan ini membahas pertimbangan arsitektur dan memberikan langkah-langkah untuk menerapkan lingkungan dasar yang berfokus pada keamanan dan kepatuhan. AWS
- Arsitektur untuk HIPAA Keamanan dan Kepatuhan Whitepaper — Whitepaper ini menjelaskan bagaimana perusahaan dapat menggunakan AWS untuk membuat aplikasi yang sesuai. HIPAA
- AWS Sumber Daya AWS — Kumpulan buku kerja dan panduan ini mungkin berlaku untuk industri dan lokasi Anda.

- Mengevaluasi Sumber Daya dengan Aturan dalam Panduan AWS Config Pengembang — AWS Config Layanan menilai seberapa baik konfigurasi sumber daya Anda mematuhi praktik internal, pedoman industri, dan peraturan.
- AWS Security Hub— AWS Layanan ini memberikan pandangan komprehensif tentang keadaan keamanan Anda di dalamnya AWS yang membantu Anda memeriksa kepatuhan Anda terhadap standar industri keamanan dan praktik terbaik.

Ketahanan di Amazon SNS

Ketahanan di Amazon SNS dipastikan melalui pemanfaatan infrastruktur AWS global, yang berputar di sekitar dan Availability Zone. Wilayah AWS Wilayah AWS menawarkan Availability Zone yang terpisah secara fisik dan terisolasi yang dihubungkan oleh latensi rendah, throughput tinggi, dan jaringan yang sangat redundan. Arsitektur ini memungkinkan failover tanpa batas antara Availability Zones tanpa gangguan, membuat aplikasi dan database secara inheren lebih toleran terhadap kesalahan dan skalabel dibandingkan dengan infrastruktur pusat data tradisional. Dengan menggunakan Availability Zones, SNS pelanggan Amazon mendapat manfaat dari peningkatan ketersediaan dan keandalan, menjamin pengiriman pesan meskipun ada potensi gangguan. Untuk informasi selengkapnya tentang Wilayah AWS dan Availability Zone, lihat Infrastruktur AWS Global.

Selain itu, langganan ke SNS topik Amazon dapat dikonfigurasi dengan percobaan ulang pengiriman dan antrian surat mati, memungkinkan penanganan otomatis kegagalan sementara dan memastikan pesan mencapai tujuan yang diinginkan dengan andal.

Amazon SNS juga mendukung pemfilteran pesan dan atribut pesan, yang memungkinkan Anda menyesuaikan strategi ketahanan dengan kasus penggunaan spesifiknya, meningkatkan ketahanan keseluruhan aplikasi Anda.

Keamanan infrastruktur di Amazon SNS

Sebagai layanan terkelola, Amazon SNS dilindungi oleh prosedur keamanan jaringan AWS global yang terdapat dalam dokumentasi Praktik Terbaik untuk Keamanan, Identitas, & Kepatuhan.

Gunakan AWS API tindakan untuk mengakses Amazon SNS melalui jaringan. Klien harus mendukung Transport Layer Security (TLS) 1.2 atau yang lebih baru. Klien juga harus mendukung cipher suite dengan Perfect Forward Secrecy (PFS), seperti Ephemeral Diffie-Hellman () atau Elliptic Curve Ephemeral Diffie-Hellman (). DHE ECDHE

Anda harus menandatangani permintaan menggunakan ID kunci akses dan kunci akses rahasia yang terkait dengan IAM prinsipal. Atau, Anda dapat menggunakanAWS Security Token Service (AWS STS) untuk menghasilkan kredensial keamanan sementara untuk permintaan penandatanganan.

Anda dapat memanggil API tindakan ini dari lokasi jaringan mana pun, tetapi Amazon SNS mendukung kebijakan akses berbasis sumber daya, yang dapat mencakup pembatasan berdasarkan alamat IP sumber. Anda juga dapat menggunakan SNS kebijakan Amazon untuk mengontrol akses dari VPC titik akhir Amazon tertentu atau spesifikVPCs. Ini secara efektif mengisolasi akses jaringan ke SNS topik Amazon tertentu dari hanya spesifik VPC dalam AWS jaringan. Untuk informasi selengkapnya, lihat Batasi publikasi ke SNS topik Amazon hanya dari titik akhir tertentu VPC.

Memecahkan masalah topik Amazon SNS

Pelajari cara menggunakan AWS X-Ray untuk memecahkan masalah SNS topik Amazon dengan menelusuri dan menganalisis pesan, mengidentifikasi masalah, dan mengoptimalkan kinerja melalui data permintaan dan respons terperinci.

Memecahkan masalah topik Amazon menggunakan SNS AWS X-Ray

AWS X-Ray mengumpulkan data tentang permintaan yang disajikan aplikasi Anda, dan memungkinkan Anda melihat dan memfilter data untuk mengidentifikasi potensi masalah dan peluang untuk pengoptimalan. Untuk setiap permintaan yang dilacak ke aplikasi Anda, Anda dapat melihat informasi terperinci tentang permintaan, respons, dan panggilan yang dilakukan aplikasi Anda ke AWS sumber daya hilir, layanan mikro, database, dan web. HTTP APIs

Anda dapat menggunakan X-Ray dengan Amazon SNS untuk melacak dan menganalisis pesan yang melakukan perjalanan melalui aplikasi Anda. Anda dapat menggunakan AWS Management Console untuk melihat peta koneksi antara Amazon SNS dan layanan lain yang digunakan aplikasi Anda. Anda juga dapat menggunakan konsol tersebut untuk melihat metrik seperti tingkat latensi dan kegagalan rata-rata. Untuk informasi selengkapnya, lihat Amazon SNS dan AWS X-Ray di Panduan AWS X-Ray Pengembang.

Penelusuran aktif di Amazon SNS

Anda dapat menggunakan AWS X-Ray untuk melacak dan menganalisis permintaan pengguna saat mereka melakukan perjalanan melalui SNS topik Amazon Anda ke langganan Amazon Data Firehose, AWS LambdaAmazon SQS, dan HTTP/S endpoint Anda. Karena X-Ray memberi Anda end-to-end tampilan seluruh permintaan, Anda dapat melihat apa yang memanggil SNS topik Amazon Anda, dan apa yang menjadi bagian hilir langganan topik Anda. Anda dapat menganalisis latensi dalam pesan Anda dan layanan backend mereka (misalnya, berapa lama permintaan dihabiskan dalam suatu topik, dan berapa lama waktu yang dibutuhkan untuk mengirimkan pesan ke setiap langganan topik).

> **Important**
>
> SNSTopik Amazon dengan banyak langganan dapat mencapai batas ukuran dan tidak sepenuhnya dilacak. Untuk informasi tentang batas ukuran dokumen jejak, lihat kuota layanan sinar-X di Referensi AWS Umum.

Jika Anda memanggil Amazon SNS API dari layanan yang sudah dilacak, Amazon SNS melewati jejaknya, bahkan jika penelusuran X-Ray tidak diaktifkan di file. API

Amazon SNS mendukung penelusuran X-Ray untuk standar dan FIFO topik. Anda dapat mengaktifkan X-Ray untuk SNS topik Amazon dengan menggunakan SNSkonsol Amazon, Amazon SNS `SetTopicAttributes` API, CLI Referensi Layanan Pemberitahuan Sederhana Amazon, atau AWS CloudFormation.

Untuk mempelajari lebih lanjut tentang menggunakan Amazon SNS dengan X-Ray, lihat Amazon SNS dan AWS X-Ray di Panduan AWS X-Ray Pengembang.

Topik

- Izin penelusuran aktif
- Mengaktifkan penelusuran aktif pada SNS topik Amazon menggunakan konsol AWS
- Mengaktifkan penelusuran aktif pada SNS topik Amazon menggunakan AWS SDK
- Mengaktifkan penelusuran aktif pada SNS topik Amazon menggunakan AWS CLI
- Mengaktifkan penelusuran aktif pada topik Amazon menggunakan SNS AWS CloudFormation
- Memverifikasi penelusuran aktif diaktifkan untuk topik Anda
- Menguji penelusuran aktif

Izin penelusuran aktif

Saat menggunakan SNS konsol Amazon, Amazon SNS mencoba membuat izin yang diperlukan untuk SNS topik Amazon untuk memanggil X-Ray. Upaya dapat ditolak jika Anda tidak memiliki izin yang cukup untuk menggunakan SNS konsol Amazon. Untuk informasi selengkapnya, silakan lihat Manajemen identitas dan akses di Amazon SNS dan Contoh kasus untuk kontrol SNS akses Amazon.

Saat menggunakanCLI, Anda harus mengonfigurasi izin secara manual. Izin tersebut dikonfigurasi menggunakan kebijakan sumber daya. Untuk informasi lebih lanjut tentang penggunaan izin yang diperlukan di X-Ray, lihat Amazon SNS dan AWS X-Ray.

Mengaktifkan penelusuran aktif pada SNS topik Amazon menggunakan konsol AWS

Saat penelusuran aktif diaktifkan pada SNS topik Amazon, ia membaca ID jejak, mengirimkan data ke pelanggan berdasarkan ID jejak, dan menyebarkan ID jejak ke layanan hilir.

1. Masuk ke SNSkonsol Amazon.
2. Pilih topik atau buat yang baru. Untuk detail selengkapnya tentang membuat topik, lihat Membuat SNS topik Amazon.
3. Pada halaman Buat topik, di bagian Detail, pilih jenis topik: FIFOatau Standar.
    a. Masukkan Nama untuk topik.
    b. (Opsional) Masukkan Nama tampilan untuk topik.
4. Perluas Penelusuran aktif, dan pilih Gunakan penelusuran aktif.

Setelah mengaktifkan X-Ray untuk SNS topik Amazon, Anda dapat menggunakan peta layanan X-Ray untuk melihat end-to-end jejak dan peta layanan untuk topik tersebut.

Mengaktifkan penelusuran aktif pada SNS topik Amazon menggunakan AWS SDK

Contoh kode berikut menunjukkan cara mengaktifkan penelusuran aktif pada SNS topik Amazon dengan menggunakan AWS SDK untuk Java.

```
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());
        }
    }
```

Mengaktifkan penelusuran aktif pada SNS topik Amazon menggunakan AWS CLI

Contoh kode berikut menunjukkan cara mengaktifkan penelusuran aktif pada SNS topik Amazon dengan menggunakan. AWS CLI

```
aws sns set-topic-attributes \
    --topic-arn arn:aws:sns:us-west-2:123456789012:MyTopic \
    --attribute-name TracingConfig \
    --attribute-value Active
```

Mengaktifkan penelusuran aktif pada topik Amazon menggunakan SNS AWS CloudFormation

AWS CloudFormation Tumpukan berikut menunjukkan cara mengaktifkan penelusuran aktif pada SNS topik Amazon.

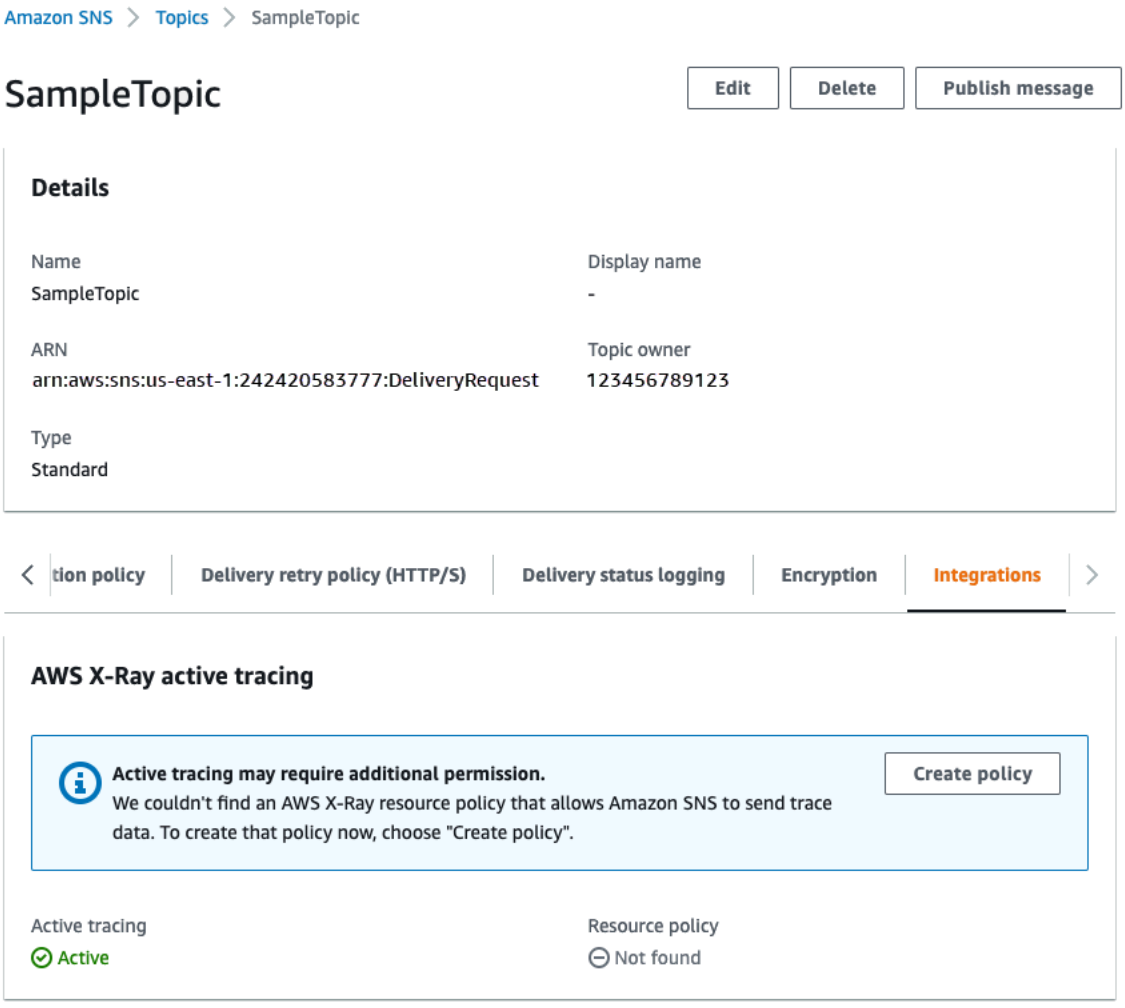
```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  MyTopicResource:
    Type: 'AWS::SNS::Topic'
    Properties:
      TopicName: 'MyTopic'
      TracingConfig: 'Active'
```

Memverifikasi penelusuran aktif diaktifkan untuk topik Anda

Anda dapat menggunakan SNS konsol Amazon untuk memverifikasi apakah penelusuran aktif diaktifkan untuk topik Anda, atau bila kebijakan sumber daya gagal ditambahkan.

1. Masuk ke SNSkonsol Amazon.
2. Di panel navigasi kiri, pilih Topics (Topik).
3. Pada halaman Topik, pilih topik.
4. Pilih tab Integrasi.

   Saat penelusuran aktif diaktifkan, ikon Aktif berwarna hijau ditampilkan.
5. Jika Anda telah mengaktifkan penelusuran aktif dan Anda tidak melihat bahwa kebijakan sumber daya telah ditambahkan, pilih Buat kebijakan untuk menambahkan izin tambahan yang diperlukan.



Menguji penelusuran aktif

1. Masuk ke SNSkonsol Amazon.
2. Buat SNS topik Amazon. Untuk detail tentang cara melakukannya, lihat Untuk membuat topik menggunakan AWS Management Console.
3. Perluas Penelusuran aktif, dan pilih Gunakan penelusuran aktif.
4. Publikasikan pesan ke SNS topik Amazon. Untuk detail tentang cara melakukannya, lihat Untuk mempublikasikan pesan ke SNS topik Amazon menggunakan AWS Management Console.
5. Gunakan peta layanan X-Ray untuk melihat end-to-end jejak dan peta layanan untuk topik tersebut.

Riwayat SNS dokumentasi Amazon

Tabel berikut menjelaskan perubahan terbaru untuk Panduan Developer Amazon Simple Notification Service.

Fitur layanan terkadang diluncurkan secara bertahap ke AWS Wilayah tempat layanan tersedia. Kami memperbarui dokumentasi ini hanya untuk rilis pertama. Kami tidak memberikan informasi tentang ketersediaan Wilayah atau mengumumkan peluncuran Wilayah berikutnya. Untuk informasi tentang ketersediaan fitur layanan wilayah dan untuk berlangganan pemberitahuan tentang pembaruan, lihat Apa yang Baru dengan AWS? .

Perubahan	Deskripsi	Tanggal
`AmazonSNSFullAccess` dan pembaruan kebijakan `AmazonSNSReadOnlyAccess` terkelola	Amazon SNS menambahkan izin baru AmazonSNSFullAccess and AmazonSNSReadOnlyAccess kebijakan terkelola, yang memungkinkan akses tambahan ke Amazon SNS melalui AWS Management Console.	September 24, 2024
SNSIntegrasi AWS Olah Pesan Pengguna Akhir SMS Amazon dengan pengiriman SMS pesan	SNSPelanggan Amazon dapat menggunakan fitur baru seperti manajemen SMS sumber daya, pesan dua arah, izin sumber daya terperinci, aturan pemblokiran negara, dan penagihan terpusat untuk semua AWS SMS pesan tanpa membuat perubahan apa pun pada konfigurasi atau jaringan global yang digunakan oleh Amazon. AWS SMS SNS	September 24, 2024

Kanada Barat (Calgary) dukungan untuk topik FIFO	Amazon SNS mendukung FIFO topik di Kanada Barat (Calgary).	Maret 28, 2024
SNSSMSDukungan Amazon di lima wilayah baru	Amazon SNS menambahk an SMS dukungan ke wilayah berikut: Asia Pasifik (Hyderaba d), Asia Pasifik (Melbourne), Timur Tengah (UAE), Eropa (Zurich). dan Eropa (Spanyol).	Februari 8, 2024
Dukungan Firebase Cloud Messaging (FCM) HTTP v1	Amazon SNS mendukung kredensi FCM v1.	Januari 18, 2024
Amazon SNS SMS didukung di Asia Pasifik (Jakarta)	Amazon SNS mendukung SMS pengiriman pesan di Asia Pasifik (Jakarta).	14 Desember 2023
AWS CloudFormation dukungan untuk mengonfig urasi `DeliveryS tatusLogging` topik Amazon SNS	AWS CloudFormation dukungan tersedia untuk mengonfigurasi `DeliveryS tatusLogging` saat membuat atau memperbarui SNS topik Amazon.	Desember 7, 2023
Operator penyaringan pesan baru ditambahkan	Sekarang Anda dapat menggunakan pencocoka n akhiran, kasus equals-ig nore, dan operator OR saat memfilter pesan Amazon. SNS	16 November 2023

Support ditambahkan untuk pengarsipan dan pemutaran ulang pesan	Pemilik topik dapat mengarsipkan pesan ke topik hingga 365 hari. Pelanggan topik dapat memutar ulang pesan yang diarsipkan kembali ke titik akhir berlangganan untuk memulihkan pesan karena kegagalan dalam aplikasi hilir, atau untuk mereplikasi status aplikasi yang ada.	26 Oktober 2023
Support ditambahkan untuk berlangganan antrian standar ke topik FIFO	Anda dapat berlangganan SQS FIFO antrian Amazon atau antrian standar ke topik Amazon SNSFIFO. Hanya SQS FIFO antrian Amazon yang menjamin pesan diterima secara berurutan dan tanpa duplikat.	14 September 2023
SMSdukungan ditambahkan untuk Israel (Tel Aviv)	Amazon sekarang SNS SMS didukung di wilayah Israel (Tel Aviv).	28 Agustus 2023
Support untuk penelusuran aktif X-Ray ditambahkan ke topik FIFO	Sebelumnya hanya didukung dengan topik SNS standar Amazon, AWS X-Ray sekarang melacak dan menganalisis permintaan pengguna saat mereka melakukan perjalanan melalui FIFO topik Anda ke langganan Amazon Data Firehose AWS Lambda, SQS Amazon, HTTP dan /S endpoint Anda.	31 Mei 2023

Dukungan header Content-Type yang disempurnakan	Anda dapat menyetel header Content-Type dalam kebijakan permintaan untuk menentukan jenis media notifikasi.	Maret 23, 2023
Dukungan penelusuran aktif ditambahkan	AWS X-Ray melacak dan menganalisis permintaan pengguna saat mereka melakukan perjalanan melalui topik SNS standar Amazon Anda ke langganan Amazon Data Firehose AWS Lambda, SQS Amazon, HTTP dan /S endpoint Anda.	Februari 8, 2023
Pendaftaran ID Pengirim Singapura	Petunjuk ditambahkan untuk mendaftarkan Pengirim IDs di Singapura.	10 Januari 2023
Pemfilteran pesan berbasis payload	Pemfilteran berbasis payload memungkinkan Anda memfilter pesan berdasarkan muatan pesan dan menghindari biaya yang terkait dengan pemrosesan data yang tidak diinginkan.	22 November 2022
SHA256algoritma hash ditambahkan untuk penandatanganan SNS pesan Amazon	Support ditambahkan untuk algoritma SHA256 hash saat menggunakan penandatanganan SNS pesan Amazon.	15 September 2022

Wilayah tambahan ditambahkan ke SMS pesan	Amazon SNS mendukung SMS pengiriman pesan di wilayah berikut: Afrika (Cape Town), Asia Pasifik (Osaka), Eropa (Milan) dan AWS GovCloud (AS-Timur).	9 September 2022
Dukungan perlindungan data pesan ditambahkan	Perlindungan data pesan melindungi data yang dipublikasikan ke SNS topik Amazon Anda dengan menggunakan kebijakan perlindungan data untuk mengaudit dan memblokir informasi sensitif yang berpindah antar aplikasi atau layanan. AWS	8 September 2022
Proses pendaftaran baru untuk nomor bebas pulsa	Support ditambahkan untuk mengirim SNS pesan Amazon menggunakan nomor telepon bebas pulsa (TFN) ke penerima Amerika Serikat.	1 Agustus 2022

Support untuk kontrol akses berbasis Atribut () ABAC	Menambahkan dukungan untuk kontrol akses berbasis atribut (ABAC) untuk API tindakan termasuk `Publish` dan. `PublishBatch` ABACadalah strategi otorisasi yang mendefinisikan izin akses berdasarkan tag yang dapat dilampirkan ke IAM sumber daya, seperti IAM pengguna dan peran, dan sumber AWS daya, seperti SNS topik Amazon, untuk menyederhanakan manajemen izin.	10 Januari 2022
Support untuk otentikasi berbasis token Apple untuk pemberitahuan push	Anda dapat mengizinkan Amazon SNS untuk mengirim pemberitahuan push ke aplikasi iOS atau macOS Anda dengan memberikan informasi yang mengidentifikasi Anda sebagai pengembang aplikasi.	28 Oktober 2021
Pengirim SMS pesan baru ditempatkan di kotak pasir SMS	SMSKotak pasir ada untuk membantu mencegah penipuan dan penyalahg unaan, dan untuk membantu melindungi reputasi Anda sebagai pengirim. Saat AWS akun Anda berada di SMS kotak pasir, Anda hanya dapat mengirim SMS pesan ke nomor telepon tujuan yang diverifikasi.	1 Juni 2021

Pengirim SMS pesan baru ditempatkan di kotak pasir SMS	SMSKotak pasir ada untuk membantu mencegah penipuan dan penyalahg unaan, dan untuk membantu melindungi reputasi Anda sebagai pengirim. Saat AWS akun Anda berada di SMS kotak pasir, Anda hanya dapat mengirim SMS pesan ke nomor telepon tujuan yang diverifikasi.	1 Juni 2021
Atribut baru untuk mengirim SMS pesan ke penerima di India	Dua atribut baru, Entity ID dan Template ID, sekarang diperlukan untuk mengirim SMS pesan ke penerima di India.	22 April 2021
Pembaruan untuk operator penyaringan pesan	Operator baru, `cidr`, tersedia untuk mencocokkan alamat IP dan subnet sumber pesan. Anda sekarang juga dapat memeriksa tidak adanya atribut kunci, dan menggunak an prefiks dengan operator `anything-but` untuk pencocokan string atribut.	7 April 2021

Mengakhiri dukungan untuk kode panjang P2P untuk tujuan AS	Efektif 1 Juni 2021, penyedia telekomunikasi AS tidak lagi mendukung penggunaan kode panjang person-to-person (P2P) untuk komunikasi application-to-person (A2P) ke tujuan AS. Sebagai gantinya, Anda dapat menggunakan kode pendek, nomor bebas pulsa, atau jenis nomor originasi baru yang disebut 10. DLC	16 Februari 2021
Dukungan untuk metrik Amazon CloudWatch 1 menit	CloudWatch Metrik 1 menit untuk Amazon sekarang SNS tersedia di semua AWS Wilayah.	28 Januari 2021
Dukungan untuk titik akhir Amazon Data Firehose	Anda dapat berlangganan aliran pengiriman Firehose ke topik. SNS Ini memungkinkan Anda mengirim pemberitahuan ke titik akhir pengarsipan dan analitik seperti bucket Amazon Simple Storage Service (Amazon S3), tabel Amazon Redshift, Amazon Service (Layanan), OpenSearch dan banyak lagi. OpenSearch	12 Januari 2021
Nomor originasi tersedia	Anda dapat menggunakan nomor originasi saat mengirim pesan teks (SMS).	23 Oktober 2020

Support untuk SNS FIFO topik Amazon	Untuk mengintegrasikan aplikasi terdistribusi yang memerlukan konsistensi data dalam waktu nyaris nyata, Anda dapat menggunakan FIFO topik Amazon SNS first-in, first-out () dengan antrian Amazon. SQS FIFO	22 Oktober 2020
Perpustakaan Klien Amazon SNS Extended untuk Java tersedia	Anda dapat menggunak an perpustakaan ini untuk mempublikasikan SNS pesan Amazon yang besar.	25 Agustus 2020
SSEtersedia di Wilayah China	Enkripsi sisi server () SSE untuk Amazon SNS tersedia di Wilayah China.	20 Januari 2020
Support untuk menggunak an DLQs untuk menangkap pesan yang tidak terkirim	Untuk menangkap pesan yang tidak terkirim, Anda dapat menggunakan antrean SQS surat mati Amazon () DLQ dengan langganan Amazon. SNS	14 November 2019
Support untuk menentukan nilai APNs header kustom	Anda dapat menentukan nilai APNs header kustom.	18 Oktober 2019
Support untuk bidang header `apns-push-type "` untuk APNs	Anda dapat menggunakan bidang `apns-push-type` header untuk notifikasi seluler yang dikirimAPNs.	10 September 2019
Support untuk pemecahan masalah topik menggunakan AWS X-Ray	Anda dapat menggunakan X-Ray untuk memecahkan masalah pesan yang melewati SNS topik.	24 Juli 2019

Support untuk pencocokan kunci atribut menggunakan operator `exists "`	Untuk memeriksa apakah pesan masuk memiliki atribut kunci yang tercantum dalam kebijakan filter, Anda dapat menggunakan operator `exists`.	5 Juli 2019
Support untuk apa pun-kecuali pencocokan beberapa nilai numerik	Selain beberapa string, Amazon SNS memungkinkan apa pun - kecuali pencocokan beberapa nilai numerik.	5 Juli 2019
Catatan SNS rilis Amazon tersedia sebagai RSS umpan	Mengikuti judul di halaman ini (Riwayat dokumentasi), pilih RSS.	22 Juni 2019

Terjemahan disediakan oleh mesin penerjemah. Jika konten terjemahan yang diberikan bertentangan dengan versi bahasa Inggris aslinya, utamakan versi bahasa Inggris.